

Dr. Joseph Teguh Santoso, S.Kom, M.Kom.

Sistem Kecerdasan Buatan (AI) dan Big Data



YAYASAN PRIMA AGUS TEKNIK



Sistem Kecerdasan Buatan (AI) dan Big Data

Dr. Joseph Teguh Santoso, S.Kom, M.Kom.



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-634-7695-46-8 (PDF)



9

786347

695468

Sistem Kecerdasan Buatan (AI) dan Big Data

Penulis :

Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

ISBN : 978-634-7695-46-8 (PDF)

Editor :

Dr. Agus Wibowo, M.Kom, M.Si, MM.

Penyunting :

Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :

Irdha Yuniarto, S.Ds., M.Kom

Penebit :

Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Anggota IKAPI No: 279 / ALB / JTE / 2023

Redaksi :

Jl. Majapahit no 605 Semarang

Telp. 08122925000

Fax. 024-6710144

Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :

Universitas STEKOM

Jl. Majapahit no 605 Semarang

Telp. 08122925000

Fax. 024-6710144

Email : info@stekom.ac.id

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara
apapun tanpa ijin dari penulis

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga buku ini dapat disusun dan dihadirkan kepada para pembaca. Buku ini disusun sebagai bahan ajar yang bertujuan memberikan pemahaman yang sistematis, menyeluruh, dan kontekstual mengenai kecerdasan artifisial (*Artificial Intelligence/AI*) dan Big Data, dua bidang yang saat ini memiliki peran sangat penting dalam perkembangan ilmu pengetahuan, teknologi, industri, dan kehidupan masyarakat modern. Kehadiran buku ini diharapkan dapat menjadi sarana pembelajaran yang membantu pembaca memahami keterkaitan antara data, algoritme, sistem komputasi, dan penerapan teknologi cerdas secara lebih terarah.

Buku ini disusun secara bertahap, dimulai dari konsep dasar hingga topik yang lebih lanjut, agar pembaca dapat mengikuti alur pembahasan dengan lebih mudah. Bab 1 mengawali pembahasan dengan pengenalan konsep AI dan Big Data, karakteristik utama Big Data, urgensi AI dalam mengolah data berskala besar, serta berbagai aplikasi AI dalam kehidupan sehari-hari. Pada bab ini juga diperkenalkan dasar-dasar bahasa pemrograman Python sebagai salah satu bahasa utama yang banyak digunakan dalam pengembangan AI dan analisis data. Selanjutnya, Bab 2 membahas sistem Big Data dan sistem AI, termasuk arsitektur, komponen utama, serta cara kerja teknologi pendukungnya. Pembaca diperkenalkan pada ekosistem pemrosesan data modern, teknik pengumpulan data, dan *web scraping* sebagai salah satu metode memperoleh data dari sumber terbuka. Bab 3 kemudian memaparkan konsep data pipeline, yaitu alur pengelolaan data dari tahap pengumpulan, pemasukan, transformasi, analisis, hingga penyajian hasil, lengkap dengan gambaran penerapannya pada perusahaan-perusahaan digital.

Pada Bab 4, buku ini membahas paradigma pemrosesan data, yaitu *batch processing*, *stream processing*, dan *hybrid processing*, serta teknologi yang mendukungnya. Bab ini juga menekankan pentingnya data cleaning dan proses ETL (*Extract, Transform, Load*) sebagai tahap penting dalam menyiapkan data yang berkualitas. Pembahasan tersebut diperdalam dalam Bab 5, yang secara khusus mengulas pembangunan pipeline pembersihan data, mulai dari identifikasi nilai hilang, deteksi outlier, pemeriksaan kualitas data, hingga penyusunan alur prapemrosesan secara sistematis. Sebagai pengayaan materi, Bab 6 memperkenalkan Swarm Intelligence dan Particle Swarm Optimization sebagai pendekatan optimasi yang terinspirasi dari perilaku kolektif makhluk hidup. Bab 7 melanjutkan pembahasan ke Genetic Programming dan Genetic Algorithm, yaitu metode komputasi evolusioner yang meniru prinsip seleksi alam untuk menemukan solusi terbaik. Kedua bab ini memberikan wawasan bahwa pengembangan AI tidak hanya bertumpu pada satu pendekatan, melainkan mencakup beragam metode yang dapat diterapkan sesuai karakteristik masalah.

Pada Bab 8, pembaca diperkenalkan pada Machine Learning dan jaringan saraf tiruan, termasuk konsep neuron, bobot, fungsi aktivasi, *forward propagation*, dan *backpropagation*. Bab ini juga membahas berbagai arsitektur penting, seperti MLP, CNN, RNN, LSTM, dan

Transformer, yang menjadi fondasi bagi banyak sistem AI modern. Bab 9 membahas Reinforcement Learning, yaitu pembelajaran berbasis interaksi antara agen dan lingkungan, dengan konsep-konsep seperti *state*, *action*, *reward*, *policy*, dan *value function*. Pembahasan ini dilanjutkan pada Bab 10 yang mengulas Deep Reinforcement Learning, termasuk Q-Learning, Policy Gradient, Actor-Critic, dan Deep Q-Network sebagai pendekatan yang lebih canggih dalam pengambilan keputusan.

Dalam ranah pengolahan bahasa, Bab 11 membahas Natural Language Processing (NLP), tahapan analisis bahasa, serta teknik-teknik dasar dalam pemrosesan teks. Bab 12 memperdalam topik ini melalui pembahasan word embeddings, seperti Word2Vec, GloVe, FastText, serta embedding kontekstual seperti BERT dan GPT yang sangat berpengaruh dalam perkembangan NLP modern. Selanjutnya, Bab 13 mengulas sistem rekomendasi, meliputi *collaborative filtering*, *content-based filtering*, sistem hibrida, serta berbagai tantangan yang muncul dalam personalisasi layanan digital. Pada Bab 14, buku ini membahas analisis deret waktu, termasuk komponen-komponennya, model peramalan, dan teknik deteksi anomali yang banyak digunakan dalam berbagai bidang. Sebagai penutup, Bab 15 menyajikan pembahasan umum mengenai Deep Learning, mencakup arsitektur jaringan saraf mendalam, proses pelatihan model, kerangka kerja yang umum digunakan, serta berbagai aplikasinya dalam visi komputer, pemrosesan bahasa, pengenalan suara, dan sistem cerdas lainnya.

Secara keseluruhan, buku ini diharapkan dapat menjadi rujukan yang membantu pembaca memahami AI dan Big Data secara lebih terarah, baik dari sisi konsep, metode, maupun penerapannya. Buku ini juga diharapkan mampu mendorong tumbuhnya minat belajar, sikap kritis, dan semangat pengembangan inovasi di bidang teknologi cerdas berbasis data. Kami menyadari bahwa buku ini masih memiliki keterbatasan. Oleh karena itu, saran dan masukan yang membangun sangat diharapkan demi penyempurnaan pada masa mendatang. Semoga buku ini memberikan manfaat bagi pengembangan pembelajaran, penelitian, dan kajian akademik di bidang kecerdasan artifisial dan Big Data.

Semarang, Agustus 2026

Dr. Joseph Teguh Santoso, S.Kom., M.Kom

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI	iii
BAB 1 KONSEP FUNDAMENTAL DALAM AI	1
1.1 Pendahuluan.....	1
1.2 Big Data.....	1
1.3 Mengapa Kita Membutuhkan AI?.....	5
1.4 Aplikasi AI	5
1.5 Bahasa Pemrograman untuk AI	6
1.6 Dasar Pemrograman Python	7
1.7 Ringkasan.....	18
1.8 Pertanyaan.....	18
BAB 2 SISTEM BIG DATA DAN AI	20
2.1 Sistem Big Data	20
2.2 Komponen Utama Big Data	20
2.3 Gambaran Umum Sistem AI	22
2.4 Bagaimana Sistem AI Bekerja	23
2.5 Evolusi Sistem AI	24
2.6 Kecerdasan Manusia Vs AI.....	24
2.7 Aplikasi Untuk Sistem AI	25
2.8 Pengumpulan Data	26
2.9 Apa Itu Beautiful Soup.....	27
2.10 Contoh Web Scraping	28
2.11 Ringkasan.....	35
2.12 Pertanyaan.....	35
BAB 3 PERANCANGAN ALUR PIPELINE BIG DATA	37
3.1 Konsep Alur Data	37
3.2 Arsitektur Lambda	38
3.3 Arsitektur Pipeline Twitter.....	39
3.4 Arsitektur Pipeline Pinterest.....	41
3.5 Arsitektur Pipeline Airbnb	42
3.6 Strategi Membuat Pipeline Big Data	43
3.7 Klasifikasi Emosi Dengan Spark Real-Time	45
3.8 Ringkasan.....	48
3.9 Pertanyaan.....	49
3.10 Tugas Pemrograman	49
BAB 4 TEKNOLOGI BIG DATA DAN KUALITAS DATA.....	51
4.1 Sistem Big Data	51
4.2 Sistem Big Data Hadoop	51

4.3	Kerangka Kerja Pemrosesan Aliran	53
4.4	Sistem Pemrosesan Hibrida	54
4.5	Apache Flink Dan Apache Samza	55
4.6	Pendahuluan Tentang Pembersihan Data	56
4.7	Ekstraksi, Transformasi, Dan Pemuatan	58
4.8	Klasifikasi Anomali Data.....	60
4.9	Masalah Kualitas Data	62
4.10	Ringkasan.....	65
4.11	Pertanyaan.....	65
BAB 5	OTOMATISASI PEMBERSIHAN DATA	67
5.1	Pengenalan Pipeline Data Cleaning	67
5.2	Pemeriksaan Awal Data	67
5.3	Mengidentifikasi Nilai Yang Hilang	68
5.4	Deteksi Anomali Outlier	69
5.5	Pemeriksaan Kualitas Data	72
5.6	Pipeline Machine Learning	75
5.7	Ringkasan.....	78
5.8	Pertanyaan.....	79
BAB 6	KECERDASAN KAWANAN	80
6.1	Definisi Kecerdasan Kawanen	80
6.2	Karakteristik Dasar SI	80
6.3	Pedoman Desain Untuk Sistem SWARM	83
6.4	Optimasi Kawanen Partikel (PSO)	84
6.5	Ringkasan.....	90
6.6	Pertanyaan.....	91
BAB 7	PEMROGRAMAN GENETIK.....	92
7.1	Definisi Pemrograman Genetik.....	92
7.2	Konsep Dasar	92
7.3	Penerapan Algoritma Genetik	94
7.4	Struktur Algoritma Genetik	95
7.5	Contoh Algoritma Genetik.....	96
7.6	Ringkasan.....	99
BAB 8	DASAR PEMBELAJARAN MESIN DAN AI	101
8.1	Definisi Pembelajaran Mesin	101
8.2	Cara Kerja Jaringan Saraf	102
8.3	Jaringan Saraf Buatan	104
8.4	Cara Mesin Belajar Dari Kesalahan.....	105
8.5	Varian Arsitektur Ann	106
8.6	Pengaturan Parameter Model	108
8.7	Jaringan Saraf Dalam	109
8.8	Perceptron Multi-Lapis	110

8.9	Jaringan Saraf Konvolusional	110
8.10	Jaringan Berulang	112
8.11	Memori Jangka Panjang (LSTM)	112
8.12	Transformer	113
8.13	Analisis Komparatif Arsitektur	114
8.14	Ringkasan.....	115
8.15	Pertanyaan.....	116
BAB 9	PEMBELAJARAN PENGUATAN	117
9.1	Pembelajaran Penguatan.....	117
9.2	Istilah Dasar RL	118
9.3	Menentukan Strategi Terbaik	119
9.4	Kelebihan Dan Tantangan RL	121
9.5	Model Pengambilan Keputusan	122
9.6	Pengembalian Dan Faktor Diskon.....	123
9.7	Apa Kita Butuh Diskon?	123
9.8	Contoh: Belajar Berjalan.....	124
9.9	Fungsi Kebijakan Dan Nilai	126
9.10	Persamaan Bellman	128
9.11	Interpretasi Persamaan Bellman	129
9.12	Pendekatan Monte Carlo.....	130
9.13	Belajar Dari Perubahan Nilai	132
9.14	Dunia Grid Dan Prediksi Mc.....	133
9.15	Ringkasan.....	134
9.16	Pertanyaan.....	135
BAB 10	PEMBELAJARAN PENGUATAN MENDALAM	137
10.1	Algoritma Q-Learning	137
10.2	Contoh Lingkungan Grid	139
10.3	Gradien Kebijakan.....	143
10.4	Metode Actor–Critic	144
10.5	Eksplorasi Dan Eksploitasi.....	149
10.6	Deep Q-Network (DQN).....	151
10.7	Ringkasan.....	153
10.8	Pertanyaan.....	154
BAB 11	PEMODELAN BAHASA ALAMI.....	156
11.1	Pemrosesan Bahasa Alami (NLP)	156
11.2	Tahap-Tahap NLP	156
11.3	Ilmu Linguistik.....	157
11.4	Cara AI Belajar Bahasa	158
11.5	Teknik dan Tugas NLP.....	160
11.6	NLP Tradisional Vs AI Modern	161
11.7	Model Bahasa	162

11.8	Model N-Gram.....	163
11.9	Semantik Vektor Dan Embedding.....	165
11.10	Ringkasan.....	167
11.11	Pertanyaan.....	168
BAB 12	ARSITEKTUR TRANSFORMER DAN EVOLUSI LLM	170
12.1	Pendahuluan.....	170
12.2	WORD2VEC.....	171
12.3	Representasi Kata Vektor Global	175
12.4	Arsitektur Transformer	177
12.5	EVOLUSI LLM	180
12.6	Kesimpulan	181
12.7	Pertanyaan.....	182
BAB 13	Merancang Sistem AI Terdistribusi	184
13.1	Pola Desain Dan Bahasa Pola.....	184
13.2	Pipeline Big Data OCI	186
13.3	Kesimpulan	194
13.4	Pertanyaan.....	194
BAB 14	PENGAMANAN SISTEM AI	196
14.1	Ancaman Dan Perlindungan AI	196
14.2	Mendefinisikan Model Ancaman	198
14.3	Contoh Stride Pada Pipeline AI.....	199
14.4	Serangan Terhadap Sistem AI	200
14.5	Praktik Aman Untuk Sistem AI.....	201
14.6	Pertahanan AI Berlapis	203
14.7	Mengukur Kekuatan Pertahanan Sistem Ai	204
14.8	Mengukur Efektivitas Pertahanan	205
14.9	Ringkasan.....	206
14.10	Pertanyaan.....	207
BAB 15	KEAMANAN SISTEM AI DALAM PRAKTIK.....	208
15.1	Definisi Pipeline Pembersihan Data.....	208
15.2	Aspek Kunci Keamanan Sistem Ai.....	208
15.3	Prinsip-Prinsip Keselamatan Ai.....	209
15.4	Tantangan Ketidaksesuaian Ai	210
15.5	Peraturan Keamanan SISTEM AI	211
15.6	Panduan Keamanan Ai.....	212
15.7	Praktik Keamanan Ai.....	213
15.8	Merancang Ai untuk Situasi Kritis.....	214
15.9	Ringkasan.....	215
15.10	Pertanyaan.....	216
BAB 16	STRATEGI PENGUJIAN UNTUK APLIKASI AI.....	218
16.1	Pengujian Aplikasi Ai.....	218

16.2	Strategi Untuk Menguji Aplikasi AI	219
16.3	Pengujian Pengguna Akhir	221
16.4	Pengujian Non-Fungsional	222
16.5	Pengujian Fungsional	224
16.6	Verifikasi Dan Validasi Model	227
16.7	Pengujian Adversarial	228
16.8	Validasi Data	229
16.9	Praktik Pengujian Terbaik	230
16.10	Kesimpulan	231
16.11	Pertanyaan	232
KUNCI JAWABAN UNTUK PERTANYAAN BAB.....		233
Bab 1:	Jawaban	233
Bab 2:	Jawaban	235
Bab 3:	Jawaban	238
Bab 4:	Jawaban	241
Bab 5:	Jawaban	245
Bab 6:	Jawaban	247
Bab 7:	Jawaban	249
Bab 8:	Jawaban	256
Bab 9:	Jawaban	260
Bab 10:	Jawaban	263
Bab 11:	Jawaban	268
Bab 12:	Jawaban	271
Bab 13:	Jawaban	275
Bab 14:	Jawaban	276
Bab 15:	Jawaban	278
Bab 16:	Jawaban	280
DAFTAR PUSTAKA.....		283

BAB 1

KONSEP FUNDAMENTAL DALAM AI

1.1 PENDAHULUAN

Kecerdasan Buatan (AI)

Otak manusia adalah salah satu sistem paling kompleks dan canggih di alam semesta, terkenal karena kemampuannya untuk belajar, bernalar, dan beradaptasi. Seiring waktu, otak telah berevolusi untuk memenuhi kebutuhan bertahan hidup, memupuk rasa ingin tahu, dan menanggapi tantangan lingkungan. Evolusi ini telah memungkinkan manusia untuk memahami proses alam, mengidentifikasi hubungan sebab-akibat, dan mengembangkan alat dan mekanisme yang meningkatkan kehidupan kita. Sebagai contoh, pertimbangkan mata manusia. Terinspirasi oleh desainnya, para insinyur menciptakan kamera yang mampu menangkap dan menyimpan gambar dan video – contoh awal kecerdasan manusia yang memengaruhi teknologi. Kecerdasan manusia beroperasi pada paradigma merasakan → menyimpan → memproses → bertindak. Dengan menggunakan organ sensorik seperti mata, telinga, dan kulit, kita mengumpulkan informasi dari lingkungan sekitar kita. Informasi ini disimpan dalam memori, diproses untuk mengidentifikasi pola, dan kemudian digunakan untuk memandu tindakan berdasarkan konteks dan rangsangan.

Selama bertahun-tahun, kemajuan signifikan telah dicapai dalam mengembangkan mesin yang meniru aspek kecerdasan manusia. Kemajuan dalam komputasi telah memungkinkan penyimpanan volume data yang sangat besar di RAM dan perangkat penyimpanan sekunder. Algoritma berbasis jaringan saraf – seperti *Deep Neural Networks* (DNN), *Recurrent Neural Networks* (RNN), dan *Convolutional Neural Networks* (CNN) – meniru kemampuan otak untuk merasakan, menyimpan, dan memproses informasi, memungkinkan mesin untuk membuat keputusan dan melengkapi kemampuan manusia. Bab ini menyiapkan landasan untuk memahami konvergensi kecerdasan manusia dan mesin, meletakkan dasar untuk Machine Learning (ML) dan Artificial Intelligence (AI). Teknologi ini memungkinkan pemrosesan data dalam jumlah besar secara efisien. Sebaliknya, membangun sistem cerdas hanya menjadi mungkin dengan tersedianya Big Data.

1.2 BIG DATA

Revolusi digital telah memicu lonjakan yang belum pernah terjadi sebelumnya dalam generasi data, mengantarkan era Big Data. Organisasi saat ini mengumpulkan, menyimpan, dan memproses data dalam skala besar di berbagai aplikasi. Keputusan yang dulunya didasarkan pada intuisi atau model terbatas kini didorong oleh pendekatan berbasis data, sehingga memungkinkan akurasi dan efisiensi yang lebih besar. Big Data berdampak pada hampir setiap sektor :

- **Pendidikan:** Pelacakan kinerja mahasiswa secara terus menerus memberikan informasi untuk pengajaran yang dipersonalisasi.

- **Kesehatan:** Integrasi Big Data dan TI telah menghasilkan penghematan sekitar Rp.3 kuadriliun setiap tahunnya di AS.
- **Perencanaan Kota:** Data geografis beresolusi tinggi mendukung desain kota yang lebih cerdas.
- **Transportasi:** Data jaringan jalan secara real-time memungkinkan sistem lalu lintas yang cerdas.
- **Lingkungan:** Jaringan sensor menyediakan data terperinci untuk pemodelan iklim dan ekologi.
- **Energi:** Pola penggunaan membantu mengoptimalkan konsumsi dan mengurangi pemborosan.
- **Keuangan dan Keamanan:** Analisis kontrak dan transaksi mengungkap risiko sistemik dan meningkatkan deteksi penipuan.

Terlepas dari potensinya, Big Data menghadirkan tantangan yang signifikan (Gambar 1.1). Hal ini umumnya dicirikan oleh Lima V :

- **Volume:** Jumlah data yang sangat besar, seringkali dalam terabyte atau petabyte.
- **Variasi:** Beragam jenis data – terstruktur, semi-terstruktur, dan tidak terstruktur (teks, gambar, audio, video).
- **Kecepatan:** Aliran data berkecepatan tinggi yang membutuhkan pemrosesan waktu nyata atau mendekati waktu nyata.
- **Kebenaran:** Memastikan kualitas, akurasi, dan kepercayaan data.
- **Nilai:** Mengekstraksi wawasan yang dapat ditindaklanjuti yang memberikan manfaat nyata.

TANTANGAN BIG DATA					
Sumber	Kecepatan	Keragaman	Volume	Terstruktur	Tidak Terstruktur
Arsip					
Dokumen					
Aplikasi Bisnis					
Media					
Jejaring Sosial					
Web Publik					
Penyimpanan Data					
Data Log Mesin					
Data Sensor					

Rendah

Sedang

Tinggi

Ya (Dominan)

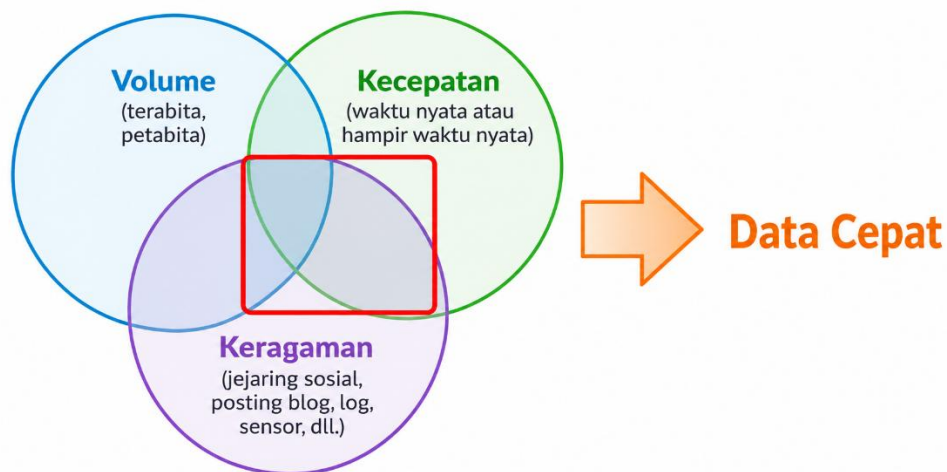
Tidak (Minor)

Gambar 1.1 Tantangan Big Data.

Organisasi yang secara efektif menangani dimensi-dimensi ini akan mendapatkan keunggulan kompetitif. Nantinya di bagian ini, kita akan membahas alur kerja Big Data dan fase-fasenya, beserta tantangan-tantangan yang terkait.

Gambar 1.2 mengilustrasikan konsep *Fast Data*, yang muncul di persimpangan tiga dimensi inti Big Data: *Volume*, *Kecepatan*, dan *Variasi*. Wilayah ini mewakili skenario di mana kumpulan data besar dan beragam dihasilkan dan diproses dengan kecepatan tinggi – seperti analitik waktu nyata di pasar keuangan atau aliran sensor IoT. Fast Data menekankan perlunya sistem yang dapat menangani penyerapan dan pemrosesan yang cepat tanpa mengorbankan akurasi atau skalabilitas. Alur kerja data dasar untuk menangani berbagai dimensi Big Data ditunjukkan pada Gambar 1.3. Pipeline ini mencakup beberapa fase: *fase akuisisi/perekaman data*, *fase ekstraksi/pembersihan/anotasi*, *fase integrasi/agregasi/representasi*, *fase analisis/pemodelan*, dan *fase interpretasi*.

Mengelola Big Data secara efektif membutuhkan pipeline terstruktur yang mengubah informasi mentah menjadi wawasan yang dapat ditindaklanjuti. Pipeline ini biasanya terdiri dari lima fase yang saling terkait, masing-masing menangani tantangan unik. Proses dimulai dengan *Akuisisi dan Perekaman Data*, di mana data yang relevan dikumpulkan dari berbagai sumber. Pada tahap ini, mendefinisikan metadata yang akurat dan mempertahankan asal usul data sangat penting untuk memastikan keterlacakan dan keandalan di seluruh langkah selanjutnya. Selanjutnya adalah fase *Ekstraksi, Pembersihan, dan Anotasi*. Data dunia nyata sering kali mengandung kesalahan, inkonsistensi, dan nilai yang hilang. Membersihkan dan menganotasi data sangat penting untuk meningkatkan kualitas dan mencegah hasil yang menyesatkan. Langkah ini menerapkan aturan validasi dan model kesalahan untuk mempersiapkan data untuk analisis.

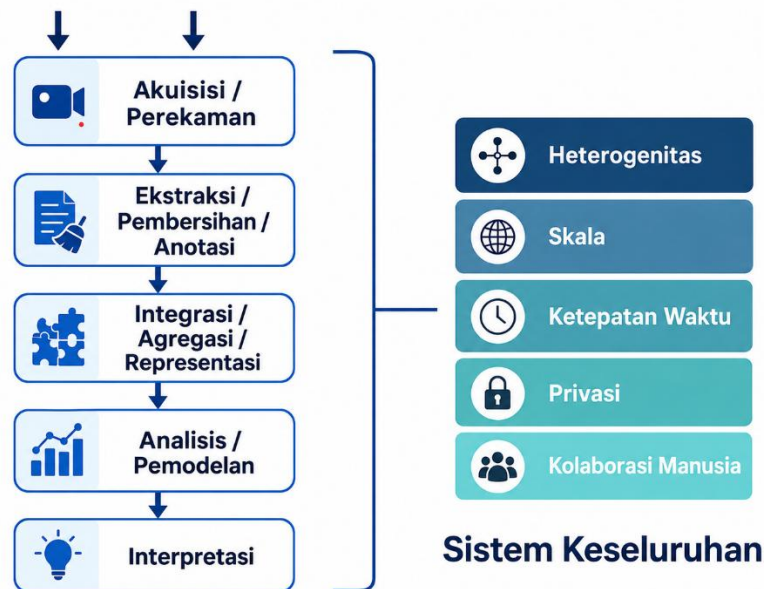


Gambar 1.2 Tampilan Konseptual Fast Data.

Fase ketiga, *Integrasi, Agregasi, dan Representasi*, berfokus pada penggabungan dataset heterogen ke dalam format yang terpadu. Data yang dikumpulkan dari sistem yang berbeda mungkin bervariasi dalam struktur, semantik, dan skema. Menyelesaikan ketidaksesuaian ini dan merepresentasikan data secara konsisten sangat penting untuk pemrosesan skala besar. Setelah data disiapkan, fase *Analisis dan Pemodelan* menerapkan teknik statistik, algoritma ML, dan alat visualisasi untuk mengungkap pola dan menghasilkan wawasan. Tantangan di sini

termasuk memilih model yang tepat, memastikan efisiensi komputasi, dan menghindari jebakan seperti overfitting.

Terakhir, fase *Interpretasi* menerjemahkan hasil analitis menjadi wawasan yang bermakna dan dapat ditindaklanjuti. Interpretasi yang efektif membutuhkan komunikasi yang jelas, relevansi kontekstual, dan teknik visualisasi yang membuat temuan dapat diakses oleh pemangku kepentingan. Salah tafsir pada tahap ini dapat menyebabkan keputusan yang buruk, yang menggarisbawahi pentingnya akurasi dan kejelasan.



Gambar 1.3 Alur Analisis Big Data.

Bersama-sama, fase-fase ini membentuk kerangka kerja komprehensif untuk menangani Big Data, memastikan bahwa informasi mentah diubah menjadi pengetahuan yang andal yang mendorong pengambilan keputusan yang terinformasi. Meskipun Big Data menawarkan peluang yang sangat besar, ia juga memperkenalkan tantangan kompleks yang mencakup dimensi teknis, hukum, dan sosial :

1. **Heterogenitas dan Ketidakeengkapan:** Big Data pada dasarnya beragam, mencakup format terstruktur, semi-terstruktur, dan tidak terstruktur. Kekayaan ini memberikan kedalaman tetapi menimbulkan kesulitan bagi algoritma, yang biasanya mengharapkan input yang seragam dan terstruktur dengan baik. Akibatnya, data harus distandarisi dengan cermat sebelum analisis untuk menghindari kesalahan dan ketidakefisienan.
2. **Skala dan Kecepatan:** Volume data yang sangat besar, ditambah dengan kecepatan pembuatannya, menciptakan tuntutan pemrosesan yang signifikan. Seiring bertambahnya ukuran dataset, analisis menjadi lebih intensif sumber daya, seringkali membutuhkan sistem terdistribusi dan teknik optimasi tingkat lanjut untuk mempertahankan wawasan yang tepat waktu.
3. **Privasi dan Keamanan:** Data sensitif – seperti rekam medis elektronik – diatur oleh peraturan yang ketat. Kekhawatiran publik tentang penyalahgunaan, terutama ketika

menghubungkan data dari berbagai sumber, memperkuat kebutuhan akan perlindungan privasi yang kuat. Mengatasi masalah ini membutuhkan solusi teknis (misalnya, enkripsi dan anonimisasi) dan pertimbangan sosiologis, seperti transparansi dan kepercayaan.

4. **Kolaborasi Manusia:** Terlepas dari kemajuan dalam otomatisasi, keahlian manusia tetap penting untuk tugas-tugas seperti pengenalan pola dan interpretasi kontekstual. Namun, memasukkan masukan manusia ke dalam alur kerja otomatis menimbulkan penundaan dan kompleksitas, sehingga otomatisasi penuh menjadi sulit.

Masalah-masalah ini menggarisbawahi pentingnya merancang sistem Big Data yang menyeimbangkan skalabilitas, keamanan, dan pengawasan manusia sambil memastikan kepatuhan terhadap standar etika dan hukum.

Tantangan dalam mengelola Big Data – mulai dari heterogenitas dan masalah privasi hingga kebutuhan akan kolaborasi manusia – menyoroti mengapa pendekatan tradisional tidak memadai. Untuk membuka potensi penuh dari kumpulan data yang sangat besar ini, kita membutuhkan sistem yang tidak hanya dapat memproses informasi secara efisien tetapi juga bernalar, belajar, dan beradaptasi. Di sinilah AI menjadi sangat diperlukan. AI menyediakan alat dan metodologi untuk mengubah data mentah menjadi tindakan cerdas, menjembatani kesenjangan antara kelimpahan data dan pengetahuan yang dapat ditindaklanjuti.

1.3 MENGAPA KITA MEMBUTUHKAN AI?

AI pada dasarnya adalah tentang memungkinkan mesin untuk berpikir dan bertindak secara cerdas. Sementara perangkat keras menyediakan fondasi fisik, perangkat lunak menentukan bagaimana mesin beroperasi. AI berfokus pada menjadikan sistem perangkat lunak ini mampu bernalar, belajar, dan mengambil keputusan – fungsi yang secara tradisional dikaitkan dengan kognisi manusia. Para ahli di bidang AI bertujuan untuk mengembangkan metode yang memungkinkan mesin untuk *merasakan, bernalar, belajar, dan bertindak secara rasional*. Dengan mempelajari bagaimana otak manusia memproses informasi, kita dapat merancang algoritma yang meniru kemampuan ini. Hasilnya adalah generasi baru sistem cerdas yang dapat melakukan tugas-tugas yang dulunya dianggap membutuhkan kecerdasan manusia. Pada bagian berikut, kita akan mengeksplorasi aplikasi praktis AI, termasuk *Generasi Bahasa Alami, Pengenalan Ucapan, Agen Virtual, dan AI Percakapan*, yang mengilustrasikan bagaimana teknologi ini mentransformasi industri dan kehidupan sehari-hari.

1.4 APLIKASI AI

Teknologi AI mentransformasi cara manusia berinteraksi dengan mesin dan cara bisnis memberikan layanan. Berikut adalah beberapa aplikasi utama:

- **Generasi Bahasa Alami (NLG):** NLG memungkinkan mesin untuk menghasilkan narasi tertulis atau lisan yang mirip manusia. Teknologi ini dibangun di atas linguistik komputasional dan Pemrosesan Bahasa Alami (NLP) untuk mengubah data terstruktur dan tidak terstruktur menjadi teks yang koheren. Misalnya, NLG mendukung

pembuatan laporan otomatis di bidang keuangan dan pembuatan konten yang dipersonalisasi di bidang pemasaran.

- **Pengenalan Ucapan:** Sistem pengenalan ucapan memungkinkan mesin untuk memahami dan memproses ucapan manusia. Teknik seperti Pengenalan Ucapan Otomatis (ASR) dan Dynamic Time Warping (DTW) mengubah kata-kata yang diucapkan menjadi teks, memungkinkan aplikasi seperti asisten suara, layanan transkripsi, dan kontrol perangkat tanpa tangan.
- **Agen Virtual:** Agen virtual adalah program berbasis AI yang berinteraksi dengan pengguna untuk melakukan tugas-tugas seperti penjadwalan, dukungan teknis, dan manajemen pesanan. Tidak seperti chatbot dasar, agen virtual memanfaatkan NLP untuk memahami maksud dan memberikan respons yang sesuai konteks, sehingga meningkatkan pengalaman pelanggan dan efisiensi operasional.
- **AI Percakapan:** AI percakapan menggabungkan NLP, pengenalan suara, dan ML untuk mensimulasikan percakapan seperti manusia. AI ini mendukung chatbot, aplikasi perpesanan, dan asisten berbasis suara yang memberikan dukungan personal dan real-time dalam skala besar. Bisnis menggunakan AI percakapan untuk mengurangi waktu penyelesaian pertanyaan dan meningkatkan keterlibatan pelanggan.
- **Antarmuka Pemrograman Aplikasi (API):** API menyederhanakan integrasi AI dengan menyediakan layanan siap pakai untuk visi, bahasa, ucapan, dan pengambilan keputusan. Contohnya meliputi:
 - **API Visi:** Pemrosesan gambar dan deteksi objek.
 - **API Bahasa:** Analisis sentimen dan ekstraksi frasa kunci.
 - **API Ucapan:** Konversi ucapan ke teks dan fitur yang diaktifkan suara.
 - **API Keputusan:** Deteksi anomali dalam data deret waktu.

Aplikasi-aplikasi ini menggambarkan keserbagunaan AI dalam mengotomatisasi tugas, meningkatkan pengambilan keputusan, dan menciptakan sistem cerdas yang meniru kemampuan manusia.

1.5 BAHASA PEMROGRAMAN UNTUK AI

Mengembangkan aplikasi berbasis AI membutuhkan bahasa pemrograman yang menyeimbangkan kinerja, fleksibilitas, dan kemudahan penggunaan. Meskipun beberapa bahasa tersedia – seperti Python, Java, C++, Julia, R, MATLAB®, Prolog, Lisp, dan Haskell – yang paling banyak diadopsi untuk pengembangan AI meliputi Python, Java, C++, dan Julia.

- **Python:** Bahasa dominan untuk pengembangan AI, Python dihargai karena keterbacaannya, sintaksis yang ramah bagi pemula, dan ekosistem pustaka yang luas (misalnya, TensorFlow, PyTorch, dan scikit-learn). Fleksibilitasnya menjadikannya ideal untuk ML, pembelajaran mendalam, dan analisis data.
- **Java:** Dikenal karena skalabilitas dan ketangguhannya, Java sering digunakan dalam sistem AI tingkat perusahaan, termasuk jaringan saraf, algoritma genetika, dan sistem multi-agen.

- **C++:** Lebih disukai untuk aplikasi yang kritis terhadap kinerja, C++ menawarkan kontrol yang lebih detail atas memori dan komputasi, sehingga cocok untuk sistem AI waktu nyata dan lingkungan dengan keterbatasan sumber daya.
- **Julia:** Dirancang untuk komputasi ilmiah berkinerja tinggi, Julia menggabungkan kecepatan dengan kemudahan penggunaan, menjadikannya pilihan yang tepat untuk analisis numerik dan aplikasi AI yang membutuhkan eksekusi cepat.

Bahasa pemrograman lain memiliki peran khusus:

- **MATLAB** unggul dalam pemodelan dan visualisasi matematika, sering digunakan untuk pembuatan prototipe.
- **R** populer untuk analisis statistik dan visualisasi data.
- **Prolog** mendukung AI berbasis logika dan sistem pakar.
- **Lisp dan Haskell** menawarkan fleksibilitas untuk pemrograman fungsional dan pengembangan AI tingkat lanjut.

Di antara semua bahasa ini, *Python* tetap menjadi pilihan utama bagi para insinyur dan profesional yang bekerja dengan AI dan Big Data. Di bagian selanjutnya, kami akan memberikan pengantar praktis tentang pemrograman Python, termasuk contoh dan paket-paket penting.

1.6 DASAR PEMROGRAMAN PYTHON

Memulai dengan Python dan Jupyter Notebook: Sebelum kita mulai menulis skrip Python dasar dan mulai bereksperimen dengan berbagai paket di Python, Anda harus mengunduh versi terbaru Python dan Jupyter Notebook, lingkungan pengembangan interaktif untuk Python. Jupyter Notebook memudahkan penulisan dan menjalankan kode secara bertahap, memvisualisasikan output, dan menambahkan dokumentasi. Sekarang kita akan memperkenalkan beberapa contoh. Pembaca disarankan untuk menyalin, menempel, dan menjalankan cuplikan kode ini di Jupyter Notebook.

- **Skrip Python Pertama Anda:** Mari kita mulai dengan menampilkan teks sederhana – keterampilan dasar namun penting:

```
print("Hello, World!")
```

Ini akan menghasilkan keluaran sebagai berikut:

```
Hello, World!
```

- **Variabel dan Aritmatika Dasar:** Sekarang, mari kita deklarasikan variabel di Python untuk menyimpan nilai bilangan bulat dan melakukan operasi aritmatika dasar. Variabel menyimpan nilai. Python menangani berbagai tipe data, seperti bilangan bulat dan string, tanpa deklarasi eksplisit.

```
v = 2 # declare a variable in Python
```

```
print(v)
# Perform some arithmetic operations on the Python variable
v = v * 5
print(" The value of the variable v is:", v)
```

Output:

```
2
The value of the variable v is: 10
```

- **Bekerja dengan String:** Python membuat penanganan string menjadi intuitif dan ampuh, memungkinkan penggabungan dan pemformatan yang mudah. Misalnya,

```
word1 = "Good" # Assign the word Good to the Python variable
word1
word2 = "morning"
word3 = "to you too!"
```

Kita dapat menggabungkan variabel string dan menampilkannya sebagai kalimat menggunakan perintah tersebut.

```
print(word1, word2)
Output:
Goodmorning
```

Saat kata 1 dan kata 2 dicetak di layar, harus ada spasi kosong di antara keduanya. Outputnya seharusnya "Selamat pagi." Untuk melakukan itu, kita harus memasukkan spasi kosong seperti yang ditunjukkan di bawah ini.

```
sentence = word1 + " " + word2 + " " + word3
```

Di sini, variabel string digabungkan menggunakan tanda +. Sekarang kita dapat melakukan hal berikut dengan hasil yang diharapkan:

```
print(sentence)
```

Output:

```
Good morning to you too!
```

- **Iterasi:** Iterasi sangat penting untuk memproses kumpulan data di Python. Salah satu cara umum untuk mengulang item adalah dengan menggunakan perulangan WHILE. Selama kondisi tertentu terpenuhi, perulangan ini akan mengulangi serangkaian instruksi. Agar dua baris di bawah perulangan WHILE dapat berjalan di dalam

perulangan WHILE, baris tersebut harus diketik dengan jarak tab (indentasi). Mari kita pertimbangkan sebuah contoh untuk mengilustrasikan hal ini:

```
a = 0 # initialize the value of the loop variable
while a < 5: # loop till a reaches the value of 10
    a = a + 1 # Increment a to make sure that the loop
terminates
    print(a) # print the value of the variable a in each
iteration.
```

Output:

```
1
2
3
4
5
```

Setelah menjalankan skrip di atas, Anda akan melihat nilai variabel a dicetak sepuluh kali, yaitu dari 1 hingga 10.

Mempertimbangkan contoh lain untuk loop WHILE di Python. Di sini, perhatikan bahwa perintah print terakhir berada di luar loop while dan dieksekusi ketika loop berakhir.

```
# This block prints integers between 10 and 20 counting by 5
# Prints 'End of while loop' when finished
v = 5
while v < 20:
    v = v + 5 # this line is part of while loop
    print(v) # this line is part of while loop
print("End of while loop")
```

Output:

```
10
15
20
End of while loop
```

Sekarang mari kita lihat contoh perulangan FOR dalam Python.

```
# This block prints value of variables 'cnt' and 'value'
for cnt, value in enumerate([10,20,30]):
    print (cnt,value)
```

Output:

```
0 10
1 20
2 30
```

- **Pengambilan Keputusan (Kondisional):** Python menggunakan `if`, `elif`, dan `else` untuk memungkinkan logika pengambilan keputusan berdasarkan kondisi. Mari kita pertimbangkan contoh perulangan FOR yang dikombinasikan dengan pernyataan if-else. Perhatikan indentasi yang digunakan di sini untuk mewakili berbagai blok perulangan.

```
# This block prints which integer between 2 and 10 is a prime
or has a factor
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print( n, 'equals', x, '*', n/x)
            break
        else:
            # no factors were found hence control is transferred
to else block
            print(n, 'is a prime number')
```

Output:

```
2 is a prime number
3 is a prime number
4 equals 2 * 2.0
5 is a prime number
6 equals 2 * 3.0
7 is a prime number
8 equals 2 * 4.0
9 equals 3 * 3.0
```

Sekarang mari kita jelajahi penggunaan pernyataan break. Sekali lagi, perhatikan bagaimana indentasi digunakan di sini untuk mewakili blok yang berbeda.

```
# This block prints integers from zero to 5
count = 0
while True:
    count += 1
    if count > 5:
        break

    print (count)
```

Output:

```
1
```

2
3
4
5

Mari kita pertimbangkan sebuah contoh untuk mengilustrasikan penggunaan pernyataan `continue` dalam Python. Blok kode ini hanya akan mencetak angka ganjil antara 0 dan 10.

```
# This block prints odd integers between 1 and 10
count = 0
while count < 10:
    count += 1
    if count % 2 == 0:
        continue
    print (count)
```

Output:

1
3
5
7

Sekarang, mari kita tulis fungsi sederhana dalam Python untuk menerima input dari pengguna. Sebagai respons terhadap input pengguna, outputnya akan bervariasi. Jika pengguna memasukkan 1, maka output yang diperoleh adalah Teh, dan jika pengguna memasukkan 2, maka output yang diperoleh adalah Kopi.

```
# This block takes user input; if use enters 1, it prints tea
otherwise prints coffee
choice = int (input ("Enter 1 for Tea and 2 for Coffee:"))
if choice == 1:
    print("tea")
else:
    print("coffee")
```

Output:

Enter 1 for Tea and 2 for Coffee:
2
Coffee

- **Penanganan Pengecualian:** Python menyediakan alat yang andal untuk penanganan kesalahan menggunakan blok `try`, `except`, `else`, dan `finally`. Di sini, klausa `try` digunakan untuk menimbulkan pengecualian yang didefinisikan pengguna, dan blok `except` menangkapnya dan menanganinya sesuai dengan definisi pengguna. Perhatikan baik-baik indentasi di dalam blok `try` dan `except`.

```
# This block takes a user input; raises an exception if the
user does not enter a
# number (ValueError) or enters a 0 (ZeroDivisionError)
try:
    value = int(input("Enter a number: "))
    result = 10 / value
except ValueError:
    print("Invalid input. Please enter a valid number.")
except ZeroDivisionError:
    print("Cannot divide by zero.")
else:
    print(f"Division successful: {result}")
finally:
    print("This always runs (e.g., for cleanup)")
```

Output:

```
Enter a number:
0
Cannot divide by 0.
This always runs
```

- **Fungsi:** Fungsi memungkinkan penggunaan kembali kode, enkapsulasi, dan modularitas. Mari kita ambil contoh untuk menunjukkan bagaimana fungsi dapat didefinisikan dan dipanggil (dirujuk) dalam Python. Pertama, mari kita definisikan fungsi *hello()* yang sudah familiar.

```
def hello(): # define the function using def
    print("Hello, World!") # statement inside the function
```

Fungsi *hello()* ini dapat dipanggil sebagai

```
hello() # call the function
```

Output:

```
Hello, World!
```

Sekarang mari kita pertimbangkan sebuah contoh untuk mengilustrasikan cara meneruskan parameter ke fungsi yang mencetak ucapan salam.

```
# This function prints a greeting to the user.
def greet(user):
    print(f"Hello, {user}!")

greet("Alice")
```

Output:
Hello, Alice!

- **Kelas:** Pemrograman berorientasi objek di Python menggunakan kelas untuk memodelkan sistem yang kompleks dan membungkus data serta perilaku. Perhatikan contoh yang menunjukkan pembuatan kelas dengan konstruktor dan inisialisasi objek.

```
# Create a class called Dog
class Dog:
    def __init__(self, name):
        self.name = name

    # Define a class method called bark
    def bark(self):
        print(f"{self.name} says woof!")

# Create an instance Fido of the Dog class and have it bark
fido = Dog("Fido")
fido.bark()
```

Output:
Fido says woof!

- **Input/Output File:** Selanjutnya, mari kita jelajahi cara membaca dan menulis ke file di Python. Contoh berikut mengubah semua huruf vokal dalam file input menjadi huruf besar dan menulis baris yang telah diubah ke file output.

```
import string
import sys

# This function converts all the vowels in an input file to an
upper case and

# writes the converted lines to an output file
def translate(infilename, outfilename):
    tran_table = str.maketrans('aeiou', 'AEIOU')
    with open(infilename, 'r') as infile, open(outfilename, 'w') as
outfile:
        for line in infile:
            line = line.rstrip()
            outfile.write('%s\n' % line.translate(tran_table))

# Name the input and output files
input_file = "/Users/Sid/Documents/lowercase.txt"
output_file = "/Users/Sid/Documents/uppercase.txt"
```

```
# Print contents of the input file
print("Input File:")
try:
    with open(input_file, 'r') as f:
        print(f.read(), end="") # Use end="" to avoid an extra newline
except FileNotFoundError:
    print(f"Error: File '{file_name}' not found.")
except Exception as e:
    print(f"An error occurred: {e}")

translate(input_file, output_file)

#Print contents of the output file
print("\Output File:")
try:
    with open(output_file, 'r') as f:
        print(f.read(), end="") # Use end="" to avoid an extra
        newline
except FileNotFoundError:
    print(f"Error: File '{file_name}' not found.")
except Exception as e:
    print(f"An error occurred: {e}"):
```

Output:

```
Input File:
hi diddle diddle
the cat and the fiddle
the cow jumped over the moon"
```

```
Output File:
hI dIddlE dIddlE
thE cAt And thE fIddlE
thE cOw jUmPEd OvEr thE mOOn"
```

- **Bekerja dengan DataFrame (Pandas):** Pandas adalah pustaka Python. DataFrame Pandas adalah struktur inti untuk menganalisis data tabular.

```
# Import Pandas library
import pandas as pd

# Create a DataFrame from dictionary data
data = {'name': ['Alice', 'Bob'], 'age': [28, 34]}
df = pd.DataFrame(data)

# Basic operations
print(df.head()) # Show first rows
```

```
print(df.describe()) # Summary statistics

# Filtering
adults = df[df['age'] >= 30]
print(adults)
```

Output:

```
   name  age
0 Alice  28
1 Bob   34

      age
count  2.000000
mean   31.000000
std     4.242641
min     28.000000
25%     29.500000
50%     31.000000
75%     32.500000
max     34.000000

   name  age
1 Bob   34
```

DataFrame memungkinkan manipulasi data yang ampuh, menjadikannya penting untuk pekerjaan analitik profesional.

- **Menginstal Paket Tambahan:** Sekarang kita memiliki pengalaman yang baik dalam membuat skrip di Python, mari kita jelajahi beberapa paket Python yang akan sangat berguna ketika kita menjelajahi topik *Pembelajaran Penguatan* di Bab 7 dan 8. Paket memperluas kemampuan Python. Pastikan Anda memiliki *pip* di mesin Anda, yang akan dibutuhkan untuk menginstal paket *TensorFlow* dan *OpenAI Gym*. Jika Anda sudah menginstal *pip* di sistem Anda, gunakan untuk menginstal *TensorFlow* di sistem menggunakan perintah ini

```
pip install tensorflow
```

Untuk instalasi *TensorFlow* dan dependensinya, silakan merujuk ke tautan berikut: <https://www.tensorflow.org/install>. Sekarang, gunakan aplikasi *pip* untuk menginstal *gym* di sistem menggunakan perintah ini

```
pip install gym[all]
```

Kami sekarang memberikan pengantar singkat tentang penggunaan paket *TensorFlow* dan *OpenAI Gym*. Pada bab-bab selanjutnya, kami akan menggunakan kedua paket ini secara ekstensif untuk mengimplementasikan dan bereksperimen dengan algoritma Reinforcement Learning.

- **Komputasi menggunakan Paket TensorFlow:** Dasar dari paket *TensorFlow* adalah grafik komputasi dan tensor. Tensor didefinisikan sebagai vektor n-dimensi. Variabel skalar dan matriks juga dapat disebut sebagai tensor. Berikut adalah beberapa komputasi dasar menggunakan *TensorFlow* untuk tujuan demonstrasi.

Impor *TensorFlow* dengan perintah berikut:

```
import tensorflow.compat.v1 as tf
```

Dengan menggunakan fungsi *tf.zeros()* dan *tf.ones()*, kita dapat membuat tensor dasar. Fungsi *tf.zeros()* menerima bentuk tensor (misalnya, tuple) dan mengembalikan tensor dengan bentuk tersebut dengan semua nilai diinisialisasi menjadi nol. Demikian pula, *tf.ones()* menerima bentuk tensor dan mengembalikan tensor dengan bentuk tersebut yang berisi semua nilai satu. Ketik perintah berikut untuk membuat tensor:

```
tf.zeros(3)
tf.ones(3)
```

Pada bab-bab selanjutnya, kita akan menjelajahi grafik dan tensor komputasional. Sekarang, hanya untuk menguji semua instalasi kita, jalankan skrip berikut:

```
with tf.compat.v1.Session() as sess:
    hello = tf.constant('hello world')
    print(sess.run(hello))
```

Saat dijalankan, Anda akan melihat "*hello world*" tercetak sebagai output. Sekarang, mari kita pertimbangkan untuk membangun sebuah grafik, meluncurkan grafik tersebut dalam sebuah sesi, dan mengevaluasi tensornya. Jalankan skrip di bawah ini untuk mengevaluasi tensor *c*.

```
tf.compat.v1.disable_eager_execution() # need to disable eager
in TF2.x
#Build a graph.
a = tf.constant(5.0)
b = tf.constant(6.0)
c = a * b
# Launch the graph in a session.
sess = tf.compat.v1.Session()
# Evaluate the tensor `c`.
```

```
print(sess.run(c)) # prints 30.0
```

Saat dieksekusi, Anda seharusnya dapat melihat nilai tensor *c*, yaitu 30.

- **Komputasi Menggunakan Paket OpenAI Gym:** *OpenAI Gym* kompatibel dengan pustaka pembelajaran mendalam seperti *TensorFlow* dan *Theano*. Di sini, kami memberikan pengantar singkat tentang *OpenAI Gym*. Ada dua bagian :
 1. **Pustaka Open-Source Gym:** Pustaka ini terdiri dari banyak lingkungan untuk berbagai masalah pengujian di mana algoritma Reinforcement Learning dapat diuji. Ini cukup dengan informasi tentang ruang keadaan dan aksi.
 2. **Layanan OpenAI Gym:** Bagian ini memungkinkan pengguna untuk membandingkan kinerja agen mereka dengan agen terlatih lainnya.

Pada bab ini, kita akan mempertimbangkan aplikasi sederhana dari Reinforcement Learning menggunakan paket *OpenAI Gym*. Dalam contoh ini, kita akan memuat lingkungan dan fungsi penting lainnya yang terkait dengan *OpenAI Gym* menggunakan contoh *lingkungan Frozen Lake*. *Lingkungan Frozen Lake* akan dibahas lebih detail lagi di Bab 7. Sebelum melanjutkan dengan contoh ini, pastikan *OpenAI Gym* telah terinstal di komputer Anda. Untuk memuat *lingkungan Frozen Lake*, ketik perintah berikut

```
import gym
env = gym.make('FrozenLake-v1') #make function of Gym loads
the specified
env
s = env.reset() # resets the environment and returns the start
state as a
print(s)
env.render()
env = env.unwrapped
print(env.action_space)
print(env.action_space.n)
print(env.observation_space)
print(env.observation_space.n)
```

Pada dua baris pertama, kita mengimpor paket *OpenAI Gym* dan memuat lingkungan untuk *FrozenLake versi 1*. Sisa skrip berfokus pada pengaturan ulang lingkungan, memungkinkan agen untuk melalui beberapa episode agar dapat belajar, melakukan tindakan, dan lain-lain. Kita akan membahas semua konsep ini di Bab 7. Setelah menjalankan skrip, Anda seharusnya dapat melihat agen di *FrozenLake versi 1*. *FrozenLake* adalah grid dengan 16 keadaan yang disusun dengan cara SFFF, FHFH, FFFH, HFFG, di mana **S**: adalah blok Mulai, **F**: adalah blok Beku, **H**: adalah Blok yang memiliki lubang, dan **G**: adalah blok Tujuan. Dalam bab ini, tujuannya hanyalah untuk memberikan paparan pemrograman dalam Python sehingga pembaca dapat terbiasa dengan sintaks yang digunakan dalam skrip Python. Di bab-bab selanjutnya, skrip Python tambahan akan diberikan untuk mengilustrasikan berbagai konsep.

1.7 RINGKASAN

Bab ini memperkenalkan konsep-konsep dasar AI dan Big Data, menekankan konvergensi keduanya di era revolusi data. Kita telah mengeksplorasi paradigma Big Data, termasuk karakteristik utamanya – Lima V – dan meneliti alur data dengan fase-fasenya dan tantangan-tantangan terkait. Diskusi ini menyoroti mengapa pendekatan tradisional tidak memadai dan menyiapkan panggung bagi AI sebagai penggerak penting pengambilan keputusan yang cerdas. Kemudian kita membahas kebutuhan akan AI, mengilustrasikan perannya dalam membangun sistem yang dapat merasakan, bernalar, dan bertindak secara cerdas. Aplikasi praktis seperti NLG, Pengenalan Ucapan, Agen Virtual, dan AI Percakapan menunjukkan bagaimana teknologi ini mentransformasi industri dan kehidupan sehari-hari.

Terakhir, bab ini memberikan pengantar tentang pemrograman Python, memperkenalkan konstruksi dan paket penting yang banyak digunakan dalam AI dan analitik data. Perhatian khusus diberikan pada pustaka seperti TensorFlow dan OpenAI Gym, yang akan menjadi dasar untuk topik-topik lanjutan di bab-bab selanjutnya. Dengan menetapkan prinsip dan alat inti ini, bab ini mempersiapkan pembaca untuk eksplorasi lebih dalam tentang teknik AI, kerangka kerja Big Data, dan integrasinya dalam sistem dunia nyata.

1.8 PERTANYAAN

1. Analisis pengaruh paradigma manusia “merasakan → menyimpan → memproses → bertindak” pada kecerdasan buatan. Dalam jawaban Anda: (a) Jelaskan bagaimana siklus ini berkaitan dengan berbagai arsitektur jaringan saraf, termasuk Jaringan Saraf Konvolusional (CNN), Jaringan Saraf Berulang (RNN), dan Jaringan Saraf Dalam (DNN). (b) Sajikan argumen yang meyakinkan yang mendukung pernyataan bahwa Big Data adalah prasyarat yang diperlukan dan realistis untuk sistem AI kontemporer.
2. Definisikan Lima V Big Data dan berikan satu contoh lintas sektor untuk setiap karakteristik.
3. Jelaskan setiap fase dari alur analisis Big Data, seperti yang diilustrasikan pada Gambar 1.3. Untuk setiap fase, identifikasi satu tantangan teknis dan satu praktik terbaik (baik yang terkait metadata atau berorientasi proses) yang mengatasi tantangan ini.
4. Teknologi manakah di antara berikut ini yang memungkinkan komputer untuk mensimulasikan percakapan seperti manusia?
 - A. AI Percakapan
 - B. Pemrosesan Bahasa Alami
 - C. API
 - D. Agen Virtual
 - E. Pengenalan Ucapan
5. Ketika ditugaskan untuk mendeteksi anomali dalam deret waktu univariat dari pembacaan sensor, kategori API mana yang paling tepat untuk digunakan pada awalnya?
 - A. API Visi
 - B. API Pencarian

- C. API Bahasa
 - D. API Ucapan
 - E. API Keputusan
6. Lakukan analisis komparatif Python, Java, C++, Julia, MATLAB, R, Prolog/Lisp, dan Haskell untuk aplikasi kecerdasan buatan. Identifikasi skenario di mana Python adalah pilihan yang lebih disukai dan keadaan di mana bahasa alternatif mungkin lebih cocok.
 7. (Tugas Pemrograman)
 - a. Implementasikan fungsi bernama *concat_two*(a, b) yang menggabungkan dua string input 'a' dan 'b', memisahkannya dengan satu spasi. Demonstrasikan fungsi tersebut dengan membaca dua kata sebagai input dan menampilkan hasilnya.
 - b. Dengan menggunakan perulangan for dengan klausa else, tulis program yang mencetak semua bilangan prima dari 2 hingga 30, tanpa menggunakan pustaka eksternal.

BAB 2

SISTEM BIG DATA DAN AI

2.1 SISTEM BIG DATA

Bab 1 telah menetapkan konteks konvergensi kecerdasan manusia dan mesin, memperkenalkan *Big Data*, *Pembelajaran Mesin* (ML), dan *Kecerdasan Buatan* (AI). Dalam bab ini, kita akan membangun fondasi tersebut dengan memeriksa arsitektur kerangka kerja Big Data dan sistem AI, mengidentifikasi komponen intinya, dan mengeksplorasi bagaimana teknologi ini memungkinkan pengambilan keputusan yang cerdas dalam skala besar. Organisasi modern menghadapi tantangan yang belum pernah terjadi sebelumnya dalam menyimpan, memproses, dan menganalisis data. Komputasi awal bergantung pada sistem file sekuensial dan basis data relasional, yang efektif untuk data terstruktur tetapi kesulitan dengan persyaratan *volume*, *kecepatan*, dan *variasi* saat ini (lihat Bab 1). *Sistem Manajemen Basis Data Relasional* (RDBMS) tradisional tidak dapat dengan mudah diskalakan ke kumpulan data tingkat petabyte atau memberikan kinerja latensi rendah tanpa biaya dan kompleksitas yang signifikan.

Untuk mengatasi keterbatasan ini, teknologi Big Data muncul, memanfaatkan arsitektur terdistribusi dan klaster perangkat keras komoditas. Sistem-sistem ini menyediakan skalabilitas tinggi, toleransi kesalahan, dan efisiensi biaya, memungkinkan pemrosesan paralel di ratusan atau ribuan node. Contohnya meliputi:

- **Hadoop:** Kerangka kerja terdistribusi untuk penyimpanan dan komputasi menggunakan model pemrograman MapReduce.
- **Basis Data NoSQL:** Seperti HBase dan Cassandra, dirancang untuk data multi-struktur dan akses waktu nyata.
- **Basis Data Pemrosesan Paralel Massif (MPP):** Platform seperti Amazon Redshift dan Teradata, yang membagi data di seluruh node untuk komputasi dalam memori berkecepatan tinggi.

Pada bagian selanjutnya, kita akan mengeksplorasi arsitektur berlapis ekosistem Big Data, dengan fokus pada komponen seperti *HDFS*, *MapReduce*, dan *YARN*, serta alat-alat canggih seperti *Spark*, *Hive*, dan *Pig*. Teknologi-teknologi ini membentuk tulang punggung platform analitik modern, memungkinkan organisasi untuk mengubah data mentah menjadi wawasan yang dapat ditindaklanjuti.

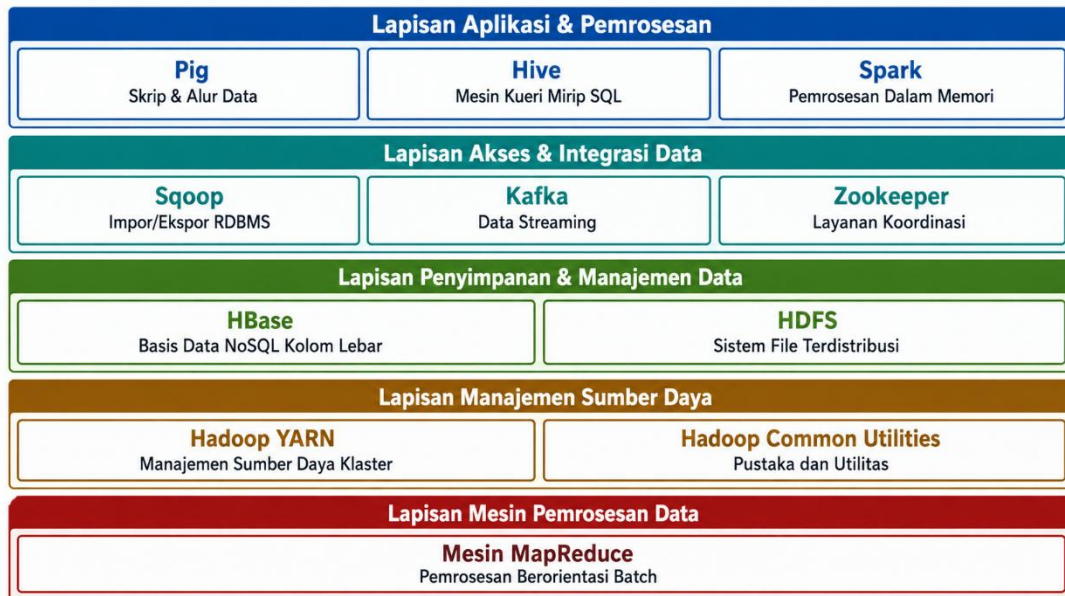
2.2 KOMPONEN UTAMA BIG DATA

Kerangka kerja Big Data modern dibangun di atas arsitektur berlapis yang memungkinkan penyimpanan, pemrosesan, dan analitik yang terukur. Kerangka kerja ini biasanya terdiri dari dua elemen inti: *Kerangka Kerja Pemrosesan* dan *Mesin Pemrosesan*, yang bekerja sama untuk memasukkan, membaca, dan menghitung dataset besar. Misalnya, Hadoop berfungsi sebagai kerangka kerja pemrosesan, sementara MapReduce bertindak

sebagai mesin pemrosesannya. Demikian pula, Apache Spark menyediakan kerangka kerja dengan kemampuan pemrosesan dalam memori untuk beban kerja batch dan real-time.

Konsep kunci dalam sistem Big Data adalah pemrosesan batch, yang melibatkan pemrosesan kumpulan data statis berukuran besar yang disimpan dalam penyimpanan persisten. Pemrosesan batch ideal untuk analisis komprehensif tetapi kurang cocok untuk persyaratan latensi rendah karena waktu eksekusinya yang lebih lama. Untuk mengilustrasikan komponen-komponen ini, pertimbangkan ekosistem Hadoop, yang diorganisasikan menjadi beberapa lapisan (lihat Gambar 2.1). Setiap lapisan menangani fungsi spesifik:

- **Lapisan Penyimpanan Data:**
 - **HDFS (Hadoop Distributed File System):** Sistem file terdistribusi yang toleran terhadap kesalahan, dirancang untuk akses throughput tinggi ke kumpulan data besar. Sistem ini berjalan pada perangkat keras berbiaya rendah dan mendukung aplikasi yang membutuhkan penyimpanan besar.
 - **HBase:** Basis data NoSQL berorientasi kolom yang beroperasi di atas HDFS, mendukung operasi baca/tulis waktu nyata.
- **Lapisan Pemrosesan Data:**
 - **MapReduce:** Model pemrograman paralel yang memproses kumpulan data besar di seluruh kluster secara efisien dan andal.
- **Lapisan Manajemen Sumber Daya:**
 - **YARN (Yet Another Resource Negotiator):** Menangani penjadwalan pekerjaan dan alokasi sumber daya di seluruh kluster.
 - **Hadoop Common:** Menyediakan pustaka dan utilitas Java penting untuk ekosistem.
- **Lapisan Aplikasi dan Pemrosesan:**
 - **Pig:** Menawarkan bahasa skrip tingkat tinggi (Pig Latin) dan mesin eksekusi untuk menyederhanakan pemrograman MapReduce.
 - **Hive:** Sistem gudang data terdistribusi dengan bahasa kueri sendiri (HQL), mirip dengan SQL.
 - **Spark:** Menyediakan pemrosesan dalam memori untuk eksekusi yang lebih cepat, mendukung pemrosesan batch dan stream. Spark terintegrasi dengan sumber seperti Kafka dan mencakup API untuk SQL, streaming, dan ML.
- **Lapisan Akses dan Integrasi Data:**
 - **Sqoop:** Memfasilitasi transfer data antara basis data relasional dan HDFS.
 - **Kafka:** Sistem perpesanan terdistribusi untuk streaming data real-time.
 - **Zookeeper:** Menyediakan layanan koordinasi dan sinkronisasi di seluruh node kluster.



Gambar 2.1 Tampilan berlapis ekosistem Hadoop.

Bersama-sama, komponen-komponen ini membentuk ekosistem yang kuat untuk menangani kumpulan data besar, memungkinkan organisasi untuk melakukan analitik kompleks dan memperoleh wawasan yang dapat ditindaklanjuti.

2.3 GAMBARAN UMUM SISTEM AI

Sistem AI adalah solusi berbasis perangkat lunak yang dirancang untuk melakukan tugas-tugas yang biasanya membutuhkan kecerdasan seperti manusia, seperti pembelajaran, penalaran, dan pemecahan masalah. Sistem-sistem ini menganalisis data, mengidentifikasi pola, dan membuat keputusan – seringkali secara real-time – menggunakan algoritma canggih. Sistem AI berkisar dari program berbasis aturan sederhana hingga teknologi canggih seperti *ML* dan *Deep Learning (DL)*. Karakteristik utama sistem AI meliputi:

- **Emulasi Kognitif:** Kemampuan untuk meniru kemampuan manusia seperti belajar, penalaran, dan pengambilan keputusan.
- **Wawasan Berbasis Data:** Kapasitas untuk memproses volume data yang besar, mengungkap pola yang kompleks, dan membuat keputusan yang tepat.
- **Adaptabilitas:** Potensi untuk belajar dari informasi baru dan menyesuaikan diri dengan lingkungan yang berubah, meningkatkan kinerja dari waktu ke waktu.

Sistem AI memainkan peran penting dalam aplikasi seperti *visi komputer*, *pemrosesan bahasa alami (NLP)*, *pengenalan gambar*, dan *robotika*. Konsep dasar meliputi:

- **Pembelajaran Mesin (ML):** Sub-bidang AI di mana sistem mempelajari pola dari data tanpa pemrograman eksplisit. Paradigma pembelajaran meliputi pendekatan terawasi, tidak terawasi, dan semi-terawasi.
- **Deep Learning (DL):** Bentuk khusus dari ML yang menggunakan jaringan saraf tiruan berlapis-lapis untuk memodelkan hubungan kompleks dalam data.

- **Natural Language Processing (NLP):** Memungkinkan mesin untuk memahami, menafsirkan, dan menghasilkan bahasa manusia, yang mendukung aplikasi seperti chatbot dan analisis sentimen.
- **Computer Vision:** Memungkinkan mesin untuk menafsirkan dan menganalisis data visual seperti gambar dan video.
- **Sistem Pakar:** Program AI yang menerapkan aturan yang telah ditentukan sebelumnya dan pengetahuan khusus domain untuk membuat keputusan, mendiagnosis masalah, dan memberikan rekomendasi.

Kemampuan-kemampuan ini menempatkan sistem AI sebagai alat penting untuk mengotomatisasi proses, meningkatkan pengambilan keputusan, dan mendorong inovasi di berbagai industri.

2.4 BAGAIMANA SISTEM AI BEKERJA

Untuk memahami bagaimana sistem AI berfungsi, akan sangat membantu jika kita membandingkan prosesnya dengan proses otak manusia. Meskipun mesin semakin meniru aspek-aspek kecerdasan manusia, otak tetap tak tertandingi dalam hal kompleksitas, kemampuan beradaptasi, dan efisiensi energi.

- **Input dan Pemrosesan Data:** Otak manusia memproses input sensorik – seperti penglihatan, pendengaran, dan sentuhan – secara real-time, mengubah rangsangan mentah menjadi informasi yang bermakna dan respons yang bergantung pada konteks. Sebaliknya, sistem AI bergantung pada dataset pelatihan untuk mempelajari pola dan perilaku. Kinerja mereka sangat bergantung pada kualitas, keragaman, dan volume data, serta ketahanan algoritma yang mendasarinya.
- **Memori dan Penyimpanan:** Memori manusia beroperasi secara sadar dan bawah sadar, dengan efisiensi yang luar biasa dalam menyaring dan menyimpan informasi yang relevan. Namun, mesin menghadapi kendala dalam memori – khususnya dalam memori utama (Random Access Memory [RAM]). Meskipun penyimpanan sekunder menawarkan kapasitas yang lebih besar, ia lebih lambat dan kurang efisien untuk pengambilan data secara real-time.
- **Daya Komputasi:** Otak manusia mencapai pengambilan keputusan yang cepat melalui miliaran neuron yang saling terhubung. Mesin mengimbanginya dengan daya komputasi yang tinggi, tetapi ini membutuhkan biaya energi yang signifikan. Arsitektur canggih seperti GPU dan TPU telah meningkatkan efisiensi, namun tetap jauh kurang hemat energi dibandingkan sistem biologis.
- **Efisiensi Energi:** Otak mengonsumsi energi yang sangat sedikit untuk merasakan, menyimpan, dan memproses informasi. Sistem AI, khususnya yang menangani data skala besar, membutuhkan daya yang besar. Tantangan ini telah memicu minat pada paradigma hemat energi seperti komputasi neuromorfik dan komputasi kuantum.
- **Skalabilitas dan Kinerja:** Mesin unggul dalam menyimpan dan memproses kumpulan data yang sangat besar, melakukan penskalaan horizontal di seluruh arsitektur terdistribusi. Mereka dapat mengirimkan dan menghitung informasi dengan kecepatan jauh melampaui kemampuan manusia. Namun, kognisi manusia tetap unggul dalam

kemampuan beradaptasi, kreativitas, dan penalaran kontekstual – area di mana AI masih kesulitan.

2.5 EVOLUSI SISTEM AI

Seiring waktu, mesin telah berevolusi dari pengolah data sederhana menjadi sistem cerdas yang dapat belajar, bernalar, dan membuat keputusan secara real-time. Awalnya, komputer dirancang untuk melakukan tugas-tugas berulang secara efisien. Komponen intinya meliputi unit pengolah pusat (CPU) untuk komputasi, RAM untuk penyimpanan sementara, dan hard drive untuk penyimpanan data permanen. Di jantung CPU terdapat unit aritmatika logika (ALU), yang melakukan perhitungan matematika dasar dan operasi logika. Operasi ini dieksekusi berdasarkan instruksi yang diberikan oleh program perangkat lunak. Dengan kemajuan perangkat keras dan perangkat lunak, mesin tradisional memperoleh daya pemrosesan dan kecepatan yang lebih besar.

Namun, mesin-mesin awal tidak memiliki kecerdasan. Mereka hanya dapat mengeksekusi instruksi yang telah ditentukan dan terbatas pada penyelesaian masalah spesifik yang telah diprogramkan untuk mereka. Terlepas dari efisiensi komputasi dan kemampuan mereka untuk memproses volume data yang besar, mereka tidak dapat beradaptasi atau belajar dari pengalaman. Keterbatasan ini membuat para ahli bertanya: *Bisakah mesin pada akhirnya berpikir dan berevolusi seperti manusia?* Jawabannya adalah ya, dan dengan demikian konsep AI lahir. Pada akhir abad ke-20, kemajuan signifikan telah dicapai dalam pengembangan teknik AI. Inovasi dalam algoritma ML, penalaran berbasis kasus, dan pemahaman bahasa alami menandai titik balik dalam menanamkan kecerdasan ke dalam mesin. Sejak saat itu, sistem AI telah berkembang pesat, dengan beberapa ahli memprediksi bahwa AI pada akhirnya dapat melampaui kemampuan manusia di banyak bidang. Mari kita sekarang meninjau kembali secara singkat konsep kecerdasan.

2.6 KECERDASAN MANUSIA VS AI

Kecerdasan dapat dikonseptualisasikan melalui empat kemampuan inti: *Mempersepsikan*, *Memproses*, *Menyimpan*, dan *Melakukan*, yang masing-masing diterapkan pada masukan sensorik atau data. Sistem AI yang efektif bertujuan meniru pendekatan siklus ini, tetapi perbedaan mendasar antara manusia dan AI tetap ada :

- **Kecerdasan Manusia:** Kognisi manusia bersifat holistik dan multifaset, meliputi kreativitas, pemikiran abstrak, pemahaman emosional, interaksi sosial, dan penalaran etis. Tugas-tugas dasar seperti persepsi dan komunikasi bersifat intuitif dan sudah tertanam. Manusia unggul dalam menafsirkan isyarat non-verbal, beradaptasi dengan situasi baru, dan menerapkan konteks pada informasi. Kecerdasan emosional memungkinkan empati, penilaian moral, dan pengambilan keputusan berdasarkan pengalaman hidup.
- **Kecerdasan Buatan:** Sistem AI pada dasarnya digerakkan oleh data. Mereka belajar dari kumpulan data besar menggunakan algoritma dan model statistik, mencapai akurasi tinggi di domain tertentu. Namun, kinerja mereka dibatasi oleh kualitas dan keragaman

data pelatihan. AI tidak memiliki kesadaran, emosi, dan pemahaman kontekstual yang bernuansa. Tugas yang membutuhkan kreativitas, kemampuan beradaptasi, dan penalaran etis tetap menjadi tantangan bagi AI.

Perbedaan utama antara keduanya adalah:

- **Kemampuan beradaptasi:** Manusia menggeneralisasi pengetahuan di berbagai konteks; AI kesulitan dalam transfer pembelajaran di luar cakupan pelatihannya.
- **Kreativitas:** Manusia berinovasi dan berimajinasi; AI mengoptimalkan dalam parameter yang telah ditentukan.
- **Etika dan Emosi:** Manusia menerapkan penalaran moral dan empati; AI beroperasi berdasarkan logika dan data tanpa nilai intrinsik.
- **Kecepatan dan Skala:** AI melampaui manusia dalam kecepatan pemrosesan dan volume data tetapi kurang dalam penilaian holistik.

Singkatnya, kecerdasan manusia bersifat dinamis dan sadar konteks, dibentuk oleh pengalaman dan emosi, sedangkan AI terspesialisasi dan berorientasi pada tugas. Meskipun AI dapat mengungguli manusia dalam komputasi dan skalabilitas, ia tetap terbatas dalam generalisasi, penalaran etis, dan nuansa emosional.

2.7 APLIKASI UNTUK SISTEM AI

Kemajuan dalam teknologi AI dan Big Data telah memungkinkan organisasi untuk mengumpulkan, menyimpan, dan menganalisis sejumlah besar informasi dari berbagai sumber. Dengan memanfaatkan pendekatan algoritmik dan komputasi terdistribusi, sistem AI meningkatkan pengambilan keputusan, mengotomatiskan tugas, dan meningkatkan efisiensi di berbagai industri. Aplikasi utama meliputi:

- **Pemrosesan Bahasa Alami (NLP):** NLP menjembatani kesenjangan antara bahasa manusia dan pemahaman mesin. Ini mendukung aplikasi seperti chatbot, asisten suara, analisis sentimen, dan terjemahan mesin, memungkinkan interaksi manusia-komputer yang lancar.
- **Sistem Logika Fuzzy:** Sistem ini memungkinkan penalaran berdasarkan derajat kebenaran daripada logika biner. Logika fuzzy banyak digunakan dalam sistem kontrol, pengambilan keputusan dalam ketidakpastian, dan elektronik konsumen (misalnya, mesin cuci dan AC).
- **Robotika Cerdas:** Robotika menggabungkan AI dengan sistem mekanik untuk melakukan tugas berulang, kompleks, atau berbahaya secara otonom. Aplikasinya meliputi otomatisasi industri, eksplorasi ruang angkasa, dan robotika perawatan kesehatan untuk bantuan bedah dan perawatan pasien.
- **Sistem Pakar:** Sistem pakar meniru kemampuan pengambilan keputusan spesialis manusia menggunakan aturan yang telah ditentukan dan pengetahuan khusus domain. Sistem ini umumnya diterapkan dalam diagnosis medis, peramalan keuangan, dan dukungan teknis.

Aplikasi-aplikasi ini menggambarkan bagaimana sistem AI mentransformasi industri, meningkatkan efisiensi operasional, dan meningkatkan kemampuan manusia. Seiring terus

berkembangnya AI, integrasinya dengan Big Data akan mendorong solusi yang lebih canggih untuk tantangan dunia nyata. Setelah mengeksplorasi arsitektur Big Data dan sistem AI, menjadi jelas bahwa data adalah sumber kehidupan teknologi cerdas. Tanpa data berkualitas tinggi, bahkan algoritma tercanggih pun gagal memberikan wawasan yang bermakna. Langkah selanjutnya dalam membangun solusi AI dan analitik yang efektif adalah *pengumpulan data* – sebuah proses yang memastikan ketersediaan informasi yang akurat, relevan, dan terstruktur dengan baik untuk analisis dan pemodelan.

2.8 PENGUMPULAN DATA

Pengumpulan data adalah langkah dasar dalam setiap alur kerja analitik atau berbasis AI. Ini melibatkan pengumpulan dan pengukuran informasi secara sistematis tentang variabel yang diminati untuk menjawab pertanyaan pengembangan, menguji hipotesis, atau mengevaluasi hasil. Pengumpulan data yang efektif dimulai dengan mengidentifikasi sumber yang tepat, memahami jenis data, dan memilih metode yang memastikan keakuratan, keandalan, dan integritas. Pengumpulan data yang akurat memungkinkan organisasi dan ahli untuk:

- Mendeteksi tren dan pola
- Mengukur kemajuan dan kinerja
- Memvalidasi atau menantang hipotesis
- Mengevaluasi hasil dan intervensi
- Mengamati fenomena dalam konteks dunia nyata

Untuk menjaga integritas analisis dan menghindari kesimpulan yang menyesatkan, sangat penting untuk memastikan bahwa data yang dikumpulkan valid dan dapat diandalkan. Mari kita tinjau secara singkat dua jenis data utama :

- **Data Kuantitatif:** Data numerik yang dapat diukur, dihitung, atau dinyatakan dalam bentuk angka. Data ini mendukung perhitungan matematika dan analisis statistik. Contohnya termasuk tinggi badan, usia, dan berat badan. Data kuantitatif dapat berupa:
 - **Diskrit:** Mengambil nilai spesifik dan terpisah (misalnya, jumlah mahasiswa di kelas).
 - **Kontinu:** Dapat mengambil nilai apa pun dalam suatu rentang (misalnya, suhu, tinggi bangunan).
- **Data Kualitatif:** Data deskriptif yang tidak mudah dikuantifikasi. Data ini mencakup karakteristik, atribut, atau opini (misalnya, warna, emosi, atau umpan balik pelanggan).

Selanjutnya, kita akan membahas *skala pengukuran data*, yang sangat penting untuk memilih teknik statistik yang tepat. Ini termasuk :

- **Skala Nominal:** Mengkategorikan data tanpa urutan inheren (misalnya, jenis kelamin dan jenis buah). Analisis biasanya melibatkan modus, frekuensi, atau persentase.
- **Skala Ordinal:** Mengkategorikan data dengan urutan yang bermakna tetapi tanpa interval yang sama (misalnya, tingkat pendidikan dan peringkat kepuasan). Analisis dapat menggunakan modus, median, atau persentil.

- **Skala Interval:** Menampilkan kategori berurutan dengan interval yang sama tetapi tanpa titik nol yang sebenarnya (misalnya, suhu dalam Celcius dan tahun kalender). Analisis mencakup rata-rata, deviasi standar, dan statistik deskriptif lainnya.
- **Skala Rasio:** Mencakup semua properti data interval, ditambah titik nol yang sebenarnya (misalnya, pendapatan, berat, dan tinggi). Ini mendukung berbagai operasi statistik.

Analisis modern sering kali bergantung pada data yang tersedia untuk umum, terutama dari web. Tekniknya berkisar dari mengunduh dataset terstruktur hingga *web scraping*, yang mengekstrak data dari halaman web menggunakan skrip otomatis. Salah satu pustaka Python yang paling banyak digunakan untuk web scraping adalah *Beautiful Soup*, yang menyederhanakan penguraian dokumen HTML dan XML. Pada bagian selanjutnya, kami memperkenalkan Beautiful Soup dan mendemonstrasikan penggunaannya melalui contoh-contoh praktis, termasuk mengekstrak data tabular dari Wikipedia dan informasi buku dari Goodreads.

2.9 APA ITU BEAUTIFUL SOUP

Beautiful Soup adalah pustaka Python yang banyak digunakan untuk web scraping. Pustaka ini memungkinkan pengembang untuk mengurai dokumen HTML dan XML, sehingga memudahkan navigasi dan ekstraksi data dari halaman web. Beautiful Soup membuat pohon parse dari struktur dokumen, memungkinkan akses efisien ke elemen dan atribut tertentu. Untuk menginstal versi terbaru Beautiful Soup, gunakan perintah berikut:

```
pip install beautifulsoup4
```

Sekarang mari kita ambil data HTML dari sebuah situs web. Langkah pertama dengan Beautiful Soup adalah mengurai dokumen HTML menjadi objek soup. Tidak semua halaman web berisi tag HTML yang sepenuhnya valid. Oleh karena itu, Beautiful Soup perlu memperbaiki tag pembuka dan penutup yang tidak tepat. Misalnya, perhatikan halaman web sederhana ini yang berisi daftar dengan tanda kutip atribut dan tag penutup yang hilang:

```
<ul class=State>
  <li>Area
  <li>Population
</ul>
```

Dalam kasus ini, elemen `<li>` kedua mungkin salah diinterpretasikan sebagai anak dari `<li>` pertama, yang menyebabkan hasil parsing yang tidak terduga. Beautiful Soup mencoba untuk memperbaiki masalah tersebut secara otomatis. Tabel 2.1 menunjukkan skrip Python yang mengurai HTML yang rusak ini menggunakan parser bawaan `html.parser`: Tabel 2.2 menunjukkan output HTML yang diparsing.

Seperti yang terlihat di atas, elemen `<li>` bersarang dapat mempersulit navigasi lebih lanjut. Untungnya, Beautiful Soup mendukung parser alternatif seperti `lxml` dan `html5lib`, yang

sering menghasilkan pohon parse yang lebih bersih untuk HTML yang salah format. Pembaca didorong untuk bereksperimen dengan parser ini dengan memodifikasi skrip di Tabel 2.1.

TABEL 2.1 Skrip Python untuk Memparsing Konten *HTML*

```
from bs4 import BeautifulSoup
from pprint import pprint
broken_html = '<ul class=State><li>Area<li>Population</ul>'
soup = BeautifulSoup(broken_html, 'html.parser')
fixed_html = soup.prettify()
pprint(fixed_html)
```

TABEL 2.2 Output yang Diuraikan dari Beautiful Soup Menggunakan Parser *HTML*

```
<ul class="State">
<li>
Area
<li>
Population
</li>
</li>
</ul>
```

2.10 CONTOH WEB SCRAPING

Untuk menunjukkan kegunaan praktis Beautiful Soup, mari kita jelajahi dua contoh nyata pengambilan data dari web menggunakan Python.

- **Mengekstrak Informasi Negara Bagian dan Ibu Kota dari Wikipedia:** Dalam contoh ini, kita bertujuan untuk mengekstrak daftar negara bagian dan wilayah persatuan India beserta ibu kota administratif, legislatif, dan yudisialnya dari halaman Wikipedia. Untuk memulai, instal pustaka yang dibutuhkan:

```
pip install requests beautifulsoup4 pandas
```

Skrip Python pada Tabel 2.3 melakukan scraping.

Skrip ini mengurai konten HTML, mengidentifikasi tabel yang relevan, dan mengekstrak data terstruktur ke dalam Pandas DataFrame. Hasilnya adalah representasi tabular yang bersih dari negara bagian, wilayah persatuan, dan ibu kotanya seperti yang ditunjukkan pada Gambar 2.2.

- **Scraping Data Buku dari Goodreads:** Mari kita pertimbangkan contoh lain dari scraping data tekstual dari <https://www.goodreads.com> untuk mencantumkan 100 buku terbaik sepanjang masa. *Untuk setiap buku yang disebutkan di halaman ini, kita dapat mengekstrak detail berikut:* Peringkat, Judul, Penulis, Rating Rata-rata, Jumlah Rating, Skor, dan Jumlah Suara. Tabel 2.4 menyediakan skrip Python [8] yang menggunakan scraper Beautiful Soup, mengurai halaman sumber, dan menyimpan data ke dalam file CSV. Setelah menjalankan skrip pada Tabel 2.4, kita dapat memperoleh daftar 100 buku terbaik (lihat Gambar 2.3). Analisis selanjutnya dapat dilakukan pada data yang diambil

dari tautan web ini. Misalnya, kita dapat mengeksplorasi data untuk mendapatkan skor peringkat rata-rata dari sepuluh judul buku teratas (lihat Gambar 2.4).

TABEL 2.3 Skrip Python untuk Mengekstrak Konten dari Halaman Wikipedia

```
# import the library used to query a website
import requests
from bs4 import BeautifulSoup

# Query the website and return the html to the variable 'page'

page = requests.get('https://en.wikipedia.org/wiki/
    List_of_state_and_union_territory_capitals_in_India')
html = page.text

# Parse the html in the 'page' variable, and store it in Beautiful
    Soup format
soup = BeautifulSoup(html, 'html.parser')

# prettify is used to look at nested structure of HTML page
print(soup.prettify())

# Here, all the links with the page's <a> tag are fetched
all_links = soup.find_all("a")
for link in all_links:
    print(link.get("href"))

# We are looking for the particular table to collect the required
    information
all_tables = soup.find_all('table')

# Once the table name is located, we have to use the "class" of the
    table to identify
right_table = soup.find('table', class_='wikitable sortable
    plainrowheaders')
right_table

# We will be generating lists to load the data into a dataframe
A = []
B = []
C = []
D = []
E = []
F = []
G = []
for row in right_table.findAll("tr"):
    cells = row.findAll('td')
    states = row.findAll('th') # To store second column data
    if len(cells) == 6: # Only extract table body not heading
        A.append(cells.find(text=True))
        B.append(states.find(text=True))
        C.append(cells.find(text=True))
        D.append(cells.find(text=True))
```

```
E.append(cells.find(text=True))
F.append(cells.find(text=True))
G.append(cells.find(text=True))
```

```
# The pandas package is used to convert the data to data frame
import pandas as pd
df = pd.DataFrame(A, columns=['Number'])
df['State/UT'] = B
df['Admin_Capital'] = C
df['Legislative_Capital'] = D
df['Judiciary_Capital'] = E
df['Year_Capital'] = F
df['Former_Capital'] = G
df
```

No		Negara Bagian/Wilayah	Ibu Kota Administratif	Ibu Kota Legislatif	Ibu Kota Yudisial	Tahun Pembentukan	Ibu Kota Sebelumnya
0	1	Kepulauan Andaman dan Nikobar	Port Blair	Port Blair	Kolkata	1955	Calcutta
1	2	Andhra Pradesh	Hyderabad	Amaravati	Hyderabad	1956	Kurnool
2	3	Arunachal Pradesh	Itanagar	Itanagar	Guwahati	1986	Tidak ada
3	4	Assam	Dispur	Guwahati	Guwahati	1975	Shillong
4	5	Bihar	Patna	Patna	Patna	1912	Tidak ada
5	6	Chandigarh	Chandigarh	-	Chandigarh	1966	-
6	7	Chhattisgarh	Nava Raipur	Raipur	Bilaspur	2000	-
7	8	Dadra dan Nagar Haveli	Silvassa	-	Mumbai	1945	Mumbai
8	9	Daman dan Diu	Daman	-	Mumbai	1987	Ahmedabad
9	10	Wilayah Ibu Kota Nasional Delhi	New Delhi	New Delhi	New Delhi	1931	Tidak ada
10	11	Goa	Panaji	Porvorim	Mumbai	1961	Panaji
11	12	Gujarat	Gandhinagar	Gandhinagar	Ahmedabad	1960	Ahmedabad

1 2	1 3	Haryana	Chandigarh	Chandigarh	Chandigarh	1966	-
1 3	1 4	Himachal Pradesh	Shimla	Shimla (Musim Panas)	Shimla	1971	Bilaspur
1 4	1 5	Jammu dan Kashmir	Srinagar	Srinagar (Musim Panas)	Srinagar (Musim Panas)	1947	-
1 5	1 6	Jharkhand	Ranchi	Ranchi	Ranchi	2000	Tidak ada
1 6	1 7	Karnataka	Bengaluru	Bengaluru	Bengaluru	1940	(Mysore)
1 7	1 8	Kerala	Thiruvananthapuram	Thiruvananthapuram	Kochi	1956	Tidak ada
1 8	1 9	Lakshadweep	Kavaratti	Kavaratti	Kochi	1956	Tidak ada
1 9	2 0	Madhya Pradesh	Bhopal	Bhopal	Jabalpur	1956	Nagpur

Gambar 2.2 Negara bagian, wilayah persatuan, dan ibu kotanya yang diambil dari halaman Wikipedia

TABEL 2.4 Skrip Python untuk Mengambil Data guna Mendaftarkan 100 Buku Terbaik Sepanjang Masa di Goodreads

```
import requests
from bs4 import BeautifulSoup as BS
import pandas
import os.path
import nltk
from nltk.corpus import stopwords

from nltk.probability import FreqDist
import matplotlib.pyplot as plt

# URL Information
url = 'https://www.goodreads.com/list/show/1?ref=ls_pl_car_0'
headers = {'User-Agent': '5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36'}
urlContentFile = "Url-Content.txt"

# Init the content variable
soup = None

# Check if the URL Content has already been stored in a local file
if os.path.isfile(urlContentFile):
    # Open the URL Content text file and convert it to soup
    soup = BS(open(urlContentFile, encoding='utf-8'), 'html.parser')
else:
    # Request the URL Content
```

```
r = requests.get(url, headers=headers)

# Check status code for response received (success code: 200)
print("Request Code: " + str(r))

# Using beautiful soup parse the whole HTML request
soup = BS(r.content, 'html.parser')

# Save the Soup content to a text file for future parsing
file = open(urlContentFile, "w", encoding='utf-8')
file.write(str(soup))
file.close()

# Find all of the HTML Elements we will need to parse
rank = soup.find_all('td', class_='number')
titles = soup.find_all('span', itemprop='name', role='heading')
names = soup.find_all('a', class_='authorName')
ratings = soup.find_all('span', class_='greyText smallText uitext')
scores = soup.find_all('span', class_='smallText uitext')
# Define the Keys for the dictionary
keyList = [
    'Rank',
    'Title',
    'Author',
    'Avg Rating',
    'Ratings Count',
    'Score',
    'Number of Votes']

# Init the dictionary
d = {}

# Setup the Dictionary's Keys
for i in keyList:
    d[i] = []

# Loop over the 100 Books
for i in range(100):
    d['Rank'].append(int(rank[i].string))
    d['Title'].append(titles[i].string)
    d['Author'].append(" ".join(names[i].string.split()))
    d['Avg Rating'].append(float(ratings[i].find('span').text.
split('-').split(' ')[-4]))
    d['Ratings Count'].append(int(ratings[i].find('span').text.
split('-').split(' ').replace(',', ' ')))
    d['Score'].append(int(scores[i].text.split(' ')[1][:2].
replace(',', ' ')))
    d['Number of Votes'].append(int(scores[i].text.splitlines().
split('
').replace(',', ' ')))

# Lets convert our dictionary to a data frame and save it as a CSV
df = pandas.DataFrame(d)
df.to_csv('Output.csv', index=False)
```

```
# Analysis on txt

txtFile = pandas.read_csv('Output.csv')

print(type(txtFile))

print(len(txtFile))

print(txtFile)
df_string = txtFile.to_string()
print(type(df_string))

print(len(df_string))

print(df_string)

words = nltk.word_tokenize(df_string)
print(words)

removePunctuation = []
for x in words:
    if x.isalpha():
        removePunctuation.append(x.lower())

print(removePunctuation)
print(len(removePunctuation))
stopwords = stopwords.words("english")

cleanedWords = []

for y in removePunctuation:
    if y not in stopwords:
        cleanedWords.append(y)

print(cleanedWords)
print(len(cleanedWords))

fdist = FreqDist(words)
print(fdist.most_common(10))
fdist.plot(30)
plt.show()
fdist = FreqDist(removePunctuation)
print(fdist.most_common(10))
fdist.plot(30)
plt.show()

fdist = FreqDist(cleanedWords)
print(fdist.most_common(10))
fdist.plot(30)
plt.show()

topTen = txtFile.head(10)
```

```
# AVG RATING
plt.plot(topTen['Title'],topTen['Avg
Rating'], marker='o',label='Avg Rating',color='pink')
plt.xlabel('Title', fontsize=10)
plt.ylabel('Values', fontsize=10)
plt.title('Average Rating by Top 10 Titles', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=6)
plt.legend()
plt.tight_layout()
plt.show()

# SCORE
plt.plot(topTen['Title'],topTen['Score'],marker='o',label='Avg
Rating',color='purple')
plt.xlabel('Title', fontsize=10)
plt.ylabel('Values', fontsize=10)
plt.title('Score by Top 10 Titles', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=6)
plt.legend()
plt.tight_layout()
plt.show()
```



Rank	Title	Author	Avg Rating	Ratings Count	Score	Number of Votes
1	The Hunger Games (The Hunger Games, #1)	Suzanne Collins	4.34	9225819	4019846	48885
2	Harry Potter and the Order of the Phoenix (Harry Potter, #5)	J.K. Rowling	4.5	3583148	3195727	32645
3	Pride and Prejudice	Jane Austen	4.29	4478458	2833034	29043
4	To Kill a Mockingbird	Harper Lee	4.26	6585357	2502887	25561
5	The Book Thief	Markus Zusak	4.39	2721329	1884669	19367
6	Twilight (The Twilight Saga, #1)	Stephenie Meyer	3.66	6944038	1720662	17537
7	Animal Farm	George Orwell	4.0	4169885	1616968	16745
8	J.R.R. Tolkien 4-Book Boxed Set: The Hobbit and The Lord of the Rings	J.R.R. Tolkien	4.61	138873	1569821	162
9	The Chronicles of Narnia (The Chronicles of Narnia, #1-7)	C.S. Lewis	4.28	681815	1481807	15442
10	The Fault in Our Stars	John Green	4.13	5437191	1378025	14181
11	The Picture of Dorian Gray	Oscar Wilde	4.13	1698738	1293559	13501
12	The Giving Tree	Shel Silverstein	4.38	1181891	1203860	12462
13	Wuthering Heights	Emily Brontë	3.9	1987481	1199597	12517
14	Gone with the Wind	Margaret Mitchell	4.31	1243455	1178799	12155
15	The Lightning Thief (Percy Jackson and the Olympians, #1)	Rick Riordan	4.31	3232702	1154247	12076
16	The Perks of Being a Wallflower	Stephen Chbosky	4.23	1895223	1132791	11793
17	Jane Eyre	Charlotte Brontë	4.15	2200136	1095010	11508
18	The Little Prince	Antoine de Saint-Exupéry	4.33	2271364	1091197	11506
19	The Great Gatsby	F. Scott Fitzgerald	3.93	5650658	1061330	11302
20	The da Vinci Code (Robert Langdon, #2)	Dan Brown	3.92	2432652	1001670	10524
21	Crime and Punishment	Fyodor Dostoevsky	4.28	976499	1000001	10408
22	Alice's Adventures in Wonderland / Through the Looking-Glass	Lewis Carroll	4.06	585693	980233	10333
23	Divergent (Divergent, #1)	Veronica Roth	4.14	4184477	970954	10180
24	Memoirs of a Geisha	Arthur Golden	4.16	2054626	950356	9900

Gambar 2.3 Daftar buku terbaik sepanjang masa di Goodreads.



Gambar 2.4 Peringkat rata-rata sepuluh judul buku terbaik sepanjang masa.

Contoh-contoh ini mengilustrasikan bagaimana Beautiful Soup dapat digunakan untuk mengekstrak data terstruktur dan tidak terstruktur dari web. Teknik-teknik tersebut merupakan dasar untuk membangun sistem AI yang bergantung pada data dunia nyata untuk pelatihan, evaluasi, dan penerapan. Pada bab selanjutnya, kita akan mengeksplorasi teknik pembersihan data untuk mempersiapkan data yang hasil scraping untuk analisis dan pemodelan.

2.11 RINGKASAN

Bab ini memperluas konsep-konsep dasar yang diperkenalkan pada Bab 1 dengan mengeksplorasi arsitektur dan komponen sistem *Big Data* dan AI modern. Kita mulai dengan tinjauan umum kerangka kerja Big Data seperti *Hadoop*, *Spark*, dan basis data *NoSQL*, menekankan peran mereka dalam memungkinkan penyimpanan dan pemrosesan kumpulan data besar yang terukur dan terdistribusi. Teknologi-teknologi ini membentuk tulang punggung aplikasi berbasis data kontemporer. Selanjutnya, kita meneliti struktur dan kemampuan sistem AI, menyoroti kemampuan mereka untuk belajar, bernalar, dan mengambil keputusan. Analisis komparatif antara *kecerdasan manusia* dan AI menggarisbawahi perbedaan utama dalam kemampuan beradaptasi, kreativitas, pemahaman emosional, dan efisiensi komputasi. Meskipun AI unggul dalam kecepatan dan skalabilitas, kognisi manusia tetap lebih unggul dalam penalaran kontekstual dan penilaian etis.

Bab ini juga memperkenalkan aplikasi praktis AI, termasuk NLP, *sistem logika fuzzy*, *robotika cerdas*, dan *sistem pakar*, yang menggambarkan bagaimana teknologi ini mentransformasi industri dan meningkatkan kemampuan manusia. Terakhir, kami membahas *pengumpulan data* sebagai pendahulu penting untuk analitik dan pengembangan AI. Kami meninjau jenis data, skala pengukuran, dan teknik untuk memperoleh data dari sumber publik, termasuk web scraping dengan *Beautiful Soup*. Contoh dunia nyata menunjukkan bagaimana data terstruktur dan tidak terstruktur dapat diekstrak dari sumber daya online, meletakkan dasar untuk bab-bab selanjutnya tentang pembersihan dan persiapan data. Ke depannya, kita akan membangun konsep-konsep ini untuk mengeksplorasi bagaimana data yang bersih dan terstruktur dengan baik mendorong model ML yang efektif dan sistem pengambilan keputusan yang cerdas.

2.12 PERTANYAAN

1. Jelaskan bagaimana Spark, Kafka, NoSQL, dan Hadoop bekerja bersama dalam stack Big Data/AI modern. Kapan Anda akan memilih masing-masing?
2. Dalam hal (a) input data, (b) penyimpanan/pengambilan, dan (c) kemampuan pemrosesan, bandingkan kecerdasan manusia dan mesin.
3. Berikan penjelasan singkat tentang masing-masing fitur utama dan subbidang sistem AI.
4. Untuk aliran data real-time, komponen mana yang biasanya berfungsi sebagai perantara pesan terdistribusi antara penyedia dan konsumen data?
 - A. Hadoop

- B. Spark
 - C. Kafka
 - D. NoSQL
5. Cabang kecerdasan buatan mana yang dibedakan oleh penerapan jaringan saraf tiruan multi-lapisan?
- A. NLP
 - B. Sistem Pakar
 - C. Pembelajaran Mendalam
 - D. Visi Komputer
6. Mengenai pemrosesan batch dalam sistem big data, pernyataan berikut yang manakah yang BENAR?
- A. Harus menghasilkan hasil dengan cepat dan bekerja dengan data yang tak ada habisnya dan terus menerus datang.
 - B. Sistem ini menggunakan kumpulan data statis dan terbatas yang disimpan secara permanen; volume data yang sangat besar mengakibatkan waktu pemrosesan yang lebih lama.
 - C. Hanya penggabungan streaming waktu nyata dengan latensi di bawah satu detik yang didukung.
 - D. Sistem ini tidak mampu menangani kumpulan data yang sangat besar.

BAB 3

PERANCANGAN ALUR PIPELINE BIG DATA

3.1 KONSEP ALUR DATA

Pada bagian ini, kita akan mengeksplorasi konsep pipeline data dan bagaimana arsitektur pipeline data dapat dirancang untuk mendukung pemrosesan data yang terukur dan cerdas. Meskipun pipeline data telah diperkenalkan secara singkat pada bab-bab sebelumnya, fokus kita di sini adalah pada desain arsitektur dan implementasinya. Mari kita mulai dengan definisi formal:

Pipeline data adalah kumpulan alat dan proses yang digunakan untuk mengumpulkan data, mengubahnya menjadi wawasan, melatih model, memberikan hasil, dan menerapkan model tersebut di mana pun dan kapan pun tindakan diperlukan untuk mencapai tujuan bisnis.

Pada intinya, pipeline data mengotomatiskan proses ETL (Ekstrak, Transformasi, Muat). Arsitektur pipeline data yang dirancang dengan baik menangkap, mengatur, dan mengarahkan data untuk memungkinkan pelaporan, analisis, dan pengambilan keputusan yang efisien. Pipeline data dapat dibayangkan sebagai kombinasi khusus dari teknologi perangkat lunak dan protokol yang mengelola, memvisualisasikan, mengubah, dan memindahkan data dari berbagai sumber sesuai dengan tujuan bisnis. Pipeline data tipikal terdiri dari lima komponen utama:

1. **Modul Pengumpulan Data:** Mengumpulkan data dari berbagai sumber seperti perangkat seluler, sensor, dan aplikasi web.
2. **Modul Pengambilan Data:** Menerima data melalui protokol seperti HTTP atau MQTT, seringkali dalam bentuk blob atau stream.
3. **Modul Transformasi Data:** Membersihkan, menyusun, dan memformat data untuk pemrosesan selanjutnya.
4. **Modul Komputasi/Pembelajaran Mesin (ML):** Menerapkan algoritma untuk mengekstrak wawasan atau melatih model.
5. **Modul Presentasi Data:** Menyampaikan wawasan kepada pengambil keputusan melalui dasbor, laporan, atau API.

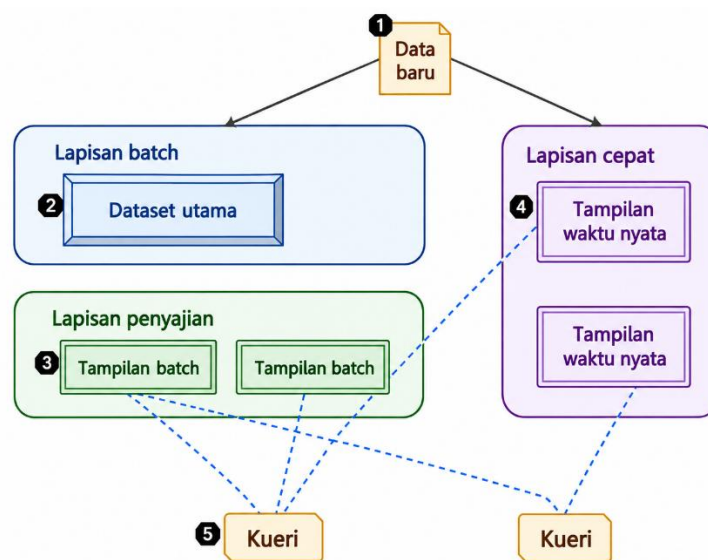
Di seluruh komponen ini, berbagai aplikasi digunakan untuk melakukan operasi pada data sebelum diteruskan ke tahap berikutnya. Misalnya, Apache Kafka umumnya digunakan sebagai layanan pengiriman pesan terdistribusi untuk mendukung pengumpulan data. Kafka menyederhanakan koneksi klien dengan mengantre dan menyiarkan pesan di seluruh topik.

Setelah dimasukkan, data biasanya dialihkan ke data lake atau area staging, di mana data tersebut disimpan sementara dalam bentuk mentah. Selama fase persiapan, data diubah menjadi format terstruktur dan dipindahkan ke data warehouse untuk penyimpanan dan akses yang aman. Operasi ETL dilakukan di sini untuk mengubah data terstruktur dan tidak terstruktur menjadi format yang dapat dikonsumsi oleh aplikasi dan pipeline ML. Sebelum

mencapai lapisan presentasi, komputasi dilakukan untuk mengekstrak intelijen bisnis dan wawasan yang dapat ditindaklanjuti. Wawasan ini kemudian disampaikan kepada pemangku kepentingan melalui alat visualisasi atau sistem otomatis. Pipeline Big Data modern bersifat luas dan fleksibel. Pipeline ini mendukung berbagai macam sumber data – terstruktur dan tidak terstruktur – yang disimpan di lingkungan cloud atau sistem terdistribusi. Beberapa pipeline ETL dapat memberi masukan ke berbagai platform dan aplikasi intelijen bisnis. Selanjutnya, mari kita bahas arsitektur Lambda.

3.2 ARSITEKTUR LAMBDA

Seperti yang diperkenalkan pada Bab 2, lanskap Big Data mencakup berbagai teknologi dan paradigma untuk memproses data skala besar. Salah satu paradigma yang banyak diadopsi adalah arsitektur Lambda, yang mengintegrasikan pemrosesan batch dan stream untuk mendukung analitik yang terukur dan toleran terhadap kesalahan (lihat Gambar 3.1). Arsitektur Lambda adalah model hibrida yang dirancang untuk menangani volume data yang sangat besar sekaligus memungkinkan analisis real-time dan historis. Sistem ini terdiri dari tiga lapisan :



Gambar 3.1 Arsitektur Lambda.

1. **Lapisan Batch:** Lapisan ini terus menerus memasukkan data mentah dan melakukan operasi ETL untuk membersihkan dan menyusunnya. Data yang telah diproses disimpan dalam gudang data dan digunakan untuk menghasilkan tampilan batch. Biasanya, lapisan ini beroperasi berdasarkan jadwal (misalnya, harian atau per jam) dan memastikan konsistensi dan kelengkapan data.
2. **Lapisan Penyajian:** Lapisan penyajian mengelola kumpulan data utama dan mengindeks tampilan batch untuk kueri yang efisien. Lapisan ini memungkinkan akses ad hoc ke hasil yang telah dihitung sebelumnya dan berfungsi sebagai jembatan antara lapisan batch dan lapisan kecepatan.

3. **Lapisan Kecepatan (Lapisan Aliran):** Lapisan ini menangani data real-time yang belum diproses oleh lapisan batch. Lapisan ini membuat tampilan real-time untuk memberikan wawasan langsung berdasarkan data terbaru. Hal ini memastikan bahwa pengguna memiliki akses ke informasi historis dan terkini.

Arsitektur Lambda menawarkan beberapa keunggulan:

- **Skalabilitas:** Dapat secara otomatis menyesuaikan kapasitas untuk menangani beban data yang bervariasi.
- **Toleransi Kesalahan:** Mekanisme bawaan memastikan ketersediaan dan ketahanan yang tinggi.
- **Operasi Tanpa Server:** Manajemen infrastruktur minimal diperlukan.
- **Responsif Waktu Nyata:** Beradaptasi dengan kondisi data yang berubah dengan latensi rendah.

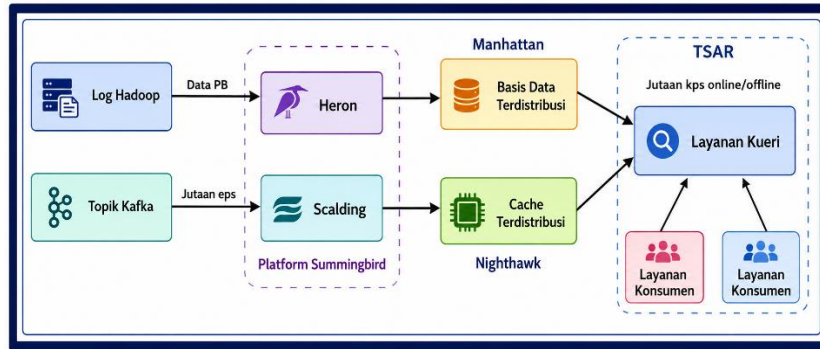
Namun, arsitektur ini juga menghadirkan tantangan. Implementasi arsitektur Lambda membutuhkan pemeliharaan dua basis kode terpisah – satu untuk pemrosesan batch dan satu lagi untuk pemrosesan stream – yang meningkatkan kompleksitas dan membuat debugging lebih sulit.

Arsitektur Pipeline Big Data: Twitter, Pinterest, dan Airbnb

Untuk memahami bagaimana arsitektur Lambda diterapkan dalam skenario dunia nyata, kami meneliti arsitektur *pipeline Big Data* dari tiga perusahaan besar: *Twitter*, *Pinterest*, dan *Airbnb*. Masing-masing organisasi ini memproses volume data yang sangat besar dan telah mengembangkan pipeline yang disesuaikan untuk memenuhi kebutuhan operasional unik mereka.

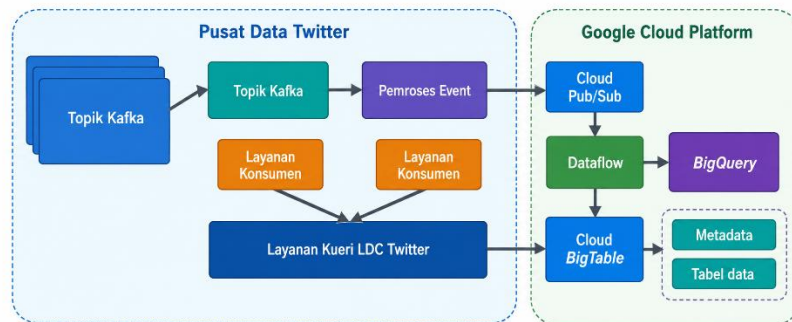
3.3 ARSITEKTUR PIPELINE TWITTER

Twitter memproses miliaran tweet setiap hari, menghasilkan petabyte data. Untuk mengelola skala ini, pipeline Twitter mengintegrasikan platform seperti Hadoop, Kafka, BigQuery, *Pub/Sub*, Vertica, dan Manhattan. Arsitektur ini mengikuti paradigma Lambda, menggabungkan pemrosesan batch dan stream. Dalam pipeline data ini (lihat Gambar 3.2), data dimasukkan secara bersamaan ke *Heron* dan *Scalding* (batch). Di dalam lapisan batch, ETL dilakukan, dan data yang telah diproses kemudian diteruskan ke dataset Master. Lapisan kecepatan menyimpan data hingga siap untuk diteruskan ke lapisan tampilan batch. Sumber untuk komponen batch adalah log Hadoop, seperti event klien, event timeline, dan event Tweet, yang disimpan di HDFS. Pipeline data Twitter juga menggunakan beberapa pipeline *Scalding* untuk memproses log mentah dan memasukkannya ke dalam platform Summingbird, yang merupakan sumber offline. Log ini juga dimasukkan ke dalam sistem penyimpanan terdistribusi Manhattan. Di dalam modul pemrosesan aliran, data real-time dari Kafka disimpan dalam cache terdistribusi Nighthawk. Layanan TSAR Query kemudian dapat mengakses data real-time dan data yang disimpan secara batch berdasarkan kueri dari pelanggan. Untuk memastikan toleransi kesalahan, data direplikasi di tiga pusat data.



Gambar 3.2 Diagram blok arsitektur Twitter.

Seiring pertumbuhan platform, arsitektur saat ini menghadapi tantangan yang semakin besar dengan throughput data yang tinggi. Tantangan ini menyebabkan kehilangan data dan ketidakakuratan dalam pipeline. *Heron bolts* yang menerima data dari Kafka menimbulkan biaya pengumpulan sampah yang tinggi, sehingga memperlambat kinerja pipeline. Saat tekanan mulai meningkat, lag sistem terus bertambah. Untuk mengatasi masalah ini, perusahaan telah mencoba memanfaatkan kekuatan komputasi awan menggunakan Google Cloud (GC) dan telah merestrukturisasi arsitektur pipeline data yang ada. Gambar 3.3 mengilustrasikan arsitektur pipeline data Twitter yang telah dimodifikasi. Dalam arsitektur yang dimodifikasi, alur kerja dibagi antara pusat data Twitter sendiri dan Google Cloud (GC).

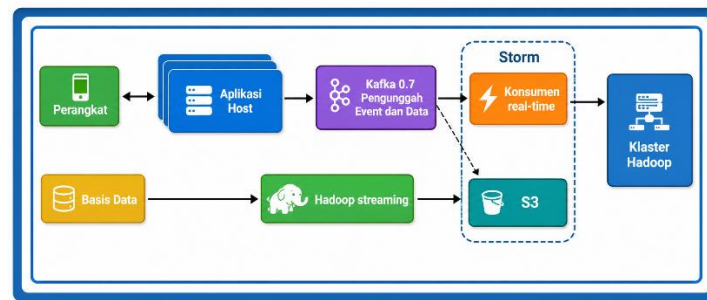


Gambar 3.3 Implementasi diagram blok arsitektur Twitter yang telah dimodifikasi.

Pada fase pertama, beberapa alur kerja pra-pemrosesan mengirimkan peristiwa atau tweet ke topik Kafka. Kemudian, prosesor peristiwa digunakan untuk mengalirkan data menggunakan semantik "setidaknya sekali". Proses peristiwa mengubah data menjadi Pub/Sub dan menghasilkan ID konteks peristiwa dan informasi meta lainnya. Baik data yang telah diubah maupun metadata kemudian dikirim dari pusat data ke GC. Di dalam GC, Twitter menggunakan kerangka kerja internalnya sendiri untuk menyelesaikan agregasi dan penghapusan duplikasi secara real-time. Alur kerja menggunakan *BigQuery* Google untuk terus memeriksa duplikasi. Sebagai langkah terakhir, jumlah agregat dan kunci kueri ditulis ke dalam *BigTable*. Di setiap tahap, aliran data dipantau, dan integritasnya diperiksa.

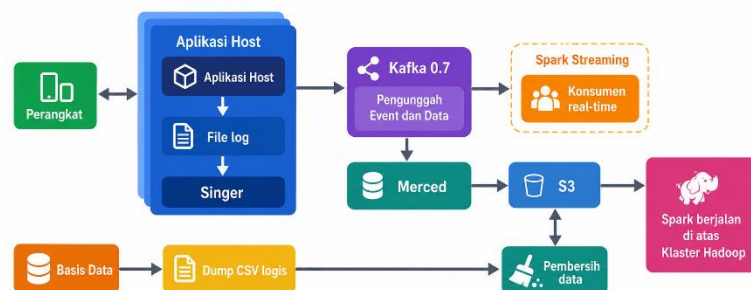
3.4 ARSITEKTUR PIPELINE PINTEREST

Pinterest sangat bergantung pada infrastrukturnya, yang terdiri dari *Apache Kafka*, *Storm*, *Hadoop*, *HBase*, dan *Redshift*, untuk melayani lebih dari 10 miliar tampilan halaman per bulan. Ada dua kategori utama kumpulan data yang dihasilkan Pinterest: (1) log layanan online dan (2) dump basis data. Log layanan adalah aliran log berkelanjutan yang *dihasilkan oleh layanan yang berjalan di ribuan host*. Di sisi lain, dump basis data adalah cadangan basis data logis, yang dihasilkan setiap jam atau setiap hari. Di sisi layanan online, Kafka digunakan sebagai pengangkut pesan pusat. Pesan log langsung ditulis ke Kafka oleh server aplikasi. Terakhir, pengunggah data di setiap Kafka mengunggah file log Kafka ke S3. Pekerjaan streaming Hadoop kemudian bertanggung jawab untuk mengambil data dari basis data dan menuliskannya ke S3 (lihat Gambar 3.4).



Gambar 3.4 Diagram blok arsitektur Pinterest.

Alur yang ditentukan pada Gambar 3.4 memiliki overhead operasi yang sangat tinggi. Sebagai contoh, mari kita pertimbangkan apa yang terjadi jika broker Kafka mengalami kegagalan. Jika broker Kafka mengalami gangguan atau kegagalan, maka akan menyebabkan kehilangan data, dan failover dump database yang sering terjadi akan meningkatkan kegagalan pekerjaan ini. Untuk mengatasi masalah ini, arsitektur pipeline *data yang dimodifikasi untuk Pinterest* diusulkan (lihat Gambar 3.5).



Gambar 3.5 Diagram blok arsitektur pipeline Pinterest yang diperbarui.

Pesan log sekarang ditulis langsung ke disk lokal alih-alih langsung menulis ke Kafka. Agen logging, yaitu Singer, kemudian mengunggah pesan log dari host ke Kafka. Data dari Kafka kemudian dipindahkan ke S3 menggunakan layanan penyimpanan data yang disebut *Merced*.

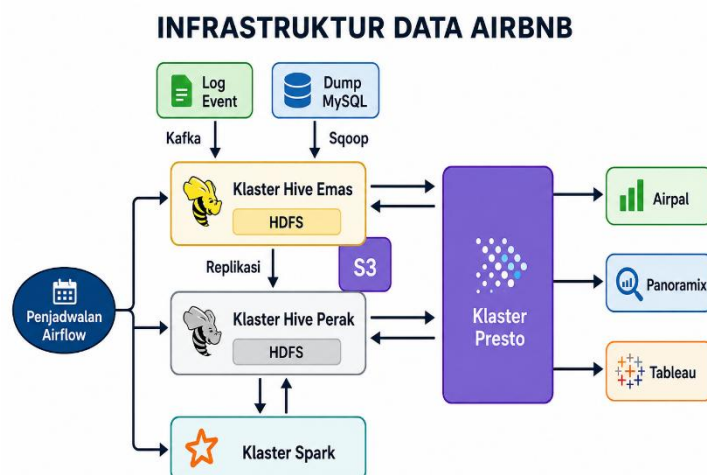
Merced menggunakan pendekatan master worker untuk mendistribusikan beban kerja penyimpanan data di antara worker.

Basis data langsung menyimpan dump logis ke S3 dan menggunakan kerangka kerja sanitasi untuk menghasilkan dataset. Sanitasi juga digunakan untuk pemeriksaan penegakan skema, deduplikasi, penegakan kebijakan data, dan lain-lain. Karena audit ujung-ke-ujung dan pemeriksaan kelengkapan, serta peningkatan visibilitas, overhead operasi berkurang. Hal ini mengakibatkan pipeline data kini memiliki kemampuan untuk menangani lebih dari 150 miliar pesan per hari dan lebih dari 50 TB dump basis data logis. Namun dengan meningkatnya jumlah pengguna, muncul tantangan baru karena pembuatan dump logis tertunda akibat failover master basis data. Hal ini mencegah data baru tiba tepat waktu untuk analisis hilir. Dikombinasikan dengan peningkatan jumlah klaster Kafka dan keterlambatan kedatangan peristiwa (data), operasi Kafka menghadapi tantangan yang berat.

3.5 ARSITEKTUR PIPELINE AIRBNB

Pipeline data Airbnb menggunakan *Kafka* untuk mengumpulkan log peristiwa dan dump *MySQL*, dan *Spark* untuk memprosesnya (lihat Gambar 3.6). Pipeline ini terdiri dari komponen-komponen berikut :

- **Apache Kafka** digunakan untuk membuat jembatan dari log peristiwa (cache memori) ke klaster HDFS (Hadoop Distributed File System) Hive.
- **Apache Sqoop** digunakan untuk mentransfer data relasional dari MySQL ke klaster HDFS/Hive.
- **Apache HIVE** digunakan untuk menyimpan data dalam gudang data OLAP (format kolom) dan memeliharanya:
 - **Klaster Gold Hive:** Pertama, data dari log dan MySQL disimpan dalam tabel Hive yang dipartisi, yang dikenal sebagai klaster gold hive.



Gambar 3.6 Arsitektur data Airbnb.

- **Klaster Silver Hive:** Data dalam klaster gold hive juga direplikasi ke klaster hive lain yang disebut klaster silver hive. Ini memastikan bahwa data tersedia untuk sumber daya yang dapat mengimplementasikan kueri ad hoc intensif, analisis, dan pelaporan untuk penggunaan bisnis.

Meskipun replikasi data yang sama mahal, ketersediaan data dijamin ketika klaster hive sedang tidak aktif.

- **Apache Spark:** Digunakan untuk mentransformasi atau menghitung data yang dibutuhkan untuk analisis bisnis.
- **Apache Presto:** Digunakan untuk melakukan query data dari tabel Silver Hive dalam format SQL untuk pelaporan ad hoc.
- **Apache Airflow:** Digunakan untuk menjadwalkan setiap komponen alur kerja pipeline data dalam bentuk DAG (Directed Acyclic Graph).
- **Tableau:** Presto terhubung langsung ke perangkat lunak visualisasi Tableau untuk memantau KPI bisnis dan untuk pelaporan.
- **Airpal:** Antarmuka web yang digunakan untuk menjalankan query ad hoc pada data warehouse Silver Hive.
- **AWS S3:** S3 Glacier merupakan kelas penyimpanan terpisah untuk data arsip dengan biaya yang lebih rendah.

Pipeline data ini memiliki beberapa keunggulan seperti yang tercantum di bawah ini

1. Semua komponen utama adalah perangkat lunak open-source.
2. Kecepatan baca/tulis disk hampir 400+MB/detik.
3. Job Hive dapat dieksekusi jauh lebih cepat menggunakan unit pemrosesan pusat (CPU).
4. Visibilitas yang sangat tinggi dari job yang sedang berjalan dan file log.
5. Visibilitas kesehatan klaster yang sangat tinggi.
6. Mampu melakukan kueri ad hoc yang kompleks tanpa memengaruhi klaster gold hive.

3.6 STRATEGI MEMBUAT PIPELINE BIG DATA

Mendesain pipeline Big Data membutuhkan pertimbangan yang cermat terhadap kasus penggunaan bisnis, karakteristik data, dan kualitas sistem yang diinginkan. Arsitektur dapat berkisar dari sederhana hingga sangat kompleks, tergantung pada ruang lingkup dan skala aplikasi. Arsitektur pipeline sederhana biasanya mencakup:

- **Pengumpulan data** dari berbagai sumber
- **Pemuatan data** ke dalam repositori (misalnya, sistem file, basis data relasional, gudang data, atau data lake)
- **Komputasi dan analisis** menggunakan algoritma ML/AI

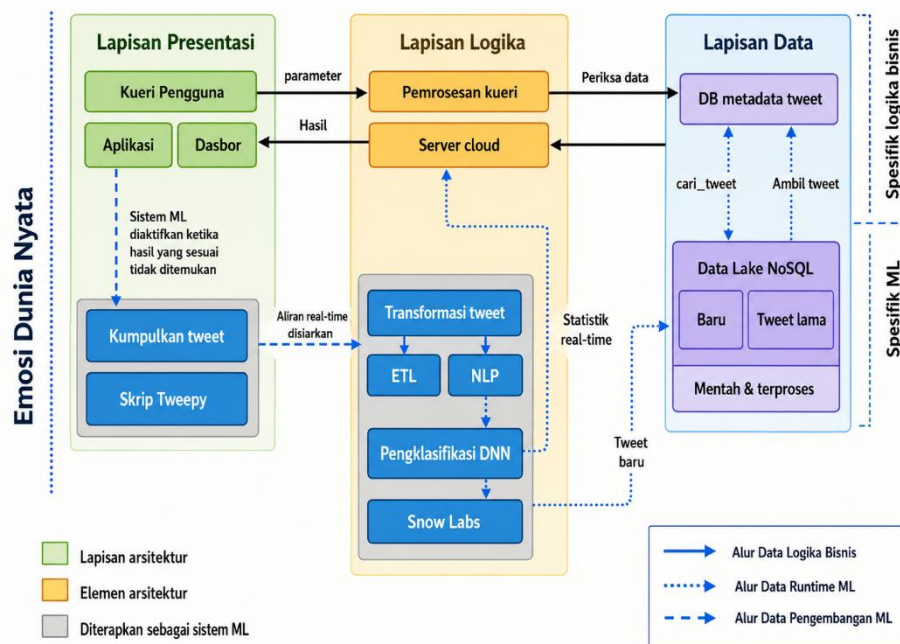
Meskipun arsitektur tersebut mungkin cukup untuk kebutuhan dasar, arsitektur tersebut sering menghadapi keterbatasan karena:

- Heterogenitas data
- Inkonsistensi dan kesalahan
- Latensi kedatangan data
- Masalah lokalisasi dan pemformatan data

- Kualitas data yang buruk

Jika tantangan ini tidak diatasi, efektivitas dan keandalan pipeline akan terganggu. Untuk mengatasi keterbatasan ini, kemajuan terbaru dalam arsitektur Big Data menekankan penggunaan pola desain – solusi yang dapat digunakan kembali untuk masalah arsitektur umum. Pola-pola ini membantu memisahkan komponen pipeline, sehingga memudahkan untuk memasukkan, memodelkan, mentransformasi, dan mengirimkan data secara fleksibel. Pola-pola ini juga mendukung pencapaian atribut kualitas seperti kinerja, kegunaan, pemeliharaan, keamanan, dan keandalan. Yokoyama [6] mengusulkan arsitektur tiga lapis yang banyak diadopsi untuk mendesain pipeline Big Data di seluruh aplikasi perusahaan. Arsitektur ini memperkenalkan prinsip utama: memisahkan logika bisnis dari model ML, sehingga meningkatkan stabilitas operasional (lihat Gambar 3.7).

- **Pemisahan Vertikal:** Arsitektur ini memisahkan *lapisan logika bisnis* (misalnya, dasbor dan aturan pelaporan) dari *lapisan ML* (misalnya, mesin inferensi dan deteksi pola). Pemisahan ini memastikan bahwa pembaruan pada model ML tidak mengganggu logika bisnis dan sebaliknya. Hal ini juga mengurangi *ketergantungan data*, sumber umum utang teknis dalam sistem Big Data.
- **Pemisahan Horizontal:** Arsitektur ini membagi pipeline menjadi tiga lapisan:
 1. **Lapisan Presentasi:** antarmuka pengguna dan mekanisme pengumpulan data
 2. **Lapisan Logika:** aturan bisnis, mesin inferensi, dan modul pra-pemrosesan
 3. **Lapisan Data:** basis data, data mart, dan data lake untuk data mentah dan data yang telah diproses



Gambar 3.7 Desain arsitektur pipeline Big Data.

Pendekatan berlapis ini meningkatkan modularitas, menyederhanakan prosedur rollback, dan meningkatkan ketahanan sistem. Misalnya, jika subsistem ML memerlukan pelatihan ulang atau penggantian, subsistem tersebut dapat diperbarui secara independen tanpa memengaruhi logika bisnis. Terlepas dari keuntungannya, arsitektur ini menghadirkan tantangan:

- Model ML secara inheren tidak stabil karena ketergantungannya pada kualitas dan kesegaran data.
- Serangan adversarial dapat mengganggu integritas model.
- Tren dan pergeseran data dapat mengurangi efektivitas model dari waktu ke waktu.

Untuk mengatasi masalah ini, arsitektur memperkenalkan beberapa elemen modular dalam subsistem ML untuk:

1. Pengumpulan Data
2. Pemrosesan Data
3. Inferensi

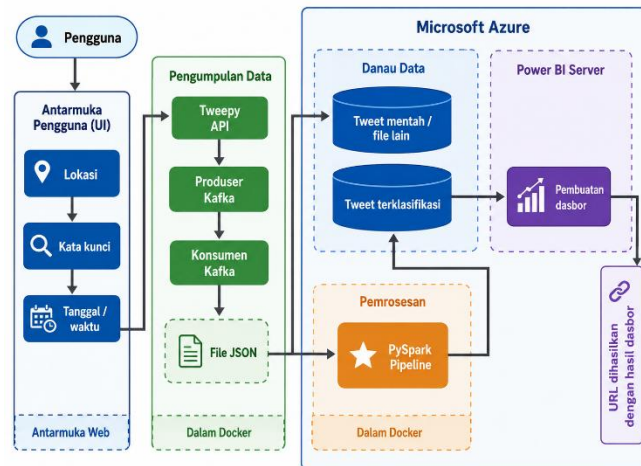
Modularisasi ini memastikan bahwa pengumpulan data independen dari antarmuka pengguna dan lapisan penyimpanan, sehingga meningkatkan fleksibilitas dan pemeliharaan. Singkatnya, strategi *Memisahkan Logika Bisnis dari Model ML* menghasilkan arsitektur terdistribusi dan terpisah yang mendukung skalabilitas, isolasi kesalahan, dan jaminan kualitas. Ini memberikan fondasi yang kuat untuk membangun sistem Big Data yang tangguh yang disesuaikan dengan kebutuhan bisnis yang terus berkembang.

3.7 KLASIFIKASI EMOSI DENGAN SPARK REAL-TIME

Untuk mendemonstrasikan penerapan strategi desain pipeline Big Data dan pola arsitektur, kami menyajikan studi kasus dunia nyata: *klasifikasi emosi dari feed Twitter*. Studi kasus ini mengilustrasikan bagaimana pipeline modular dan terukur dapat dibangun menggunakan teknologi open-source dan berbasis cloud untuk mendukung analitik real-time.

- **Gambaran Umum Kasus Penggunaan:** Dalam lanskap digital yang terhubung saat ini, pengguna sering berbagi opini dan emosi di platform media sosial. Twitter (sekarang dikenal sebagai X) adalah sumber konten real-time yang kaya dan sarat emosi. Mendeteksi dan mengklasifikasikan emosi dalam tweet dapat memberikan wawasan berharga tentang sentimen publik pada topik seperti politik, hiburan, kesehatan, dan produk konsumen.
- **Arsitektur Pipeline:** Pipeline Spark yang diusulkan dirancang untuk mengumpulkan, memproses, dan mengklasifikasikan tweet secara real-time. Pipeline ini mengintegrasikan komponen-komponen seperti:
 - *Tweepy* untuk mengakses API Twitter
 - *Apache Kafka* untuk antrian dan penyiaran pesan
 - *MongoDB* untuk menyimpan data JSON mentah
 - *Apache Spark* untuk pemrosesan dan klasifikasi terdistribusi
 - *Docker* untuk kontainerisasi
 - *Azure Data Lake* untuk penyimpanan berbasis cloud

- *Power BI/Tableau/Qlik* untuk visualisasi



Gambar 3.8 Alur kerja Spark untuk klasifikasi emosi.

- **Detail Implementasi:** Pipeline beroperasi sebagai berikut :
 1. Pengguna mengakses antarmuka front-end yang dihosting di *Azure*, memasukkan kata kunci untuk memulai analisis.
 2. Aplikasi memicu *API Tweepy*, yang mulai mengumpulkan tweet yang sesuai dengan kriteria yang ditentukan.
 3. Topik *Kafka* dibuat di dalam kontainer *Docker*, dan tweet dialirkan secara real-time.
 4. Konsumen *Kafka* mengirimkan setiap tweet (sebagai objek JSON) ke server *MongoDB* lokal dan *Azure Data Lake* yang ditunjuk.
 5. Pipeline *Spark*, yang juga dikontainerisasi, memproses tweet menggunakan pustaka NLP untuk mengklasifikasikan emosi (kejutan, kegembiraan, ketakutan, kesedihan).
 6. Data yang diproses disimpan dalam partisi terpisah dari *Azure Data Lake* dan tersedia untuk visualisasi dan pelaporan.

Arsitektur ini mendukung penyebaran terdistribusi, skalabilitas, dan modularitas, sehingga cocok untuk lingkungan produksi dan dapat diperluas ke kasus penggunaan lain seperti deteksi berita palsu atau analisis sentimen berbasis gambar.

- **Evaluasi Spark Pipeline Berdasarkan Bahasa Pola:** Arsitektur pipeline Spark selaras dengan beberapa pola desain ML yang diidentifikasi oleh Washizaki dan kawan-kawan. Pola desain sering dikenal sebagai solusi umum dan dapat digunakan kembali untuk masalah yang sering terjadi. Pola desain tidak mewakili produk jadi yang dapat langsung diubah menjadi produksi (kode, sistem yang berfungsi, dan lain-lain.). Sebaliknya, pola desain adalah deskripsi atau templat yang menjelaskan cara memecahkan masalah.

Pola desain dapat mempercepat proses pengembangan. Pola desain menyediakan paradigma pengembangan yang telah diuji dan terbukti. Membangun sistem yang efektif membutuhkan pertimbangan masalah yang mungkin tidak terlihat

sampai nanti dalam implementasi. Ketika suatu sistem dirancang dari awal tanpa mengikuti templat apa pun, hal itu dapat mengakibatkan masalah halus tersembunyi yang membutuhkan waktu untuk dideteksi. Sistem yang dirancang dengan buruk seperti itu dapat mengakibatkan masalah besar di kemudian hari. Penggunaan pola desain dapat membantu mencegah masalah-masalah halus tersebut dan juga meningkatkan kemampuan penggunaan kembali sistem.

Bahasa pola berasal dari kombinasi berbagai pola yang bersama-sama memberikan solusi untuk masalah-masalah kompleks. Bahasa pola dapat dianggap sebagai cara untuk mengekspresikan solusi kompleks yang diperoleh dari pengalaman. Bahasa pola dapat membantu perancang sistem untuk mendapatkan pemahaman yang lebih baik tentang solusi tersebut.

- **Pola Desain Utama yang Diterapkan:** Pola ML yang diterapkan meliputi:
 1. **Penyeimbangan Beban Kerja:** Pipeline *Spark* memanfaatkan elemen arsitektur *Lambda*. Tweet dari konsumen *Kafka* diubah menjadi file JSON dan dikirimkan ke data lake *Azure*. Pada saat yang sama, tweet ini diproses dalam batch mini di dalam pipeline *PySpark*, di mana data (teks) melalui ETL dan diklasifikasikan sesuai dengan emosi yang ditunjukkan di dalamnya. Baik *Azure* maupun *Spark* sangat skalabel, dengan *Spark* mampu mengalokasikan node tambahan jika perangkat keras komoditas yang digunakannya dapat mendukungnya.
 2. **Memisahkan Logika Bisnis dari Logika ML:** Pipeline *Spark* menerima kueri pengguna, memeriksa basis data yang ada sebelum meneruskan kueri ke mesin inferensi.
 3. **Arsitektur Perutean Gateway:** Pipeline dapat menerima satu atau lebih kueri sekaligus. Kueri paralel diimplementasikan menggunakan penyeimbang beban di dalam *Azure* untuk mengoptimalkan latensi.
 4. **Arsitektur Layanan Mikro:** Setiap komponen dari pipeline *Spark* terhubung secara longgar dan dapat diimplementasikan secara independen melalui kontainer *Docker*. Data lake di *Azure* dan *Power BI* juga bersifat modular dan dapat berfungsi secara independen.
 5. **Arsitektur Lambda:** Pipeline ini menggunakan aliran data real-time (yang memproses dan mengklasifikasikan tweet) sekaligus menyediakan penyimpanan persisten paralel di dalam partisi *Azure Data Lake* yang didedikasikan untuk data mentah. Jika terjadi latensi dalam pipeline real-time, tweet-tweet ini akan diantrekan dan dimasukkan ke dalam pipeline ketika sumber daya mampu menerimanya. Saat ini, seluruh proses klasifikasi mini batch tweet membutuhkan waktu kurang dari 60 detik. Ini dapat dengan mudah ditingkatkan skalanya saat diterapkan dalam produksi menggunakan perangkat keras komoditas yang lebih canggih (atau komputasi awan). Penyimpanan persisten menyediakan penyeimbangan beban untuk pipeline real-time (yang memproses data dalam mini batch) dan merupakan cerminan dari pola desain Just-In-Time.
 6. **Data Lake untuk ML dan Ekstraktor Multisumber:** Pipeline *Spark* sesuai dengan pola desain ini dengan memanfaatkan data lake multi-partisi di dalam *Azure*. Satu partisi data lake menyimpan semua tweet mentah (dan berpotensi data tidak terstruktur

lainnya seperti gambar, audio, dan video). Ini bertindak sebagai area penampungan untuk semua data mentah. Partisi sekunder mengumpulkan semua data yang telah diekstrak dan ditransformasikan.

7. **Modularisasi: Salah satu keunggulan inti dari arsitektur Apache Spark adalah modularitasnya.** Fungsionalitas tambahan dapat dengan mudah dibangun dan dilampirkan ke pipeline yang ada untuk menghasilkan kategori prediksi lainnya. Misalnya, pipeline sekunder dapat dilampirkan ke data lake yang menyimpan data yang diproses untuk mengklasifikasikan tweet lebih lanjut sebagai palsu atau asli. Pipeline sekunder dan paralel juga dapat dibuat untuk mengekstrak gambar dari tweet dan mengklasifikasikan objek dan sentimen berdasarkan ekspresi wajah.
8. **Mengkapsulasi Model ML dalam Perlindungan Berbasis Aturan:** Meskipun pipeline menggunakan modul NLP yang terverifikasi dan banyak digunakan, validasi data berkala dipertimbangkan untuk memastikan bahwa model tidak memerlukan penyetelan hiperparameter tambahan. Model ML dirancang sedemikian rupa sehingga kuat dan tidak dipengaruhi oleh serangan adversarial.
9. **Pola Transformasi Bertahap:** Data lake menyediakan penyimpanan persisten. Jika pengguna menjalankan kueri untuk tweet masa lalu (dalam rentang tanggal tertentu), aplikasi akan terlebih dahulu mengambil tweet yang sesuai dari data lake. Jika tweet yang diinginkan tidak ada, aplikasi akan memulai aliran baru untuk mengambil data dari Twitter.
10. **Pola NoSQL:** Tweet disimpan dalam format JSON, memastikan bahwa data dapat dengan cepat dikueri berdasarkan kunci yang telah ditentukan sebelumnya.
11. **Pola Stateless Ringan:** Pipeline dapat diakses melalui API berbasis web yang diimplementasikan di *Azure*, meningkatkan skalabilitas dan toleransi kesalahan.
12. **Pola Hampir Real-Time:** Pipeline saat ini memiliki latensi 4 ms dalam pengambilan data, dan sekitar 1 menit waktu tunggu dalam mengklasifikasikan tweet.

3.8 RINGKASAN

Dalam bab ini, kita telah mengeksplorasi arsitektur pipeline Big Data, berdasarkan konsep dasar yang diperkenalkan dalam Bab 1 dan 2. Kita mulai dengan mendefinisikan pipeline data dan memeriksa komponen-komponen utamanya, termasuk pengumpulan data, penyerapan, transformasi, komputasi, dan presentasi. Komponen-komponen ini bekerja sama untuk mengotomatiskan proses ETL dan memungkinkan analitik waktu nyata yang terukur. Kami memperkenalkan arsitektur Lambda, model hibrida yang mengintegrasikan pemrosesan batch dan stream untuk mendukung sistem data yang toleran terhadap kesalahan dan memiliki latensi rendah. Struktur berlapisnya – yang terdiri dari lapisan batch, serving, dan speed – menawarkan fleksibilitas dan ketahanan tetapi juga memperkenalkan kompleksitas implementasi.

Untuk mengilustrasikan aplikasi praktis, kami meneliti arsitektur pipeline Twitter, Pinterest, dan Airbnb, menyoroti bagaimana setiap organisasi menyesuaikan infrastrukturnya untuk memenuhi kebutuhan pemrosesan data tertentu. Contoh-contoh ini menunjukkan

penggunaan teknologi open-source dan berbasis cloud untuk menangani volume data yang sangat besar dengan throughput dan keandalan yang tinggi. Kemudian, kami membahas strategi desain untuk membangun pipeline Big Data yang disesuaikan, menekankan pentingnya modularitas, skalabilitas, dan pemeliharaan. Bab ini memperkenalkan pola arsitektur Membedakan Logika Bisnis dari Model ML dan manfaat pemisahan lapisan sistem untuk meningkatkan stabilitas dan adaptabilitas. Terakhir, kami mengevaluasi pipeline Spark real-time untuk klasifikasi emosi dalam feed Twitter, menerapkan berbagai pola desain ML. Pola-pola ini – seperti penyeimbangan beban kerja, microservices, modularisasi, dan pengamanan berbasis aturan – meningkatkan kinerja, keamanan, dan ekstensibilitas pipeline. Bersama-sama, konsep dan contoh ini menyediakan kerangka kerja komprehensif untuk merancang sistem Big Data terdistribusi yang tangguh yang dapat mendukung pengambilan keputusan cerdas di berbagai domain.

3.9 PERTANYAAN

1. Jelaskan sebuah data pipeline dan sebutkan lima komponen utamanya, berikan satu atau dua contoh teknologi untuk masing-masing komponen.
2. Ringkas arsitektur Lambda, termasuk tiga lapisannya, dan diskusikan dua keuntungan dan dua kerugiannya.
3. Bandingkan arsitektur data pipeline yang digunakan oleh Twitter, Pinterest, dan Airbnb, dan identifikasi fitur-fitur khas masing-masing.
4. Jelaskan mengapa penting untuk membedakan logika bisnis dari model pembelajaran mesin. Berikan satu manfaat dan satu risiko yang dikurangi oleh perbedaan ini.
5. Lapisan mana yang bertanggung jawab untuk mengindeks tampilan batch untuk mendukung kueri ad-hoc?
 - A. Batch
 - B. Serving
 - C. Speed
 - D. Data Lake
6. Manakah dari pasangan berikut yang benar?
 - A. Kafka → mesin kolom
 - B. Presto → SQL ad-hoc melalui Hive/S3
 - C. Airflow → broker pesan
 - D. BigQuery → runtime kontainer

3.10 TUGAS PEMROGRAMAN

1. Tulis fungsi bernama `clean` yang menerima string teks dan melakukan operasi berikut
 1. Mengubah teks menjadi huruf kecil,
 2. Menghapus URL dan nama pengguna berawalan `@`,
 3. Menghapus karakter non-alfanumerik dengan tetap mempertahankan spasi, dan
 4. Menggabungkan beberapa spasi berurutan menjadi satu spasi.

2. Diberikan daftar kamus, masing-masing berisi kunci “id” dan “teks”, lakukan langkah-langkah berikut
 1. Bersihkan setiap teks menggunakan fungsi dari Tugas 1,
 2. Tandai setiap teks yang telah dibersihkan dengan satu emosi berdasarkan pencocokan kata kunci (kemungkinan emosi: kejutan, kegembiraan, ketakutan, dan kesedihan; jika tidak, tandai sebagai tidak diketahui), dan
 3. Cetak hitungan setiap emosi. (*Catatan: Jangan gunakan file, kelas, atau jendela.*)

BAB 4

TEKNOLOGI BIG DATA DAN KUALITAS DATA

4.1 SISTEM BIG DATA

Pada Bab 2, kita telah memperkenalkan sistem pemrosesan batch dan stream sebagai komponen dasar arsitektur Big Data. Bab-bab sebelumnya telah memperkenalkan konsep kerangka kerja Big Data dan teknologi AI, sehingga menjadi dasar untuk memahami blok pembangun dan proses pengumpulan datanya. Pada bab ini, kita akan memperluas fondasi tersebut dengan mengeksplorasi mekanisme pemrosesan batch, stream, dan hybrid, yang masing-masing penting bagi analitik Big Data tingkat lanjut. Arsitektur Big Data modern dirancang untuk menyimpan, memproses, dan menganalisis data dalam skala besar, dan tiga paradigma pemrosesan – batch, stream, dan hybrid – membentuk inti dari setiap stack Big Data yang tangguh.

- **Pemrosesan Batch** ideal ketika seluruh dataset tersedia dan perlu diproses secara holistik. Misalnya, menghitung total atau rata-rata di seluruh dataset memerlukan pemeliharaan status selama komputasi. Meskipun pemrosesan batch sangat ampuh untuk analisis komprehensif, ia memiliki kelemahan – terutama, waktu pemrosesan yang lama, sehingga tidak cocok untuk aplikasi yang sensitif terhadap waktu.
- **Pemrosesan Stream menangani** aliran data real-time yang berkelanjutan. Tidak seperti sistem batch, pemrosesan stream menganalisis data saat dihasilkan atau diterima, memungkinkan wawasan dan respons langsung. Ini penting untuk aplikasi seperti deteksi penipuan, analitik real-time, dan pemantauan IoT. Karakteristik utama meliputi:
 - Respons real-time
 - Arsitektur berbasis peristiwa
 - Skalabilitas
 - Toleransi kesalahan
- **Sistem Pemrosesan Hibrida** menggabungkan kekuatan pemrosesan batch dan stream. Sistem ini menawarkan respons latensi rendah untuk kebutuhan real-time sambil mendukung analisis offline komprehensif dari kumpulan data besar. Contoh umum adalah arsitektur Lambda (diperkenalkan di Bab 3), yang memisahkan pemrosesan data real-time dan historis ke dalam lapisan "kecepatan" dan "batch" yang berbeda.

Pada bagian berikut, kita akan memeriksa satu kerangka kerja representatif dari setiap kategori – *Hadoop* untuk pemrosesan batch, *Apache Storm* untuk pemrosesan stream, dan *Apache Spark* untuk pemrosesan hibrida – untuk memahami peran mereka dalam analitik data yang terukur.

4.2 SISTEM BIG DATA HADOOP

Sistem Big Data Hadoop adalah kerangka kerja dasar untuk pemrosesan batch di lingkungan terdistribusi. Sistem ini terdiri dari beberapa komponen terintegrasi yang bekerja

sama untuk menyimpan dan memproses kumpulan data besar secara efisien. Inti dari Hadoop adalah *Hadoop Distributed File System* (HDFS), yang menyediakan penyimpanan yang toleran terhadap kesalahan dan dapat diskalakan di seluruh kluster node. HDFS memastikan ketersediaan data bahkan jika terjadi kegagalan node dengan mereplikasi data di beberapa mesin. HDFS berfungsi sebagai sumber dan tujuan untuk hasil pemrosesan perantara dan akhir. Apache Hadoop adalah kerangka kerja yang banyak digunakan untuk penyimpanan dan pemrosesan terdistribusi kumpulan data besar. Arsitekturnya mengikuti paradigma master-slave, yang terdiri dari dua lapisan utama: Lapisan HDFS dan Lapisan MapReduce.

Inti dari Lapisan HDFS adalah *NameNode*, yang bertindak sebagai master dan mengelola namespace sistem file, mengontrol akses ke file dan direktori, dan memelihara metadata untuk semua blok data yang disimpan di seluruh kluster. Penyimpanan data aktual ditangani oleh beberapa *DataNode*, yang berfungsi sebagai slave. Setiap *DataNode* menyimpan blok data dan melaporkan statusnya ke *NameNode*. Lapisan MapReduce bertanggung jawab untuk memproses data. *JobTracker* berfungsi sebagai master, mengatur eksekusi pekerjaan MapReduce dengan menetapkan tugas ke sumber daya yang tersedia dan memantau kemajuannya. Eksekusi aktual tugas *map* dan *reduce* terjadi pada *TaskTracker*, yang berjalan pada node slave.

Setiap *TaskTracker* berkomunikasi dengan *JobTracker*, melaporkan statusnya dan kemajuan tugas yang ditugaskan. Arsitektur ini memastikan ketersediaan tinggi, toleransi kesalahan, dan skalabilitas untuk menangani volume data yang sangat besar. Hadoop menggunakan YARN (*Yet Another Resource Negotiator*) untuk mengelola sumber daya kluster dan menjadwalkan pekerjaan. YARN memungkinkan berbagai beban kerja untuk berjalan secara bersamaan pada kluster yang sama, meningkatkan pemanfaatan sumber daya. Untuk pemrosesan batch, Hadoop menggunakan mesin MapReduce, yang mengikuti model tiga fase:

1. **Fase Map:** Dataset dibaca dari HDFS, dibagi menjadi beberapa bagian, dan didistribusikan ke seluruh node. Setiap node menerapkan komputasi ke subset data yang ditugaskan kepadanya.
2. **Fase Pengacakan (Shuffle):** Hasil sementara dikelompokkan berdasarkan kunci dan didistribusikan kembali ke seluruh node.
3. **Fase Pengurangan (Reduce):** Setiap node menggabungkan dan meringkas data yang dikelompokkan, menghasilkan hasil akhir yang ditulis kembali ke HDFS.

Meskipun Hadoop sangat toleran terhadap kesalahan dan hemat biaya, ketergantungannya pada operasi berbasis disk menimbulkan latensi. Beberapa siklus baca/tulis membuatnya lebih lambat dibandingkan dengan kerangka kerja dalam memori seperti Spark. Selain itu, MapReduce memiliki kurva pembelajaran yang curam, meskipun skalabilitas dan integrasinya dengan alat lain dalam ekosistem Hadoop tetap menjadi keunggulan yang kuat.

Hadoop terus menjadi solusi yang andal untuk *beban kerja berorientasi batch*, terutama ketika biaya dan toleransi kesalahan menjadi prioritas. Namun, untuk analitik interaktif atau waktu nyata, kerangka kerja yang lebih baru seperti *Apache Spark* menawarkan peningkatan kinerja yang signifikan.

4.3 KERANGKA KERJA PEMROSESAN ALIRAN

Sistem pemrosesan aliran dirancang untuk menganalisis data yang bergerak, melakukan komputasi pada data saat data tersebut masuk ke sistem. Tidak seperti pemrosesan batch, yang beroperasi pada seluruh kumpulan data setelah pengumpulan, pemrosesan aliran menerapkan operasi pada setiap item data individual secara real-time. Sistem ini dioptimalkan untuk lingkungan dengan latensi rendah dan throughput tinggi di mana wawasan langsung sangat penting. Dalam pemrosesan aliran, kumpulan data tidak terbatas, artinya kumpulan data terus bertambah seiring dengan masuknya data baru. Konsep kumpulan data kerja mengacu pada item yang sedang diproses saat ini, bukan seluruh riwayat data. Hasil dihasilkan dan diperbarui secara terus menerus, memungkinkan respons dinamis terhadap perubahan kondisi. Kerangka kerja pemrosesan aliran terbagi dalam dua kategori:

- **Pemrosesan Aliran Sejati (True Stream Processing):** Memproses satu item dalam satu waktu.
- **Pemrosesan Mikro-Batch (Micro-Batch Processing):** Memproses sejumlah kecil item dalam interval waktu yang singkat.

Pemrosesan stream sangat ideal untuk aplikasi seperti:

- Deteksi penipuan secara real-time
- Pemantauan perangkat IoT
- Analisis media sosial
- Mesin penetapan harga dan rekomendasi dinamis

Apache Storm adalah sistem komputasi real-time terdistribusi yang dirancang untuk memproses stream data. Mirip dengan Hadoop, Storm beroperasi pada model master-slave tetapi dioptimalkan untuk pemrosesan throughput tinggi dan latensi rendah. Daemon Nimbus berfungsi sebagai node master, bertanggung jawab untuk mendistribusikan kode di seluruh cluster, menetapkan tugas ke node pekerja, dan memantau kesehatannya. Zookeeper memainkan peran penting dalam koordinasi cluster, mengelola konfigurasi, mempertahankan status cluster, dan memfasilitasi komunikasi antara Nimbus dan node pekerja. Di sisi pekerja, daemon Supervisor berjalan di setiap node pekerja. Supervisor mendengarkan pekerjaan yang ditugaskan oleh Nimbus dan memulai atau menghentikan proses Pekerja sesuai arahan.

Setiap proses Pekerja, pada gilirannya, mengeksekusi sebagian dari topologi Storm, melakukan komputasi real-time yang sebenarnya. *Pengaturan terdistribusi ini* memungkinkan Storm untuk memproses aliran data secara *terus menerus dan toleran* terhadap kesalahan, sehingga cocok untuk aplikasi yang membutuhkan analisis dan respons data segera. Apache Storm adalah kerangka kerja pemrosesan aliran berkinerja tinggi yang dikenal karena latensinya yang sangat rendah. Ini sangat cocok untuk beban kerja di mana waktu pemrosesan secara langsung memengaruhi pengalaman pengguna. Storm mengatur tugas pemrosesan menggunakan grafik asiklik terarah (DAG) yang disebut topologi, yang mendefinisikan aliran data melalui sistem. Sebuah topologi terdiri dari tiga komponen inti:

- **Aliran:** Urutan tupel data yang tidak terbatas.

- **Spout:** Sumber data yang memancarkan aliran ke dalam topologi (misalnya, API dan antrian).
- **Bolt:** Unit pemrosesan yang mengonsumsi aliran, menerapkan operasi, dan memancarkan aliran baru.

Bolt terhubung ke spout dan bolt lain untuk membentuk pipeline transformasi. Keluaran akhir topologi dapat diteruskan ke sistem eksternal untuk pemrosesan atau penyimpanan lebih lanjut. Storm menyediakan jaminan *pemrosesan setidaknya sekali*, artinya setiap pesan diproses setidaknya sekali. Namun, dalam skenario kegagalan, pesan dapat diproses beberapa kali, dan urutan pesan tidak dijamin. Untuk mengatasi keterbatasan ini, Storm menawarkan abstraksi yang disebut *Trident*, yang memungkinkan:

- Pemrosesan tepat sekali
- Komputasi stateful
- Pemrosesan mikro-batch

Meskipun Trident memperkenalkan latensi tambahan, ia meningkatkan keandalan dan toleransi kesalahan dengan menangani pesan duplikat secara cerdas. Storm terintegrasi dengan baik dengan ekosistem Hadoop, termasuk YARN untuk manajemen sumber daya. Ia juga mendukung berbagai bahasa pemrograman, menawarkan fleksibilitas dalam mendefinisikan topologi. Ini adalah pilihan yang ampuh untuk analitik mendekati waktu nyata, terutama ketika latensi dan responsivitas sangat penting. Desain modular dan dukungan bahasa yang luas menjadikannya alat serbaguna dalam pipeline Big Data modern.

4.4 SISTEM PEMROSESAN HIBRIDA

Sistem pemrosesan hibrida dirancang untuk menangani beban kerja batch dan stream, menawarkan solusi terpadu untuk beragam kebutuhan pemrosesan data. Kerangka kerja seperti Apache Spark dan Apache Flink mencontohkan pendekatan ini, menyediakan arsitektur fleksibel yang mendukung analitik waktu nyata, pembelajaran mesin, pemrosesan grafik, dan kueri interaktif. Sistem ini bertujuan untuk menggabungkan kelengkapan pemrosesan batch dengan responsivitas latensi rendah dari pemrosesan stream, menjadikannya ideal untuk aplikasi Big Data modern yang membutuhkan wawasan historis dan waktu nyata. Spark berfokus pada percepatan beban kerja pemrosesan batch dengan menawarkan komputasi dalam memori penuh dan optimasi pemrosesan. Flink, di sisi lain, adalah kerangka kerja pemrosesan stream yang dapat menangani tugas batch. Batch aliran data dengan batasan terbatas diproses oleh Flink sebagai subset dari pemrosesan stream.

Apache Spark: Apache Spark adalah mesin pemrosesan generasi berikutnya yang meningkatkan model MapReduce Hadoop dengan menawarkan *komputasi dalam memori*, secara signifikan mempercepat beban kerja batch. Tidak seperti MapReduce, Spark memuat data ke dalam memori sekali dan melakukan semua komputasi perantara tanpa menulis ke disk. Ini mengurangi overhead I/O dan meningkatkan kinerja. Spark menggunakan DAG untuk merepresentasikan urutan operasi dan ketergantungannya, memungkinkan penjadwalan dan optimasi yang cerdas. Abstraksi inti Spark adalah *resilient distributed dataset* (RDD) – kumpulan objek yang tidak dapat diubah dan toleran terhadap kesalahan yang didistribusikan

di seluruh kluster. Operasi pada RDD menghasilkan RDD baru, dan silsilahnya dapat dilacak kembali ke sumber data asli, memungkinkan pemulihan tanpa penulisan disk berulang.

Untuk pemrosesan stream, Spark menggunakan *strategi micro-batching*, di mana data yang masuk di-buffer dalam interval kurang dari satu detik dan diproses sebagai batch kecil. Pendekatan ini memungkinkan Spark untuk menggunakan kembali semantik mesin batch-nya untuk beban kerja stream. Sebelumnya di Bab 3, ketika kita membahas arsitektur Lambda, kita menyoroti fitur utama arsitektur ini. Kemampuan pemrosesan stream dalam arsitektur Lambda didukung oleh Spark streaming menggunakan strategi *micro-batching*. Spark menawarkan banyak keunggulan:

- Komputasi dalam memori yang cepat
- Penjadwalan berbasis DAG tingkat lanjut
- Ekosistem pustaka yang kaya (misalnya, MLlib, GraphX, dan Spark SQL)
- Model pemrograman yang lebih mudah dibandingkan dengan MapReduce
- Dapat dijalankan sebagai kluster mandiri atau diintegrasikan dengan Hadoop

Namun, Spark memiliki beberapa keterbatasan:

- Kebutuhan memori yang lebih tinggi (RAM lebih mahal daripada disk)
- Perebutan sumber daya di lingkungan bersama
- Mungkin kurang efisien untuk pemrosesan aliran data sebenarnya dibandingkan dengan Flink

Apache Spark adalah sistem komputasi kluster serbaguna terdistribusi sumber terbuka yang kuat. Spark dikelilingi oleh berbagai teknologi lain, yang menunjukkan keserbagunaannya dan kemampuannya untuk berinteraksi dengan berbagai sumber data, basis data, dan alat big data. Kekuatan Spark terletak pada mesin terpadunya untuk pemrosesan data skala besar, yang mencakup modul untuk pemrosesan batch, pemrosesan aliran data, kueri SQL, pembelajaran mesin (MLlib), dan pemrosesan grafik (GraphX). Spark dapat terhubung dengan mudah dengan:

- **Basis Data NoSQL:** Cassandra, MongoDB, Apache HBase, Redis.
- **Basis Data Relasional:** IBM DB2, MySQL, PostgreSQL.
- **Sistem File Terdistribusi:** HDFS.
- **Sistem Pesan:** Apache Kafka.
- **Mesin Pencarian dan Analitik:** Elastic.

Integrasi yang luas ini memungkinkan Spark untuk berfungsi sebagai mesin pemrosesan pusat dalam berbagai arsitektur big data, memungkinkan pengguna untuk memanfaatkan kinerja tinggi dan kemudahan penggunaannya di berbagai kebutuhan penyimpanan dan pemrosesan data.

4.5 APACHE FLINK DAN APACHE SAMZA

Pada bagian ini, kita akan membandingkan dua kerangka kerja pemrosesan stream yang terkemuka: *Apache Flink* dan *Apache Samza*. Meskipun keduanya dirancang untuk pemrosesan data real-time, keduanya berbeda secara signifikan dalam arsitektur, kemampuan, dan strategi integrasi. *Apache Flink* adalah kerangka kerja yang mengutamakan

stream dan juga mendukung beban kerja batch. Flink mengikuti *arsitektur Kappa*, memperlakukan semua data sebagai aliran data. Flink memproses data satu per satu, menawarkan semantik aliran data yang sebenarnya dan latensi yang sangat rendah. API *DataStream Flink* menyediakan abstraksi untuk stream, operator, sumber, dan sink. Untuk tugas batch, Flink mengoptimalkan eksekusi dengan menghilangkan snapshotting dan meminimalkan penggunaan sumber daya. Untuk pemrosesan batch, Flink memperlakukan dataset terbatas sebagai kasus khusus dari streaming. Flink membaca data dari penyimpanan persisten dan menerapkan optimasi seperti:

- Menghilangkan snapshotting untuk eksekusi yang lebih cepat
- Memecah tugas menjadi beberapa tahapan untuk meminimalkan penggunaan sumber daya.

Flink menawarkan keunggulan berikut:

- Latensi rendah dan throughput tinggi
- Pemrosesan stream sejati dengan komputasi entri demi entri
- Manajemen memori otomatis (menghindari pengumpulan sampah Java)
- Lokalisasi dan caching data yang cerdas
- Ekosistem yang kaya: kueri SQL, pemrosesan grafik, dan pembelajaran mesin
- Integrasi tanpa batas dengan tumpukan Hadoop (YARN, HDFS, dan Kafka)
- Kompatibilitas dengan kerangka kerja lain (misalnya, Hadoop dan Storm).

Namun, ia memiliki beberapa keterbatasan:

- Masih merupakan proyek yang relatif baru
- Lebih sedikit penerapan produksi skala besar dibandingkan dengan Spark

Apache Samza terintegrasi erat dengan *Apache Kafka*, memanfaatkan arsitektur Kafka untuk toleransi kesalahan, buffering, dan penyimpanan status. Samza unggul dalam skenario yang membutuhkan *keterkaitan yang longgar* antara langkah-langkah pemrosesan dan mendukung berbagi data berbasis topik yang terukur di seluruh tim. Meskipun Samza menawarkan abstraksi tingkat tinggi dan integrasi Kafka yang kuat, saat ini hanya mendukung bahasa JVM, yang membatasi fleksibilitas dibandingkan dengan Storm atau Flink.

Setelah mengeksplorasi fondasi arsitektur dan paradigma pemrosesan sistem Big Data – batch, stream, dan hybrid – sekarang kita beralih ke aspek penting dari pipeline data: *kualitas data*. Terlepas dari kecanggihan kerangka kerja pemrosesan, nilai analitik bergantung pada integritas data yang mendasarinya. Data mentah yang dikumpulkan dari berbagai sumber seringkali tidak lengkap, tidak konsisten, atau salah. Tanpa pembersihan yang tepat, masalah ini menyebar melalui pipeline, menyebabkan wawasan yang tidak dapat diandalkan dan pengambilan keputusan yang salah. Bagian selanjutnya memperkenalkan *pembersihan data*, perannya dalam alur kerja analitik, dan tantangan yang diatasinya baik dalam lingkungan sumber tunggal maupun multi-sumber.

4.6 PENDAHULUAN TENTANG PEMBERSIHAN DATA

Pada Bab 2, kita telah membahas secara singkat *proses pengumpulan data*. Bab ini sekarang berfokus pada fase penting berikutnya dari alur kerja analitik: *Ekstraksi*,

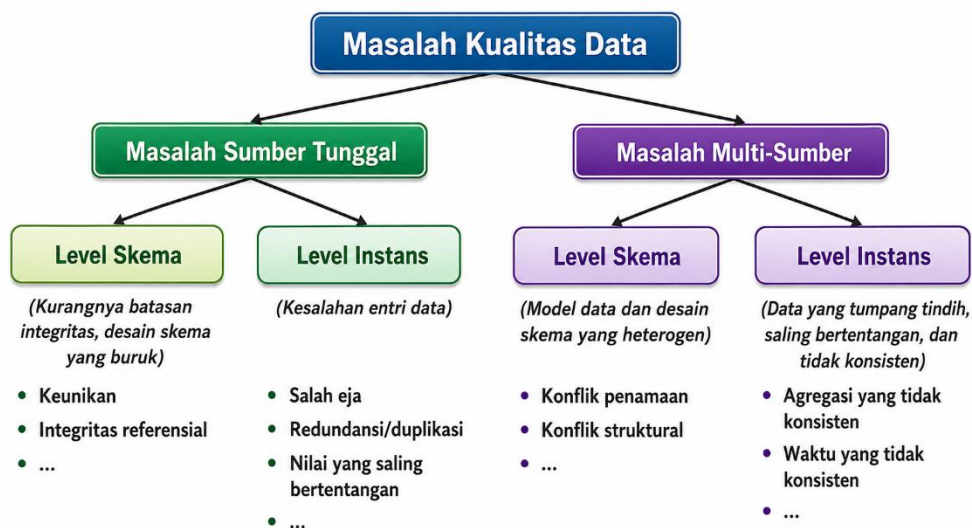
Pembersihan, dan *Anotasi*, yang umumnya disebut sebagai *pembersihan data*. Data yang dikumpulkan dari sumber heterogen – seperti rekam medis elektronik, sensor IoT, dan umpan media sosial – jarang siap untuk dianalisis. Data tersebut sering kali mengandung ketidakakuratan, inkonsistensi, dan nilai yang hilang. Misalnya, rekam medis rumah sakit dapat mencakup catatan dokter yang ditranskripsikan, data sensor terstruktur, dan file gambar, masing-masing dengan format dan profil kesalahan yang berbeda.

Apa Itu Pembersihan Data: Pembersihan data (atau *pembersihan data*) melibatkan pendeteksian dan koreksi kesalahan dan inkonsistensi untuk meningkatkan kualitas data. Masalah biasanya muncul selama pengumpulan data karena kesalahan ejaan, informasi yang hilang, atau entri yang tidak valid. Masalah juga terjadi ketika mengintegrasikan beberapa sistem – seperti basis data terfederasi atau sumber berbasis web – ke dalam satu gudang data. Masalah kualitas data dapat diklasifikasikan sebagai :

- Masalah sumber tunggal vs. masalah sumber ganda
- Masalah tingkat skema vs. masalah tingkat instansi

Masalah tingkat skema berkaitan dengan desain skema yang buruk, kurangnya batasan integritas, atau model data yang tidak kompatibel. Masalah tingkat instansi melibatkan kesalahan dalam nilai data aktual, seperti duplikat, kontradiksi, atau format yang salah.

Gambar 4.1 menjelaskan klasifikasi masalah kualitas data dalam sumber data. Dalam dataset sumber tunggal, masalah tingkat skema sebagian besar berkaitan dengan kurangnya batasan integritas dan desain skema yang buruk. Di sisi lain, masalah tingkat instansi sebagian besar berkaitan dengan kesalahan entri data. Dalam dataset sumber ganda, masalah tingkat skema sebagian besar berkaitan dengan model data dan desain skema yang heterogen, dan masalah tingkat instansi berkaitan dengan tumpang tindih, kontradiksi, dan data yang tidak konsisten (lihat Gambar 4.1).



Gambar 4.1 Klasifikasi masalah pembersihan data dalam sumber data.

Terdapat beberapa pendekatan untuk pembersihan data :

- **Analisis Data:** Analisis data dapat membantu mengidentifikasi kesalahan dan inkonsistensi dalam sumber data. Ini mencakup inspeksi manual sampel data serta pemeriksaan metadata untuk mendapatkan wawasan tentang properti data dan untuk mendeteksi masalah kualitas data.
- **Alur Kerja Transformasi dan Aturan Pemetaan:** Untuk memperhitungkan tingkat heterogenitas dalam jumlah sumber data yang berbeda dan untuk membersihkan inkonsistensi dalam dataset, diperlukan alur kerja transformasi strategis dan aturan pemetaan yang mapan. Pendekatan tersebut harus mencakup sejumlah besar langkah transformasi dan pembersihan data yang dijalankan untuk memetakan setiap sumber ke model data umum untuk setiap gudang data yang diberikan.
- **Verifikasi:** Pengujian dan evaluasi dilakukan untuk memastikan bahwa alur kerja transformasi dan aturan pemetaan telah diimplementasikan secara korektif dan efektif pada sampel, atau pada salinan sumber data. Langkah ini juga membantu dalam meningkatkan definisi transformasi.
- **Transformasi:** Langkah transformasi akan dibahas nanti dalam bab ini ketika kita membahas alur kerja *ETL (Ekstraksi, Transformasi, dan Pemuatan)*. Transformasi dilakukan setiap kali ada kebutuhan untuk memuat dan menyegarkan gudang data atau saat menjawab kueri dari berbagai sumber.
- **Alur Balik Data yang Dibersihkan:** Setelah kesalahan dan data kotor dihilangkan dan data dibersihkan, alur balik data yang dibersihkan dilakukan untuk menjaga sumber asli dalam kondisi baik. Langkah ini memastikan bahwa aplikasi lama ditingkatkan dan menghindari pengulangan pekerjaan pembersihan untuk ekstraksi data di masa mendatang.

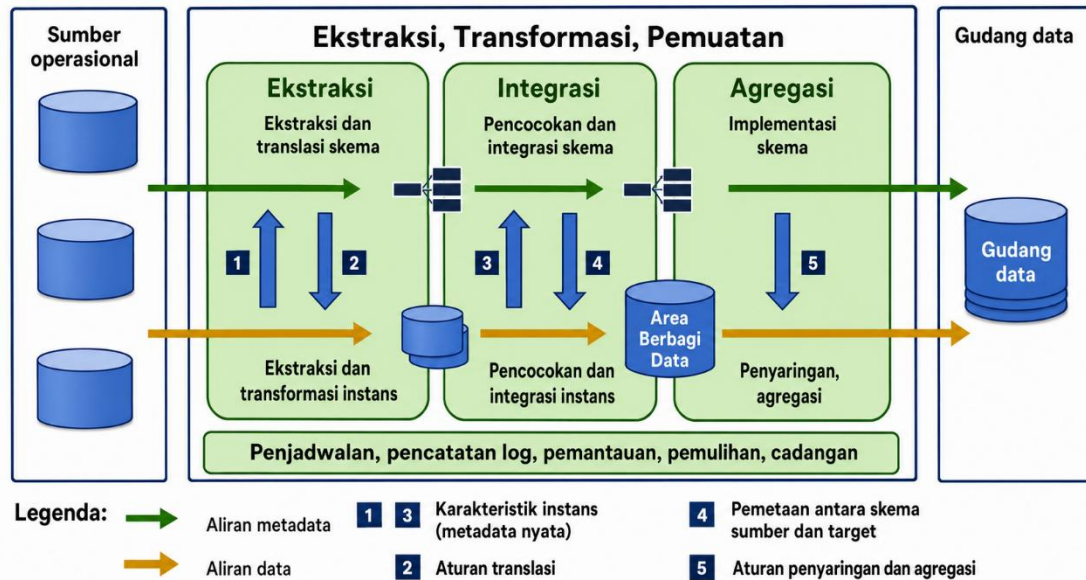
Pendekatan pembersihan data yang efektif harus :

- Mendeteksi dan menghilangkan kesalahan utama di seluruh sumber tunggal dan ganda.
- Meminimalkan intervensi manual melalui otomatisasi.
- Mengintegrasikan transformasi tingkat skema dengan pembersihan tingkat instance.
- Menggunakan fungsi pemetaan deklaratif dan dapat digunakan kembali.
- Mendukung infrastruktur alur kerja untuk lingkungan multi-sumber skala besar.

Pada bagian selanjutnya, kita akan mengeksplorasi proses ETL, yang mengkonsolidasikan data dari berbagai sumber ke dalam gudang data terpadu.

4.7 EKSTRAKSI, TRANSFORMASI, DAN PEMUATAN

Untuk membangun sistem Big Data yang kompetitif, organisasi harus mengintegrasikan data dari berbagai sumber terfederasi ke dalam gudang data terpusat. Integrasi ini harus dilakukan secara efisien dan andal, terutama mengingat kemungkinan besar ditemukannya *data kotor* – informasi yang tidak konsisten, hilang, atau duplikat. Karena kemungkinan adanya berbagai macam inkonsistensi data dan volume data yang sangat besar untuk diproses di gudang data, pembersihan data dianggap sebagai salah satu masalah terbesar.



Gambar 4.2 Langkah-langkah pembersihan data dalam membangun gudang data: Proses ETL.

Proses ETL adalah pendekatan standar untuk mengkonsolidasikan dan membersihkan data sebelum masuk ke gudang data. Gambar 4.2 mengilustrasikan alur kerja ETL yang umum.

- **Fase Ekstraksi:** Data dikumpulkan dari berbagai sistem sumber, yang mungkin meliputi:
 - File datar (misalnya, CSV)
 - Dokumen XML
 - Basis data non-relasional (misalnya, IMS)
 - Sistem lama dengan format kepemilikan
 - Hasil web scraping atau screen scraping

Pada fase ini, *metadata skema* dan *instance data* diekstrak dan dipindahkan ke area penampungan sementara. Tujuannya adalah untuk mengubah format heterogen menjadi struktur terpadu yang sesuai untuk transformasi.

- **Fase Transformasi:** Transformasi melibatkan penerapan serangkaian langkah manipulasi data, seperti:
 - Memindahkan, memisahkan, menggabungkan, dan mengurutkan data
 - Mengkodekan nilai bentuk bebas
 - Menerjemahkan nilai yang dikodekan
 - Mengagregasi (misalnya, *roll-up*) dan mentransposisikan data

Skema dicocokkan dan diintegrasikan untuk membuat model terpadu untuk gudang data. Meskipun transformasi meningkatkan konsistensi struktural, sebagian besar tugas pembersihan data terjadi di area staging setelah transformasi dan sebelum pemuatan.

- **Fase Pemuatan:** Pada fase ini, skema yang telah ditransformasi dan instance yang telah dibersihkan dimuat ke dalam gudang data target. Frekuensi pemuatan bervariasi tergantung kebutuhan organisasi – harian, mingguan, atau bulanan.

Proses ETL membutuhkan banyak sumber daya. Penundaan komunikasi dan pemrosesan membuat pencapaian waktu respons yang dapat diterima menjadi sulit, dan upaya yang diperlukan untuk pembersihan selama ekstraksi dan integrasi semakin meningkatkan kompleksitas. Terlepas dari tantangan ini, ETL tetap sangat diperlukan untuk memastikan analitik yang andal. Pada bagian selanjutnya, kita akan membahas klasifikasi anomali data, yang menyoroti jenis masalah yang harus ditangani oleh alur kerja ETL.

4.8 KLASIFIKASI ANOMALI DATA

Data dunia nyata jarang sekali murni. Data tersebut seringkali mengandung berbagai anomali – kesalahan, inkonsistensi, dan ketidakberaturan – yang mengganggu kegunaannya untuk analisis dan pengambilan keputusan. Anomali ini meningkatkan biaya pra-pemrosesan dan mengurangi keandalan wawasan yang diperoleh dari data. Proses mengidentifikasi dan memperbaiki masalah ini dikenal sebagai pembersihan data. Meskipun otomatisasi dapat membantu dalam pembersihan data, keahlian domain sangat penting. Mendeteksi dan menyelesaikan anomali seringkali membutuhkan pengetahuan kontekstual, sehingga pembersihan data menjadi proses semi-otomatis. Mengingat volume data dalam sistem Big Data, otomatisasi sebanyak mungkin dari proses ini sangat diinginkan, meskipun otomatisasi penuh dibatasi oleh ketersediaan pengetahuan khusus domain. Anomali data dapat diklasifikasikan secara luas menjadi tiga kategori :

- **Anomali Sintaksis:** Anomali ini berkaitan dengan format dan representasi nilai data. Termasuk:
 - **Kesalahan Leksikal:** Terjadi ketika entri data menyimpang dari format yang diharapkan. Misalnya, nilai yang hilang dalam sebuah kolom, seperti nomor jaminan sosial (SSN), tuple 2 dan 3 pada Tabel 4.1, menunjukkan kesalahan leksikal.
 - **Kesalahan Format Domain:** Muncul ketika nilai tidak sesuai dengan tipe atau struktur data yang diharapkan. Konvensi penamaan yang tidak konsisten – misalnya, pada Tabel 4.1, tuple 1, 4, dan 5 mencantumkan nama orang tersebut sebagai {nama depan, nama belakang}, tetapi dua tuple lainnya, yaitu tuple kedua dan ketiga, hanya mencantumkan nama depan orang tersebut – melanggar ekspektasi format domain.
 - **Ketidakberaturan:** Melibatkan penggunaan satuan, singkatan, atau format yang tidak konsisten. Pada Tabel 4.1, perhatikan kolom ukuran, ukuran orang tersebut direpresentasikan baik dalam sentimeter maupun dalam kaki dan inci.

Karena kelas anomali ini mewakili pelanggaran format yang dibutuhkan atau ditentukan, maka ini adalah anomali sintaksis.

TABEL 4.1 Tabel Data yang Berisi Anomali Sintaksis, Semantik, dan Cakupan
Berikut adalah bentuk tabel dari gambar tersebut dengan judul kolom yang telah diterjemahkan ke dalam bahasa Indonesia:

No. Seri	Nama	Usia	Jenis Kelamin	SSN	Ukuran
1	Peter, Winter	23	M	XXX-XX-3456	7'1
2	Tom	-34	M		181
3	Sue	21	F		6'5
4	Sam, Joseph	19	M	XXX-XX-3456	5'9
5	Kalyan, Ram	56	M	XXX-XX-5101	5'7
6	Sam, Joseph	19	M	XXX-XX-3456	5'9
7	Kalyan, Ram	47	M	XXX-XX-5101	5'5

- **Anomali Semantik:** Anomali ini memengaruhi makna dan konsistensi logis data. Anomali semantik dapat berupa empat jenis dan meliputi :
 - **Pelanggaran Batasan Integritas:** Pelanggaran ini terkait dengan sekumpulan tupel yang tidak memenuhi satu atau lebih batasan integritas dalam skema relasional. Misalnya, pada Tabel 4.1, atribut usia harus lebih besar dari nol. Namun, kita dapat melihat bahwa nilai negatif telah diberikan pada usia seseorang dalam tupel 2 pada tabel.
 - **Kontradiksi:** Antar tupel, kontradiksi muncul setiap kali beberapa jenis ketergantungan antara nilai-nilainya dilanggar. Misalnya, skema relasional pada Tabel 4.1 menetapkan ketergantungan fungsional antara atribut SSN dan Nama. Namun, kita dapat melihat pelanggaran ketergantungan fungsional ini. Dua orang yang berbeda, yaitu Peter Winter dan Sam Joseph, keduanya memiliki SSN yang sama (XXX-XX-3456), yang merupakan kontradiksi.
 - **Duplikat:** Ketika dua atau lebih tupel yang mewakili entitas yang sama berada bersama, kita memiliki skenario di mana tupel tersebut sepenuhnya identik atau duplikat persis. Duplikat tidak persis dapat terjadi dalam suatu dataset sebagai kasus kontradiksi. Mereka mewakili entitas yang sama tetapi dengan nilai yang berbeda untuk semua atau sebagian atributnya. Dalam Tabel 4.1, tupel 4 dan 6 adalah duplikat persis, dan tupel 5 dan 7 adalah duplikat tidak persis.
 - **Tupel Tidak Valid:** Tupel tidak valid adalah kelas anomali yang paling rumit. Tidak valid berarti bahwa tupel secara struktural benar tetapi tetap tidak sesuai dengan entitas nyata apa pun. Anomali ini mengikuti aturan batasan integritas tetapi kasus-kasus ini tidak ada di dunia nyata. Kasus-kasus ini sangat sulit ditemukan dan tidak mungkin diperbaiki.
- **Anomali Cakupan:** Anomali cakupan berkaitan dengan data yang hilang dan representasi dunia nyata yang tidak lengkap:
 - **Nilai yang Hilang:** Hasil dari kelalaian selama pengumpulan data. Ini dapat melanggar batasan seperti NOT NULL dan memerlukan imputasi atau koreksi khusus domain.

- **Tuple yang Hilang:** Mengacu pada entitas utuh yang ada di dunia nyata tetapi tidak ada dalam dataset. Kelalaian ini dapat memengaruhi analisis dan mengurangi kelengkapan wawasan.

Klasifikasi ini memberikan pendekatan terstruktur untuk memahami jenis masalah yang harus ditangani oleh pembersihan data. Di bagian selanjutnya, kita akan mengeksplorasi bagaimana anomali ini muncul dalam *dataset sumber tunggal dan multi-sumber*, dan strategi yang digunakan untuk menyelesaikannya.

4.9 MASALAH KUALITAS DATA

Masalah kualitas data dapat muncul dari sumber data individual maupun integrasi beberapa sumber. Masalah-masalah ini selanjutnya dikategorikan sebagai masalah tingkat skema atau tingkat instansi. *Masalah tingkat skema* berkaitan dengan struktur dan desain model data. *Masalah tingkat instansi* menyangkut nilai data aktual dan konsistensinya. Mari kita periksa setiap kategori secara lebih rinci.

- **Masalah Sumber Tunggal:** Kualitas data dari satu sumber sangat bergantung pada seberapa baik data tersebut diatur oleh skema dan batasan integritas. Misalnya:
 - *File datar*, yang tidak memiliki penegakan skema, rentan terhadap kesalahan karena entri data yang tidak dibatasi.
 - *Sistem basis data*, di sisi lain, menerapkan model data dan batasan integritas yang ketat, mengurangi kemungkinan kesalahan.

Masalah tingkat skema dalam sumber tunggal meliputi desain skema yang buruk, batasan integritas yang hilang atau lemah, dan penegakan tipe dan hubungan data yang tidak memadai.

Berikut adalah bentuk tabel dari gambar tersebut yang telah diterjemahkan ke dalam bahasa Indonesia:

Ruang Lingkup Masalah		Data Kotor	Alasan/Keterangan
Atribut	Nilai tidak sah	bdate=30.13.70	Nilai di luar rentang domain
Rekaman	Dependensi atribut dilanggar	age=22, bdate=12.02.70	age=(tanggal saat ini - tanggal lahir) seharusnya berlaku
Tipe rekaman	Pelanggaran keunikan	emp1=(name="John Smith", SSN="123456") emp2=(name="Peter Miller", SSN="123456")	Keunikan untuk SSN (nomor jaminan sosial) dilanggar
Sumber	Pelanggaran integritas referensial	emp=(name="John Smith", deptno=127)	departemen yang direferensikan (127) tidak terdefinisi

Gambar 4.3 Contoh masalah sumber tunggal pada tingkat skema (pelanggaran batasan integritas).

Ruang Lingkup	Masalah	Data Kotor	Alasan/Keterangan
Atribut	Nilai yang hilang	phone=999-999999	Nilai tidak tersedia selama entri data (nilai <i>dummy</i> atau <i>null</i>)
	Salah eja	city="Liipzig"	biasanya salah ketik, kesalahan fonetik
	Nilai kriptik_Singkatan	experience="B"; occupation="DB Prog."	
	Nilai yang tersemat	name="J.Smith 12.02.70 New York"	beberapa nilai dimasukkan dalam satu atribut (misalnya dalam bidang format bebas)
	Nilai salah kolom	city="Germany"	
Rekaman	Dependensi atribut dilanggar	city="Redmond", zip=77777	kota dan kode pos seharusnya sesuai
	Transposisi kata	nme1="J.Smith", name2="Miller P."	biasanya dalam bidang format bebas
Tipe rekaman	Rekaman terduplikasi	emp1=(name="John Smith", ...); emp2=(name="J.Smith"... ,)	karyawan yang sama direpresentasikan dua kali karena beberapa kesalahan entri data
	Rekaman yang kontradiktif	emp1=(name="John Smith", bdate=12.02.70); emp2=(name="John Smith", bdate=12.12.70)	entitas dunia nyata yang sama dideskripsikan dengan nilai yang berbeda
Sumber	Referensi yang salah	emp=(name="John Smith", deptno=17)	departemen yang direferensikan (17) telah terdefinisi tetapi salah

Gambar 4.4 Contoh masalah sumber tunggal pada tingkat instans.

Masalah pada tingkat instansi meliputi kesalahan ketik, format yang tidak konsisten, catatan duplikat, dan nilai yang hilang. Gambar 4.3 dan 4.4 mengilustrasikan contoh masalah skema dan tingkat instansi, masing-masing. Misalnya, sebuah skema dapat mengizinkan entri duplikat untuk seseorang karena batasan yang tidak memadai, seperti mengizinkan "John Smith" dan "J. Smith" untuk ada bersamaan tanpa validasi. Untuk mengurangi masalah ini, sangat penting untuk:

- Merancang skema yang kuat dengan batasan yang sesuai
- Menerapkan mekanisme validasi pada titik entri data
- Mencegah data kotor masuk ke sistem sejak awal
- **Masalah Sumber Ganda:** Ketika mengintegrasikan data dari berbagai sumber, masalah yang ada seringkali diperparah. Setiap sumber mungkin telah dikembangkan secara independen, yang menyebabkan heterogenitas yang signifikan dalam model data, desain skema, terminologi, dan format data.

Konflik tingkat skema dalam data tersebut meliputi:

- **Konflik Penamaan:** Nama yang sama untuk entitas yang berbeda (homonim) atau nama yang berbeda untuk entitas yang sama (sinonim)

- **Konflik Struktural:** Variasi dalam cara data direpresentasikan (misalnya, atribut vs. tabel, tipe data yang berbeda, atau struktur komponen)

Konflik tingkat instansi meliputi:

- Catatan yang saling bertentangan
- Entri duplikat di berbagai sumber
- Representasi nilai yang tidak konsisten (misalnya, “M” vs. “Laki-laki” atau “0” vs. “F” untuk jenis kelamin)
- Satuan pengukuran yang berbeda (misalnya, dolar vs. euro)
- Tingkat agregasi yang bervariasi (misalnya, penjualan per produk vs. penjualan per kelompok produk)
- Ketidaksesuaian temporal (misalnya, data dari periode waktu yang berbeda)

Pelanggan (sumber 1)

CID	NAMA	Jalan	Kota	Jenis Kelamin
11	Kristen Smith	2 Hurley Pl	South Fork, Mk 48503	0
24	Christian Smith	Hurley St 2	S Fork MN	1

Klien (sumber 2)

Cno	Nama Belakang	Nama Depan	Jenis Kelamin	Alamat	Telepon/Faks
24	Smith	Christoph	M	23 Harley St, Chicago IL, 60633-2394	333-222-6542/ 333-222-6599
493	Smith	Kris L.	F	2 Hurley Place, South Fork MN, 48503-5998	444-555-6666

Pelanggan (target terintegrasi dengan data yang dibersihkan)

No	Nama Belakang	Nama Depan	Jenis Kelamin	Jalan	Kota	Negara Bagian	Kode Pos	Telepon	Faks	CID	Cno
1	Smith	Kristen L.	F	2 Hurley Place	South fork	MN	48503-5998	444-555-6666		11	493
2	Smith	Christian	M	2 Hurley Place	South fork	MN	48503-5998			24	
3	Smith	Christoph	m								

Gambar 4.5 Contoh masalah multi-sumber pada tingkat skema dan instansi.

Tantangan utama dalam integrasi multi-sumber adalah menyelesaikan *masalah identitas objek* – menentukan apakah catatan dari sumber yang berbeda merujuk pada entitas dunia nyata yang sama. Ini juga dikenal sebagai *penghapusan duplikat* atau *masalah penggabungan/pembersihan*. Gambar 4.5 mengilustrasikan konflik skema dan tingkat instance antara dua sumber relasional. Ini menunjukkan bagaimana inkonsistensi penamaan (misalnya,

"Pelanggan" vs. "Klien") dan perbedaan struktural (misalnya, format alamat) mempersulit integrasi. Solusinya melibatkan harmonisasi skema diikuti dengan pembersihan data untuk memastikan konsistensi dan akurasi. Bagian ini menyoroti pentingnya mengatasi masalah skema dan tingkat instance selama integrasi data.

4.10 RINGKASAN

Dalam bab ini, kita telah mengeksplorasi teknologi dan proses utama yang mendukung penanganan *data* yang terukur dan andal dalam sistem Big Data, dengan penekanan khusus pada *pembersihan data*. Kami memulai dengan meninjau tiga paradigma pemrosesan data utama – *batch*, *stream*, dan *hybrid* – dan memeriksa kerangka kerja representatif untuk masing-masing:

- *Hadoop* untuk pemrosesan batch
- *Apache Storm* untuk pemrosesan stream
- *Apache Spark*, *Flink*, dan *Samza* untuk arsitektur hybrid dan stream-first

Kerangka kerja ini memungkinkan organisasi untuk memproses kumpulan data besar secara efisien, baik untuk analisis historis maupun pengambilan keputusan secara real-time. Kemudian kami mengalihkan fokus ke *proses pembersihan data*, langkah penting dalam pipeline data yang memastikan akurasi, konsistensi, dan kegunaan. Data yang bersih sangat penting untuk analitik yang bermakna dan aplikasi AI yang dapat dipercaya. Kami membahas:

- Peran *alur kerja ETL* dalam mengkonsolidasikan dan mempersiapkan data
- *Klasifikasi anomali data*, termasuk masalah sintaksis, semantik, dan cakupan
- Tantangan dalam menjaga kualitas data di lingkungan *sumber tunggal dan multi-sumber*

Sepanjang bab ini, kami menekankan bahwa meskipun otomatisasi dapat membantu dalam pembersihan data, keahlian domain tetap sangat diperlukan. Pembersihan data yang efektif membutuhkan kombinasi alat teknis, alur kerja strategis, dan pemahaman kontekstual. Landasan ini menjadi landasan bagi bab-bab selanjutnya, di mana kita akan menerapkan teknik AI pada kumpulan data yang telah dibersihkan untuk mengekstrak wawasan, membangun model prediktif, dan mendukung pengambilan keputusan yang cerdas.

4.11 PERTANYAAN

1. Bandingkan dan kontraskan pemrosesan batch, pemrosesan stream, dan pemrosesan hibrida dalam sistem Big Data. Berikan contoh kasus yang relevan untuk setiap paradigma pemrosesan.
2. Analisis arsitektur sistem Big Data Hadoop, termasuk HDFS, YARN, dan MapReduce. Evaluasi keuntungan dan keterbatasan Hadoop dalam konteks pemrosesan Big Data modern.
3. Definisikan pembersihan data dan jelaskan klasifikasi anomali data (sintaksis, semantik, dan cakupan). Ilustrasikan setiap jenis dengan contoh yang sesuai.
4. Manakah dari berikut ini yang secara akurat menggambarkan karakteristik Apache Spark?

- a. Ketergantungan yang tinggi pada penyimpanan disk untuk hasil sementara
 - b. Pemanfaatan RDD untuk komputasi dalam memori
 - c. Penyediaan default jaminan pemrosesan tepat sekali
 - d. Dukungan terbatas untuk pemrosesan stream
5. Dalam konteks proses ETL, fase Transformasi terutama melibatkan
- a. Ekstraksi data mentah dari berbagai sumber
 - b. Memuat data yang bersih dan terintegrasi ke dalam gudang data
 - c. Penerapan aturan seperti penggabungan, pengurutan, pengkodean, dan penggabungan data
 - d. Deteksi konflik skema antara beberapa sumber
6. Latihan Pemrograman – Dasar Kembangkan program Python untuk mengidentifikasi dan mengganti nilai yang hilang dalam sebuah dataset. Implementasikan imputasi rata-rata untuk kolom numerik dan imputasi modus untuk kolom kategorikal.
7. Latihan Pemrograman – Lanjutan Rancang dan implementasikan pipeline ETL berbasis Python yang menyelesaikan tugas-tugas berikut:
- Mengekstrak data dari dua file CSV dengan skema yang berbeda
 - Mentransformasi data untuk memastikan keseragaman skema (misalnya, nama dan format kolom yang konsisten)
 - Membersihkan data dengan menghapus duplikat dan menyelesaikan inkonsistensi
 - Memuat data yang telah dibersihkan ke dalam DataFrame yang terkonsolidasi

BAB 5

OTOMATISASI PEMBERSIHAN DATA

5.1 PENGENALAN PIPELINE DATA CLEANING

Dalam bab ini, kita akan mendemonstrasikan cara membangun pipeline otomatis yang memfasilitasi pembersihan data – langkah penting dalam mempersiapkan data untuk analisis dan pemodelan. Kita berasumsi bahwa data telah dikumpulkan menggunakan teknik yang dibahas dalam Bab 2. Tujuan dari pipeline ini adalah untuk menilai dan meningkatkan kualitas data melalui operasi pembersihan otomatis. Fungsi-fungsi yang diperkenalkan di sini dirancang agar modular dan dapat digunakan kembali, sehingga dapat diterapkan secara konsisten di berbagai dataset atau berulang kali dalam satu dataset.

Modularitas ini mendukung penggunaan kembali kode dan menyederhanakan pemeliharaan. Skrip Python yang digunakan dalam bab ini diadaptasi dari pekerjaan sebelumnya dan bergantung pada pustaka yang banyak digunakan seperti *NumPy*, *Matplotlib*, dan *Pandas*. Pembaca harus memastikan paket-paket ini telah diinstal sebelum melanjutkan. Kita mulai dengan memeriksa struktur dan properti suatu dataset menggunakan serangkaian tugas standar:

1. Melihat beberapa baris dan kolom pertama
2. Menghitung baris dan kolom
3. Memeriksa tipe data
4. Memeriksa pengidentifikasi unik

Tugas-tugas ini dapat dikapsulkan dalam fungsi Python dan diorganisasikan ke dalam modul – kumpulan skrip yang dapat diimpor ke dalam aplikasi lain. Untuk mengilustrasikan konsep-konsep ini, kita menggunakan dataset *National Longitudinal Survey of Youth 1997 (NLSY97)*, yang berisi data longitudinal yang dikumpulkan setiap tahun dari tahun 1997 hingga 2023. Dataset ini, yang tersedia di Kaggle, sangat cocok untuk mendemonstrasikan penilaian kualitas data dan teknik pembersihan. Skrip python yang disediakan di sini diambil dari.

5.2 PEMERIKSAAN AWAL DATA

Sebelum melakukan pembersihan data, penting untuk melakukan pemeriksaan awal terhadap dataset. Terlepas dari struktur atau sumbernya, langkah pertama adalah memuat data ke dalam Pandas DataFrame dan memeriksa properti dasarnya. Ini termasuk:

- Melihat beberapa baris dan kolom pertama
- Menghitung jumlah baris dan kolom
- Memeriksa tipe data setiap kolom
- Memeriksa keunikan pengidentifikasi baris

TABEL 5.1 Fungsi untuk Menjelajahi Konten Data dalam Kerangka Data

```
import pandas as pd
def getfirstlook(df, nrows=5, uniqueids=None):
```

```
out = {}
out['head'] = df.head(nrows)
out['dtypes'] = df.dtypes
out['nrows'] = df.shape
out['ncols'] = df.shape
out['index'] = df.index
if uniqueids is not None:
    out['uniqueids'] = df[uniqueids].nunique()
return out

def displaydict(dicttodisplay):
    print(*( ': '.join(map(str, x)) \
    for x in dicttodisplay.items()), sep='\n\n')
```

Tugas-tugas ini dapat dienkapsulasi dalam fungsi Python yang dapat digunakan kembali. Modul pada Tabel 5.1 menyediakan dua fungsi tersebut: satu untuk meringkas dataset dan yang lainnya untuk menampilkan ringkasan dalam format yang mudah dibaca. Skrip python yang disediakan di sini telah diadaptasi dari. Fungsi *getfirstlook* mengembalikan kamus yang berisi data ringkasan dalam *DataFrame*. Kedua, fungsi *displaydict* diimplementasikan untuk meningkatkan tampilan kamus. Selanjutnya, kita memuat data NLS dan mengimpor pustaka *pandas*, *os*, dan *sys* (Tabel 5.2).

5.3 MENGIDENTIFIKASI NILAI YANG HILANG

Data yang hilang adalah salah satu masalah paling umum dan kritis dalam kumpulan data dunia nyata. Untuk mengatasi hal ini, kita mendefinisikan fungsi yang disebut *getmissings* yang menghitung nilai yang hilang berdasarkan kolom dan baris (Tabel 5.3). Fungsi ini dapat digunakan kembali di berbagai kumpulan data dan mendukung pelaporan absolut dan berbasis persentase.

TABEL 5.2 Skrip Python untuk Mengakses Kumpulan Data dan Memeriksa

```
import pandas as pd
import os
import sys
from google.colab import drive
drive.mount('/content/drive')
nls97 = pd.read_csv('/content/drive/My Drive/nls97g.csv',
    low_memory=False)
nls97.set_index('personid', inplace=True)
```

Catatan: Tutorial ini dikembangkan di lingkungan Google *Colab*. Dataset NLS (*nls97g.csv*) diunduh dan disimpan di folder */content/drive*.

TABEL 5.3 Skrip Python untuk Memeriksa Kumpulan Data NLS

```
dinfo = bd.getfirstlook(nls97)
bd.displaydict(dinfo)
dinfo['nrows']
dinfo['dtypes']
```

```
def getmissings(df, byrowperc=False):
```

```
return (
    df.isnull().sum(),
    df.isnull().sum(axis=1)
).value_counts(normalize=True)
.sort_index()

missingsbycols, missingsbyrows = \
    bd.getmissings(nls97[['<weeksworked20>',
    '<weeksworked21>']], True)
missingsbycols
missingsbyrows
ssingsbycols, missingsbyrows = \
    bd.getmissings(nls97[['<weeksworked20>',
    '<weeksworked21>']])
Missingsbyrows
```

Fungsi *getmissings* akan menerima dataset NLS (sebagai dataframe) dan sebuah argumen, dan akan *mengembalikan dua seri: satu dengan nilai yang hilang per baris*, dan satu dengan nilai yang hilang untuk setiap kolom. Kita juga memanggil fungsi *getmissings* pada dataframe *nls97* untuk menganalisis jumlah entri yang hilang per baris (rekaman) dan kolom (variabel). Pada bagian selanjutnya, kita akan *menunjukkan cara mengidentifikasi outlier* dan nilai yang tidak terduga dalam dataset menggunakan fungsi.

5.4 DETEKSI ANOMALI OUTLIER

Deteksi outlier adalah langkah penting dalam pembersihan data. Outlier dan nilai yang tidak terduga dapat mendistorsi analisis statistik, membiaskan model pembelajaran mesin, dan menyebabkan kesimpulan yang salah. Oleh karena itu, mengidentifikasi dan mengatasi anomali ini sangat penting sebelum melanjutkan tugas pemodelan atau pengambilan keputusan. Untuk mendukung hal ini, kami memperkenalkan serangkaian fungsi Python yang dapat digunakan kembali pada Tabel 5.4 yang :

- Menganalisis distribusi variabel
- Menghitung properti statistik seperti mean, median, skewness, dan kurtosis
- Mendeteksi outlier menggunakan metode rentang interkuartil (IQR)
- Memvisualisasikan distribusi menggunakan histogram dan boxplot

Fungsi-fungsi ini dirancang agar modular dan dapat diterapkan di berbagai dataset. Fungsi-fungsi ini bergantung pada *Pandas*, *NumPy*, *Matplotlib*, dan *SciPy*. Pembaca harus menginstal pustaka-pustaka ini sebelum menggunakan kode ini. Mari kita bahas setiap fungsi pada Tabel 5.4:

- **getdistprops:** Metrik distribusi, bentuk, dan kecenderungan sentral dapat ditetapkan menggunakan fungsi ini. Fungsi ini mengembalikan kamus yang berisi ukuran-ukuran ini. Perhatikan juga bahwa fungsi ini menangani keadaan di mana uji Shapiro untuk normalitas tidak menghasilkan hasil.

- **getoutliers:** Fungsi ini menetapkan ambang batas outlier untuk kolom yang berada di atas kuartil ketiga atau di bawah kuartil pertama sebesar 1,5 kali IQR, yaitu jarak antara kuartil pertama dan ketiga.
- **makeplot:** Fungsi ini menerima seri, judul, dan label untuk sumbu x dan membuat histogram sebagai plot default.

TABEL 5.4 Skrip Python untuk Fungsi Menampilkan Distribusi, Mengidentifikasi Nilai Ekstrem, dan Memverifikasi Distribusi Variabel

```
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as scistat
import math
def getdistprops(seriestotest):
    out = {}
    normstat, normpvalue = scistat.shapiro(seriestotest.dropna())
    if not math.isnan(normstat):
        out['normstat'] = normstat
    if normpvalue >= 0.05:
        out['normpvalue'] = str(round(normpvalue, 2)) + ": Accept Normal"
    elif normpvalue < 0.05:
        out['normpvalue'] = str(round(normpvalue, 2)) + ": Reject Normal"
    out['mean'] = seriestotest.mean()
    out['median'] = seriestotest.median()
    out['std'] = seriestotest.std()
    out['kurtosis'] = seriestotest.kurtosis()
    out['skew'] = seriestotest.skew()
    out['count'] = seriestotest.count()
    return out

import numpy as np
def getoutliers(dfin, sumvars, othervars):
    dfin = dfin[sumvars + othervars]
    dfout = pd.DataFrame(columns=dfin.columns)

    for col in sumvars:
        col_data = dfin[col].dropna().to_numpy() # Convert to NumPy array and drop NaNs
        q1 = np.percentile(col_data, 25)
        q3 = np.percentile(col_data, 75)
        iqr = 1.5 * (q3 - q1)
        outlier_low = q1 - iqr
        outlier_high = q3 + iqr

        mask = (dfin[col] < outlier_low) | (dfin[col] > outlier_high)
        df = dfin.loc[mask].copy()
        df['varname'] = col
        df['threshlow'] = outlier_low
        df['threshhigh'] = outlier_high
```

```
dfout = pd.concat([dfout, df], ignore_index=True)

return dfout
def makeplot(seriestoplot, title, xlabel, plottype="hist"):
    data = seriestoplot.dropna().to_numpy() # ensure clean NumPy
    array without NaNs
    if plottype == "hist":
        plt.hist(data, bins='auto', alpha=0.7, edgecolor='black')
        plt.axvline(np.mean(data), color='red', linestyle='dashed',
linewidth=1, label='Mean')
        plt.xlabel(xlabel)
        plt.ylabel("Frequency")

        plt.legend()
    elif plottype == "box":
        plt.boxplot(data, labels=[xlabel])
        plt.ylabel(xlabel)
    plt.title(title)
    plt.tight_layout()
    plt.show()
```

Tabel 5.5 menyediakan skrip python untuk menjalankan fungsi-fungsi ini. Pembaca dianjurkan untuk menjalankan skrip python yang disediakan dalam Tabel 5.4 dan 5.5 dan meluangkan waktu untuk memahami hasil analisis dan visualisasi yang diperoleh.

TABEL 5.5 Skrip Python untuk Memanggil Fungsi-Fungsi Praktis yang Diimplementasikan pada Tabel 5.4

```
import pandas as pd
import os
import sys
import pprint
nls97 = pd.read_csv('/content/drive/My Drive/nls97g.csv',
low_memory=False)
nls97.set_index('personid', inplace=True)

sumvars = ['satmath', 'wageincome20']
othervars = ['originalid', 'highestdegree', 'gender',
'maritalstatus']
outliers = getoutliers(nls97, sumvars, othervars)

# Display outlier counts per variable
print(outliers['varname'].value_counts(sort=False))

outliers.loc[outliers.varname == 'satmath', othervars + sumvars]
outliers.to_excel('/content/nlsoutliers.xlsx')

makeplot(nls97.satmath, "Histogram of SAT Math", "SAT Math")

makeplot(nls97['satmath'], "Boxplot of SAT Math", "SAT Math", "box")
```

5.5 PEMERIKSAAN KUALITAS DATA

Memastikan kualitas data adalah langkah mendasar dalam setiap proyek analisis data atau pembelajaran mesin. Kualitas data yang buruk dapat menyebabkan wawasan yang menyesatkan, model yang bias, dan prediksi yang tidak dapat diandalkan. Pendekatan yang kuat untuk validasi data melibatkan penetapan tujuan kualitas data dan implementasi fungsi yang secara sistematis memeriksa apakah dataset memenuhi tujuan tersebut. Tujuan kualitas data yang umum meliputi :

- Rentang yang dapat diterima untuk variabel numerik
- Kategori yang valid untuk variabel kategorikal
- Ambang batas untuk nilai yang hilang
- Tingkat kemiringan dan kurtosis yang dapat diterima
- Keunikan dan kelengkapan bidang pengidentifikasi

Fungsi Python berikut membantu memvalidasi aspek-aspek dataset ini menggunakan pendekatan modular dan dapat digunakan kembali. Mari kita jelajahi lebih lanjut dan demonstrasikan tujuan kualitas data untuk dataset NLS yang ditunjukkan pada Gambar 5.1. Untuk demonstrasi ini, kami berasumsi bahwa tidak lebih dari 20% entri untuk kategori *status perkawinan* boleh hilang, dan bahwa nilai kategori hanya akan mencakup nilai *Cerai, Menikah, Belum pernah menikah, Berpisah, dan Janda/Duda*. Tujuan kualitas data juga mewajibkan bahwa variabel numerik *tidur malam* akan memiliki kemiringan dan kurtosis seperti distribusi normal, memiliki nilai hanya antara 3 dan 9, dan tidak lebih dari 30% nilai yang hilang. Selain itu, tujuan tersebut juga memperhatikan outlier. Pada Tabel 5.6, kami membuat berbagai fungsi yang akan membantu mencapai tujuan kualitas data. Skrip python yang disediakan di sini telah diadaptasi dari.

Nama Variabel	Tipe	Kategori	Rentang	Ambang Batas Hilang	Target Skewn ess	Target Kurtosis	Tampilkan Outlier	Sertakan
maritalstatus	kategorikal	Bercerai Menikah Belum menikah		0.2				Y
colenroct21	kategorikal	1.Tidak terdaftar 2. 2-tahun		0.2				N
personid	unik							N
originalid	unik							Y
birthyear	numerik		1980 1984	0.3	0	0	N	N
highestgradecompleted	numerik		5 16	0.3	0	3	N	Y
childathome	numerik		0 7	0.3	0	3	Y	N
childnotathome	numerik		0 7	0.3	0	3	N	N

satverbal	numerik	200 800	0.3	0	3	N	N
satmath	numerik	200 800	0.3	0	3	N	N
gpaoverall	numerik	100 400	0.3	0	3	N	Y
gpaenglish	numerik	100 400	0.3	0	3	N	N
gpamath	numerik	100 400	0.3	0	3	N	N
gpascience	numerik	100 400	0.3	0	3	N	N
weeksworked21	numerik	0 52	0.3	0	3	N	N
nightlyhrssleep	numerik	3 9	0.3	0	3	Y	Y

Gambar 5.1 Pemeriksaan data untuk kolom NLS terpilih.

TABEL 5.6 Skrip Python untuk Fungsi Memeriksa Kualitas Keseluruhan Kumpulan Data

```
def checkcats(cat1,cat2):
    missingcats = set(cat1).symmetric_difference(set(cat2))
    return missingcats
def checkoutliers(values):
    thirdq, firstq = values.quantile(0.75),values.quantile(0.25)
    interquartilerange = 1.5*(thirdq-firstq)
    outlierhigh, outlierlow = interquartilerange+thirdq,
firstq-interquartilerange

    return outlierhigh, outlierlow
def runchecks(df,dc,numvars,catvars,idvars):
    for col in df[catvars]:
        dcvals = dc.loc[col]
        print(«\n\nChecks for categorical variable», col)
        compcat = list(dcvals.categories.split('|'))
        valuediff = checkcats(compcat,df[col].dropna().str.strip().
unique())
        if len(valuediff) > 0:
            print(«at least one non-matching category:»,valuediff)
        missingper = df[col].isnull().sum()/df.shape
        if missingper > dcvals.missingthreshold:
            print(«missing percent beyond threshold of»,dcvals.
missingthreshold, «is», missingper)
    for col in df[numvars]:
        dcvals = dc.loc[col]
        print(«\n\nChecks for numeric variable», col)
        range = np.fromstring(dcvals.range, sep='|')
        min = df[col].min()
        max = df[col].max()
        if min < range:
            print(«at least one record below range starting at «,range,
"min value is", min)
        if max > range:
```

```
        print(«at least one record above range ending at »,range,
"max value is", max)
        missingper = df[col].isnull().sum()/df.shape
        if missingper > dcvals.missingthreshold:
            print(«missing percent beyond threshold of», dcvals.
missingthreshold, «is», missingper)
            if dcvals.showoutliers == "Y":
                outlierhigh, outlierlow = checkoutliers(df[col])
                print(«\nvalues less than», outlierlow, «\n»,
df.loc[df[col]<outlierlow,col].agg(["min", 'max', 'count']), end="\n")
                print(«\nvalues greater than», outlierhigh,»\n»,
df.loc[df[col]>outlierhigh,col].agg(["min", 'max', 'count']),
end="\n")
                skewcol = df[col].skew()
                if abs(skewcol-dcvals.skewtarget)>1.2:
                    print(«skew substantially different from target of»,dcvals.
skewtarget, «is», skewcol)

                kurtosiscol = df[col].kurtosis()
                if abs(kurtosiscol-dcvals.kurtosistarget)>1.2:
                    print(«kurtosis substantially different from target of»,dcvals.
kurtosistarget, «is», kurtosiscol)

for col in df[idvars]:
    print(«\n\nChecks for id variable», col)
    uniquevals = df[col].nunique()
    nrows = df.shape
    if uniquevals != nrows:
        print(«not unique identifier», uniquevals,»unique values not
equal to», nrows, «rows.»)
    missingvals = df[col].isnull().sum()
    if missingvals > 0:
        print("unique value has", missingvals,"missing values")
```

Mari kita bahas secara singkat fungsi-fungsi dalam Tabel 5.6:

- **Checkcat:** Fungsi ini mengambil semua nilai yang diberikan dalam beberapa daftar dan membandingkannya untuk melihat perbedaan antar daftar.
- **Checkoutliers:** Fungsi ini mengambil nilai-nilai dalam daftar dan mengujinya untuk keberadaan outlier.
- **runchecks:** Fungsi ini mengambil data frame (df), target data (dc), daftar kolom numerik (numvars), daftar kolom kategorikal (catvars), dan daftar kolom pengidentifikasi (idvars) untuk menjalankan semua pemeriksaan.

(Catatan: Jika flag *showoutliers* diatur ke Y, maka outlier akan ditampilkan. Fungsi *checkoutliers* digunakan untuk mengidentifikasi outlier menggunakan perhitungan IQR yang sederhana. Kemiringan dan kurtosis suatu variabel diperiksa untuk menentukan seberapa jauh nilainya dari distribusi normal. Beberapa pemeriksaan sederhana dilakukan untuk setiap variabel untuk menentukan apakah ada duplikat atau nilai yang hilang). Pada Tabel 5.7, kami menjalankan pemeriksaan data. Kami mulai dengan menyediakan data frame NLS dan target pemeriksaan data sebagai argumen untuk fungsi tersebut.

Setelah menjalankan skrip python pada Tabel 5.7, kami memperoleh hasil yang ditunjukkan pada Gambar 5.2. Perhatikan bahwa variabel *status perkawinan* memiliki lebih banyak entri yang hilang (26%). Skor untuk *nilai tertinggi yang diselesaikan* dan *IPK* keseluruhan keduanya lebih tinggi dari yang diharapkan. Kedua variabel tersebut memiliki kurtosis minimal. Terdapat outlier di bawah dan di atas IQR untuk *tidur malam*. Beberapa pengamatan menarik yang diperoleh adalah :

TABEL 5.7 Skrip Python untuk Memeriksa Kualitas Keseluruhan Dataset NLS

```
import pandas as pd
import numpy as np
import os
import sys
nls97 = pd.read_csv('/content/drive/My Drive/nls97g.csv',
    low_memory=False)
dc = pd.read_csv("/content/drive/My Drive/datacheckingtargets.csv")
dc.set_index('varname', inplace=True)

nls97.originalid.head(7)

nls97.loc[nls97.originalid==2, "originalid"] = 1
nls97.loc[nls97.originalid.between(3,7), "originalid"] = np.nan
nls97.originalid.head(7)

nls97["highestgradecompleted"] = nls97.highestgradecompleted.
    replace(95, np.nan)
dc = dc.loc[dc.include=="Y"]
numvars = dc.loc[dc.type=="numeric"].index.to_list()
catvars = dc.loc[dc.type=="categorical"].index.to_list()
idvars = dc.loc[dc.type=="unique"].index.to_list()

runchecks(nls97, dc, numvars, catvars, idvars)
```

- Tiga puluh satu responden tidur dua jam atau kurang setiap malam
- Dua puluh tujuh responden tidur 12 jam atau lebih setiap malam, yang sangat tinggi

Latihan dataset NLS telah menunjukkan bagaimana seseorang dapat secara efektif memeriksa kualitas data dengan memanfaatkan keahlian domain dari masa lalu dan menggunakan latar belakang serta pengetahuan statistik mereka.

5.6 PIPELINE MACHINE LEARNING

Tugas pembersihan data – seperti penskalaan, pengkodean, penanganan outlier, pengisian nilai yang hilang, dan penghapusan duplikat – merupakan komponen penting dari setiap alur kerja analitik. Tugas-tugas ini sering dilakukan berulang kali di berbagai proyek, sehingga otomatisasi melalui pipeline menjadi solusi yang praktis dan terukur. Pustaka *scikit-learn* Python menyediakan kerangka kerja yang kuat untuk membangun pipeline tersebut. Pipeline membantu menyederhanakan pra-pemrosesan, mengurangi kebocoran data, dan memastikan reproduksibilitas dengan menggabungkan beberapa langkah transformasi dan komponen pemodelan.

```
Checks for categorical variable maritalstatus
missing percent beyond threshold of 0.2 is 0.25701246660730187

Checks for numeric variable highestgradecompleted
at least one record above range ending at 16.0 max value is 20.0
kurtosis substantially different from target of 3.0 is -0.6076960550376977

Checks for numeric variable gpaoverall
at least one record below range starting at 100.0 min value is 10.0
at least one record above range ending at 400.0 max value is 417.0
missing percent beyond threshold of 0.3 is 0.3317008014247551
kurtosis substantially different from target of 3.0 is 0.3125440750593671

Checks for numeric variable nightlyhrssleep
at least one record below range starting at 3.0 min value is 0.0
at least one record above range ending at 9.0 max value is 20.0

values less than 3.0
min      0.0
max      2.0
count    31.0
Name: nightlyhrssleep, dtype: float64

values greater than 11.0
min      12.0
max      20.0
count    27.0
Name: nightlyhrssleep, dtype: float64

kurtosis substantially different from target of 3.0 is 5.087650564985399

Checks for id variable originalid
not unique identifier 9978 unique values not equal to 9984 rows
```

Gambar 5.2 Ringkasan pemeriksaan data pada dataset NLS.

Beberapa manfaat menggunakan pipeline meliputi:

- **Modularitas:** Setiap langkah dalam pipeline dikapsulkan sebagai komponen yang dapat digunakan kembali.
- **Reproduksibilitas:** Pipeline memastikan pra-pemrosesan yang konsisten di seluruh dataset pelatihan dan pengujian.
- **Otomatisasi:** Setelah didefinisikan, pipeline dapat dieksekusi dengan intervensi manual minimal.
- **Integrasi:** Pipeline dapat disematkan ke dalam alur kerja ML yang lebih besar, termasuk validasi silang dan evaluasi model.

Untuk mendemonstrasikan konstruksi pipeline, kami menggunakan versi sederhana dari dataset COVID-19 dari Our World in Data (<https://ourworldindata.org/covid-cases>). Pipeline ini akan:

- Menskalakan fitur numerik
- Mendeteksi dan menangani outlier
- Mengisi nilai yang hilang
- Melatih model regresi

Tabel 5.8 menyediakan skrip python untuk mengimplementasikan pipeline ini.

TABEL 5.8 Skrip Python untuk Membuat Pipeline

```
import pandas as pd
```

```
import os
import sys

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_validate
from sklearn.model_selection import KFold

covidtotals = pd.read_csv("/content/sample_data/covidtotals.csv")
feature_cols = ['total_cases', 'total_deaths', 'population']

X_train, X_test, y_train, y_test = \
train_test_split(covidtotals[feature_cols], \
covidtotals[['total_cases']], test_size=0.1, random_state=0)

kf = KFold(n_splits=5, shuffle=True, random_state=0)
class OutlierTrans():
    def __init__(self, threshold=1.5):
        self.threshold = threshold

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):
        Xnew = X.copy()
        for col in Xnew.columns:
            thirdq, firstq = Xnew[col].quantile(0.75), \
                Xnew[col].quantile(0.25)
            inlierrange = self.threshold*(thirdq-firstq)
            outlierhigh, outlierlow = inlierrange+thirdq, \
                firstq-inlierrange
            Xnew.loc[(Xnew[col]>outlierhigh) | \
                (Xnew[col]<outlierlow), col] = np.nan
        return Xnew.values

pipeline = \
make_pipeline(StandardScaler(), OutlierTrans(2),
SimpleImputer(strategy="mean"), LinearRegression())

scores = \
cross_validate(pipeline, X=X_train, y=y_train.values,
cv=kf, scoring=['r2', 'neg_mean_absolute_error'],
n_jobs=1)
print("Mean Absolute Error: %.2f, R-squared: %.2f" %
(scores['test_neg_mean_absolute_error'].mean(),
scores['test_r2'].mean()))
```

Di sini, kita mulai dengan memuat modul scikit-learn. Pipeline yang dirancang mengotomatiskan aktivitas pembersihan data pada dataset COVID-19. Fungsi *StandardScaler* digunakan untuk menstandarisasi fitur dan fungsi *SimpleImputer* digunakan untuk mengimputasi nilai untuk data yang hilang. Selain itu, fungsi *outliertrans* yang ditentukan

pengguna digunakan untuk menangani lebih lanjut nilai yang hilang. Fungsi *make_pipeline* menggabungkan semua tugas pembersihan. Tugas pemodelan analitik lainnya termasuk pemisahan dataset untuk pelatihan dan pengujian, dan validasi silang *k-fold* untuk menentukan kinerja model juga dilakukan dalam pipeline.

Dataset COVID-19 pertama kali diunduh ke drive */content/sample_data* di lingkungan *colab*. Setelah dimuat, pemisahan pelatihan dan pengujian dilakukan pada dataset. Selain itu, validasi silang *k-fold* ($k = 5$) dengan pengacakan data juga dilakukan. Langkah selanjutnya adalah mempersiapkan pipeline. Pipeline ini mampu melakukan penskalaan standar, memasukkan nilai rata-rata ketika nilai hilang, dan kemudian memodelkan regresi linier pada dataset. Semua fitur yang dipilih dari dataset ditangani dengan cara yang serupa. Setelah membangun pipeline dan memulai objek validasi silang *k-fold*, tugas pembersihan dilakukan, estimasi model diperoleh, dan semua metrik evaluasi dihasilkan.

Perhatikan bahwa pipeline dan dataset pelatihan keduanya diberikan sebagai input ke fungsi *cross_validate*. Kelas *OutlierTrans* mengidentifikasi nilai yang hilang berdasarkan jarak yang diwakili sebagai IQR. Untuk bekerja dengan pipeline *scikit-learn*, kelas ini mengimplementasikan metode *fit* dan *transform*. Hampir semua aksi terjadi di dalam metode *transform* dari kelas tersebut. Setiap nilai yang lebih dari 1,5 kali IQR di atas kuartil ketiga atau di bawah kuartil pertama dianggap hilang. Kami berharap latihan sederhana pembuatan pipeline data otomatis ini akan berfungsi sebagai pengantar dasar untuk pipeline data. Pipeline data yang kompleks dapat dibuat dengan menambahkan kemampuan tambahan yang diperlukan sesuai kebutuhan.

5.7 RINGKASAN

Bab ini menyajikan pendekatan sistematis untuk membangun pipeline pembersihan data otomatis – kemampuan penting untuk sistem AI dan Big Data modern. Kami menekankan pentingnya modularitas, penggunaan kembali, dan skalabilitas dalam merancang fungsi dan alur kerja yang mendukung pra-pemrosesan data yang konsisten dan efisien. Beberapa aspek penting yang dibahas meliputi:

- **Fungsi Modular untuk Inspeksi:** Kami memperkenalkan fungsi Python yang dapat digunakan kembali untuk memeriksa dataset, termasuk melihat struktur, menghitung baris dan kolom, memeriksa tipe data, dan memverifikasi pengidentifikasi unik.
- **Analisis Nilai yang Hilang:** Kami menunjukkan cara mengidentifikasi nilai yang hilang berdasarkan baris dan kolom, mendukung pelaporan absolut dan berbasis persentase.
- **Deteksi Pencilan dan Analisis Distribusi:** Kami mengimplementasikan fungsi untuk menghitung properti statistik, mendeteksi pencilan menggunakan IQR, dan memvisualisasikan distribusi menggunakan histogram dan boxplot.
- **Validasi Kualitas Data:** Kami mendefinisikan tujuan kualitas data dan membuat fungsi untuk memvalidasi bidang kategorikal, numerik, dan pengidentifikasi terhadap ambang batas untuk nilai yang hilang, kemiringan, kurtosis, dan rentang yang dapat diterima.
- **Konstruksi Pipeline dengan *scikit-learn*:** Kami membangun pipeline otomatis dasar menggunakan *scikit-learn* untuk menskalakan fitur, menangani outlier, mengimputasi

nilai yang hilang, dan melatih model regresi. Pipeline ini dievaluasi menggunakan validasi silang dan menunjukkan bagaimana komponen modular dapat dirantai bersama untuk pembersihan data yang dapat direproduksi dan diskalakan.

Bersama-sama, teknik-teknik ini membentuk fondasi yang kuat untuk mengotomatiskan pembersihan data dalam alur kerja pembelajaran mesin di dunia nyata. Dengan menggabungkan ekosistem pustaka Python yang kaya dengan praktik terbaik dalam desain modular, para praktisi dapat membangun pipeline yang tidak hanya efisien dan mudah dipelihara, tetapi juga selaras dengan standar kualitas data dan tujuan analitis.

5.8 PERTANYAAN

1. Jelaskan pentingnya membangun pipeline otomatis untuk pembersihan data. Bagaimana fungsi modular dan dapat digunakan kembali meningkatkan efisiensi dalam proses ini?
2. Diskusikan peran fungsi seperti *getFirstLook*, *getMissing*, *getDisports*, dan *getOutliers* dalam menilai dan meningkatkan kualitas data. Bagaimana fungsi-fungsi ini berkontribusi pada desain pipeline yang efektif?
3. Apa langkah-langkah kunci yang terlibat dalam memeriksa kualitas data secara keseluruhan menggunakan fungsi seperti *checkCats*, *checkOutliers*, dan *runChecks*? Berikan contoh dari dataset NLS untuk mengilustrasikan jawaban Anda.
4. Dalam kelas *OutlierTrans* yang digunakan dalam pipeline, bagaimana outlier ditangani?
 - a. Outlier dibatasi pada nilai rata-rata
 - b. Nilai outlier diganti dengan nilai NaN
 - c. Nilai outlier dihilangkan dari dataset
 - d. Nilai outlier diabaikan selama pelatihan model
5. Fungsi scikit-learn manakah yang digunakan untuk menggabungkan beberapa langkah pra-pemrosesan dan model ke dalam satu proses sekuensial?
 - a. `cross_validate`
 - b. `make_pipeline`
 - c. `StandardScaler`
 - d. `SimpleImputer`
6. [Latihan Pemrograman – Dasar] Implementasikan fungsi 'missing' di Python untuk menghitung nilai yang hilang berdasarkan baris dan kolom dalam dataset yang diberikan. Uji fungsi tersebut pada DataFrame kecil yang berisi sedikit nilai *NaN*.
7. [Latihan Pemrograman – Lanjutan] Buat pipeline scikit-learn untuk dataset yang mencakup langkah-langkah berikut:
 - Standardisasi fitur numerik
 - Penanganan outlier menggunakan transformer khusus
 - Mengisi nilai yang hilang dengan rata-rata
 - Melatih model regresi linier

Evaluasi pipeline menggunakan validasi silang 5-fold.

BAB 6

KECERDASAN KAWANAN

6.1 DEFINISI KECERDASAN KAWANAN

Kecerdasan Kawan (Swarm Intelligence/SI) mengacu pada kelas algoritma dan konsep yang memodelkan perilaku kelompok cerdas untuk mencapai tujuan bersama. Dalam sistem SI, agen individu – yang disebut boid – beroperasi dalam kelompok yang disebut kawan. Setiap boid berkontribusi pada munculnya perilaku kolektif yang cerdas melalui interaksi lokal, meskipun memiliki kemampuan individu yang terbatas. Boid ini berfungsi secara asinkron dan tanpa kendali terpusat. Kecerdasan mereka muncul dari pengambilan keputusan paralel yang terdesentralisasi dan interaksi dengan lingkungan mereka. Meskipun setiap boid mungkin tidak sepenuhnya memahami tugas secara keseluruhan, upaya gabungan mereka menghasilkan pemecahan masalah yang efektif dan perilaku global yang adaptif. Konsep SI terinspirasi oleh perilaku serangga sosial seperti semut, lebah, dan rayap. Spesies ini menunjukkan koordinasi yang luar biasa dalam tugas-tugas seperti mencari makan dan membangun sarang, semuanya tanpa pengawasan langsung atau sistem saraf yang kompleks. Keberhasilan kelompok diprioritaskan daripada kinerja individu, menjadikannya model ideal untuk mempelajari kecerdasan kolektif.

SI (Smart Intelligence) mulai dikenal luas dalam pengembangan AI pada akhir tahun 1980-an, khususnya dalam konteks sistem robot seluler. Seiring dengan meningkatnya volume data dan kompleksitas sistem, SI telah muncul sebagai alternatif yang menjanjikan untuk membangun sistem cerdas dan terdistribusi. Sistem SI dicirikan oleh tiga prinsip dasar: *pembagian kerja*, *stigmergi*, dan *pengaturan diri* (self-organization/SO). Pada bagian selanjutnya, kita akan mengeksplorasi masing-masing prinsip ini secara detail.

6.2 KARAKTERISTIK DASAR SI

Sistem SI dibangun di atas tiga prinsip dasar yang memungkinkan agen terdesentralisasi (boid) untuk memecahkan masalah kompleks secara kolaboratif:

1. Pembagian Kerja
2. Stigmergi
3. Pengaturan Diri (Self-Organization/SO)

Bersama-sama, prinsip-prinsip ini mendorong kecerdasan adaptif dan pemecahan masalah yang efisien.

- **Pembagian Kerja:** Dalam sistem SI, setiap boid menjalankan peran khusus yang berkontribusi pada tujuan keseluruhan kawan. Tugas-tugas yang terlalu kompleks untuk dilakukan oleh satu boid (burung pipit) diselesaikan melalui upaya kelompok yang terkoordinasi. Prinsip ini terinspirasi oleh serangga sosial, seperti lebah, di mana individu mengambil peran seperti pejantan atau ratu berdasarkan isyarat lingkungan dan kemampuan internal.

Distribusi peran adaptif ini memungkinkan kawanan untuk merespons secara dinamis terhadap perubahan kondisi. Misalnya, ketika makanan menjadi langka, lebih banyak boid dapat ditugaskan untuk mencari makan. Dengan membagi tugas dan melaksanakannya secara paralel, kawanan meningkatkan efisiensi operasional, mengurangi konsumsi sumber daya, dan mempercepat penyelesaian tugas. Kemunculan perilaku global muncul dari interaksi lokal – boid berkoordinasi berdasarkan aturan sederhana dan umpan balik lingkungan, tanpa kontrol terpusat.

- **Stigmergi:** Stigmergi mengacu pada koordinasi tidak langsung di antara boid melalui isyarat lingkungan daripada komunikasi langsung. Misalnya, semut meninggalkan jejak feromon untuk memandu yang lain menuju sumber makanan. Umpan balik positif memperkuat jalur yang berhasil, sementara umpan balik negatif mencegah ketergantungan berlebihan pada jalur yang sudah usang.

Respons adaptif terhadap perubahan lingkungan dilakukan secara otonom sambil mempertahankan efisiensi dalam keadaan yang berubah. Boid mencari rute baru atau solusi baru sambil tetap menandai lingkungan sekitar mereka dengan tanda untuk mengarahkan boid lain. Lingkaran umpan balik merupakan stigmergi yang mengintegrasikan komentar yang menguntungkan dan tidak menguntungkan. Umpan balik negatif mencegah ketergantungan berlebihan pada jalur yang sudah usang, sementara umpan balik positif meningkatkan jalur yang efektif. Koordinasi tidak langsung memungkinkan kecerdasan kolektif, dengan setiap boid berkontribusi pada perilaku kawanan global tetapi hanya berinteraksi dengan informasi lokal. Hal ini memungkinkan boid untuk berkolaborasi secara tidak langsung dan tanpa kendali terpusat untuk mencapai tujuan bersama.

- **Pengorganisasian Diri (SO):** SO adalah salah satu fitur paling penting dari sistem SI. SO membangun kolaborasi mendasar antara boid SI untuk mencapai perilaku kolektif yang diinginkan. Boid memanfaatkan perilaku yang muncul dari interaksi lokal dengan yang lain untuk mengatasi berbagai masalah, sehingga meningkatkan efektivitas Sistem SI. Interaksi ini bergantung pada seperangkat aturan yang diterapkan sesuai dengan lingkungan atau konteks lokal tempat mereka beroperasi.

Tidak ada pola atau perilaku global yang disadari oleh boid-boid ini. Tindakan individu setiap boid menghasilkan perilaku global. Penting untuk diingat bahwa perilaku lokal boid tidak diatur oleh entitas eksternal apa pun. Singkatnya, kapasitas SO dari boid individu, yaitu, cakupan fungsionalnya, terbatas pada lingkungan lokal. Bekerja sama di antara mereka sendiri adalah apa yang memungkinkan aktivitas kelompok global dalam sistem SI apa pun. SO memiliki empat komponen fundamental yang tercantum di bawah ini :

- **Umpan Balik Positif:** Di sini mari kita ambil contoh sarang lebah. Kita dapat mengatakan bahwa semakin banyak nektar yang dikumpulkan lebah, semakin besar penekanan sumber daya keseluruhan kawanan. Penguatan positif atau umpan balik positif membantu boid dalam kawanan untuk menemukan solusi terbaik.

- **Umpan Balik Negatif:** Umpan balik negatif adalah mekanisme penting untuk stabilisasi dan mencegah konvergensi yang terlalu cepat ke satu solusi. Mekanisme ini membantu kawanan dalam mempertahankan keragaman dan menghindari terjebak dalam optimasi lokal dengan mengurangi pengaruh perilaku atau keadaan yang tidak berhasil. Dengan menggunakan mekanisme ini, kita dapat menghentikan boid dari terus-menerus mengikuti solusi yang kurang ideal dan memastikan mereka tidak menyadari rute yang sudah usang. Baik umpan balik positif maupun negatif dapat digunakan secara bersamaan untuk menjaga keseimbangan dan menghindari penggunaan berlebihan.
- **Fluktuasi Perilaku Acak:** Fluktuasi perilaku acak sangat penting bagi sistem SI untuk mengeksplorasi ruang solusi secara efektif dan untuk menghindari kemungkinan terjebak dalam optimasi lokal. Fluktuasi tersebut sering diperkenalkan sebagai gerakan acak atau sebagai variasi dalam perilaku boid, sehingga memungkinkan kawanan untuk menemukan solusi baru dan berpotensi lebih baik yang mungkin terlewatkan ketika menggunakan pendekatan deterministik murni.
- **Interaksi Ganda di antara Boid:** Dalam sistem SI, boid berinteraksi satu sama lain melalui aturan sederhana dan terlokalisasi yang mengarah pada perilaku kelompok yang kompleks dan adaptif. Interaksi ini dapat berupa langsung atau tidak langsung (melalui lingkungan). Kuncinya adalah bahwa boid individu tidak perlu memahami tujuan keseluruhan; interaksi lokal mereka, ditambah dengan elemen stokastik, memungkinkan kawanan untuk mencapai optimasi global.

Mekanisme-mekanisme ini secara kolektif memungkinkan sistem SI untuk beradaptasi, mengoptimalkan, dan tetap tangguh dalam lingkungan yang dinamis. Keuntungan sistem SI meliputi :

- **Ketangguhan:** Sistem SI tangguh terhadap kegagalan komponen individual, sehingga cocok untuk lingkungan yang tidak dapat diprediksi. Kelompok boid tidak terpengaruh oleh kegagalan satu boid, dan tujuan keseluruhan kemungkinan besar tercapai.
- **Skalabilitas:** Algoritma SI dapat beradaptasi dengan banyak boid atau masalah kompleks, sehingga cocok untuk berbagai aplikasi.
- **Pengorganisasian diri:** Boid dalam sistem SI memodifikasi dan beradaptasi dengan lingkungan sekitarnya dan muncul dengan sendirinya tanpa perencanaan sebelumnya.
- **Adaptasi:** Karena boid tidak dikendalikan secara terpusat, mereka memiliki kemampuan khusus untuk beradaptasi dengan perubahan lingkungan.
- **Kelincahan dan Kecepatan:** Dalam sistem SI, boid menunjukkan bakat dan peningkatan kecepatan di lingkungan sekitarnya dan potensi interaksi dengan sistem.
- **Fleksibilitas:** Algoritma SI dapat digunakan untuk menyelesaikan berbagai macam masalah, termasuk optimasi, pengenalan pola, dan pengambilan keputusan. Berkaitan dengan boid, mereka bereaksi dengan tepat terhadap perubahan lingkungan, dan populasi secara keseluruhan menunjukkan kemampuan beradaptasi.

- **Desentralisasi:** Tidak adanya pengontrol terpusat dalam sistem SI mengurangi risiko titik kegagalan tunggal dan menyederhanakan desain sistem.
- **Pengorganisasian Diri:** Sistem SI dapat beradaptasi dan mengatur ulang diri mereka sendiri berdasarkan interaksi lokal, sehingga memungkinkan mereka untuk merespons perubahan lingkungan.

Namun, sistem SI menghadirkan tantangan seperti kompleksitas desain dan potensi mereka untuk terjebak pada optimasi lokal. Tantangan dalam sistem SI meliputi :

- **Kompleksitas:** Mendesain dan menganalisis sistem SI dapat menjadi tugas yang kompleks dan akan membutuhkan penggunaan pengetahuan dan alat khusus.
- **Optimasi Lokal:** Algoritma SI seringkali terjebak pada optimasi lokal, sehingga gagal menemukan solusi terbaik secara global.
- **Penyesuaian Parameter:** Parameter algoritma SI dapat dioptimalkan untuk mendapatkan solusi global yang andal. Namun, penyesuaian parameter memerlukan eksperimen yang cermat.
- **Biaya Komputasi:** Untuk menemukan solusi terbaik global, algoritma SI mungkin membutuhkan sumber daya komputasi yang lebih tinggi, sehingga menimbulkan biaya yang lebih tinggi.
- **Transparansi Terbatas:** Dalam lingkungan yang kompleks, memahami dan memprediksi perilaku sistem SI dapat menjadi sulit dan memakan waktu, sehingga memberikan transparansi yang terbatas.
- **Sensitivitas terhadap Gangguan:** Algoritma SI seringkali sensitif terhadap gangguan eksternal dan/atau ketidakpastian, sehingga memengaruhi stabilitas dan kinerjanya.

6.3 PEDOMAN DESAIN UNTUK SISTEM SWARM

Untuk membangun sistem SI yang andal dan efisien, prinsip-prinsip desain tertentu harus diintegrasikan. Prinsip-prinsip ini memastikan bahwa agen-agen terdesentralisasi dapat berkolaborasi secara efektif sambil mempertahankan kemampuan beradaptasi dan ketahanan dalam lingkungan yang dinamis :

- **Prinsip Kedekatan:** Prinsip ini menetapkan batasan logis atau geografis untuk komunikasi yang efisien antar boid. Prinsip ini memastikan bahwa berbagi informasi terjadi dalam lingkup yang dapat dikelola, mengurangi biaya operasional dan meningkatkan daya tanggap.
- **Prinsip Kualitas:** Boid harus menjelajahi ruang pencarian secara independen sambil secara kolektif mengevaluasi kualitas solusi. Tanpa otoritas terpusat, kawanan memprioritaskan solusi yang unggul menggunakan isyarat lingkungan, memastikan bahwa keputusan kelompok selaras dengan hasil yang optimal.
- **Prinsip Respons Beragam:** Untuk mempertahankan ketahanan dan kemampuan beradaptasi, boid harus menjelajahi area yang luas dari ruang pencarian. Prinsip ini mencegah konvergensi prematur dan mendukung pemecahan masalah yang kuat dengan memanfaatkan keragaman dalam strategi eksplorasi.

- **Prinsip Kemampuan Beradaptasi:** Dalam lingkungan yang cepat berubah, sistem SI harus tetap fungsional dan efisien tanpa intervensi eksternal. Boid secara dinamis beradaptasi dengan perubahan lingkungan dan mengatur ulang diri mereka sendiri berdasarkan kondisi baru, memastikan kinerja yang berkelanjutan.

Prinsip-prinsip ini membentuk dasar untuk merancang sistem SI yang dapat *diskalakan*, *tangguh*, dan *adaptif*, sehingga memungkinkan sistem tersebut untuk menangani tugas-tugas optimasi dan pengambilan keputusan yang kompleks secara efektif. Pada bagian selanjutnya, kita akan memperkenalkan dan membahas model (algoritma) Optimasi Swarm Partikel (PSO).

6.4 OPTIMASI KAWANAN PARTIKEL (PSO)

Algoritma PSO memodelkan perilaku bergerombol dan berkelompok burung dan ikan untuk menyelesaikan masalah optimasi. Tujuannya adalah untuk memanfaatkan perilaku kolektif untuk menemukan solusi terbaik dalam ruang pencarian dinamis.

- **Fitur Utama Model PSO:** Terdapat beberapa fitur utama model PSO, termasuk :
 - **Tujuan:** Menemukan solusi optimal dalam ruang pencarian multidimensi menggunakan strategi eksplorasi dan eksploitasi.
 - **Perilaku Boid:** Setiap partikel (boid) menjaga jarak aman dari partikel lain untuk menghindari tabrakan sambil menyesuaikan posisi dan kecepatannya berdasarkan informasi lokal dan global.
 - **Komunikasi:** Partikel berbagi informasi posisi dan kecepatan dengan tetangga, memengaruhi pergerakan menuju solusi yang lebih baik.
 - **Adaptabilitas:** Menggabungkan keacakan (R_1 , R_2) dan parameter kognitif-sosial (c_1 , c_2) untuk menyeimbangkan eksplorasi dan eksploitasi.

PSO dapat menangani ruang pencarian *kontinu*, *biner*, atau *diskrit*, sehingga serbaguna untuk berbagai tugas optimasi.

- **Algoritma PSO:** Deskripsi algoritma PSO telah diadopsi dari Boid dalam algoritma PSO pertama-tama diinisialisasi ke titik acak dan mereka bergerak dengan kecepatan acak. Aktivitas kolektif mendorong proses pencarian saat boid menyisir seluruh ruang pencarian. Pergerakan boid yang terkoordinasi adalah fitur mendasar dari PSO, yang memaksimalkan efisiensi pencarian. Pada saat yang sama, hal ini memastikan bahwa upaya yang dibutuhkan untuk konvergen ke solusi terbaik adalah minimal.

Untuk menjamin pencarian dan konvergensi yang efisien, boid dalam kawanan PSO mematuhi dua prinsip panduan, yaitu *pemisahan* dan *penyelarasan*. Untuk menghindari tabrakan dan menjaga wilayahnya sendiri di dalam formasi, setiap boid diprogram untuk menjaga jarak aman dari tetangganya. Untuk mempertahankan pemisahan saat mencari solusi terbaik, boid memodifikasi posisi dan kecepatannya dengan mengikuti posisi dan kecepatan tetangga terdekatnya. Di sisi lain, setiap boid mencocokkan kecepatan kelompok rata-rata di ruang pencarian dan mengoordinasikan pergerakannya dengan pola keseluruhan boid tetangganya.

Setiap anggota boid terus-menerus mengkomunikasikan posisi dan kecepatannya kepada kelompok, lebih tepatnya, kepada tetangga terdekatnya. Setiap boid

mengamati bagaimana boid lain di sekitarnya berperilaku dan belajar. Berdasarkan pengamatan ini, boid dapat memodifikasi pergerakannya sendiri atau dapat memengaruhi pergerakan (posisi dan kecepatan) boid tetangga jika pengalamannya terbukti menguntungkan untuk optimasi lokal. Tujuan boid adalah untuk menyelaraskan diri dengan tetangga terdekatnya, yang menjamin bahwa komunitas boid (swarm) bekerja sama untuk mencapai tujuan. Persamaan pembaruan kecepatan dan posisi, yang menentukan bagaimana boid harus memodifikasi gerakannya untuk konvergen menuju solusi ideal, mengontrol algoritma PSO (lihat persamaan 6.1 dan 6.2).

Algoritma PSO memperbarui kecepatan dan posisi partikel menggunakan dua persamaan:

$$v_{id}^{t+1} = v_{id}^t + c_1 R_1 (p_{id}^t - x_{id}^t) + c_2 R_2 (p_{gd}^t - x_{id}^t) \quad (6.1)$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \quad (6.2)$$

di mana:

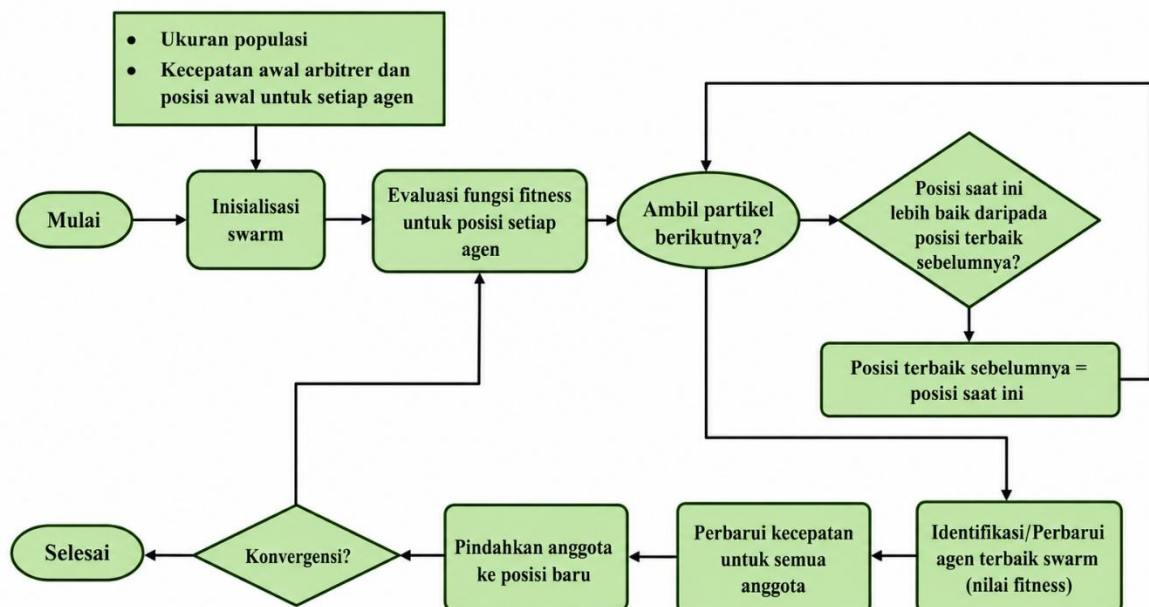
- $v_{id}^{(t+1)}$: Kecepatan partikel i pada dimensi d dalam iterasi $t + 1$
- $x_{id}^{(t+1)}$: Posisi partikel yang diperbarui
- p_{id}^t : Posisi terbaik yang ditemukan oleh partikel i
- p_{gd}^t : Posisi terbaik global dalam kawanan
- c_1, c_2 : Koefisien kognitif dan sosial
- R_1, R_2 : Faktor acak yang memperkenalkan stokastisitas

Di sini, dua karakteristik penting diperlukan untuk menentukan kecepatan boid tunggal. Posisi terbaik yang telah dicapai boid individu dari waktu ke waktu diwakili oleh $p_{id}(t)$ dan $p_{gd}(t)$, yang mewakili posisi boid terbaik dalam posisi global kawanan terhadap lingkungan. Ketika kedua parameter ini memengaruhi kecepatan keseluruhan kawanan, hal itu menghasilkan kecepatan terbaik untuk menemukan solusi dalam ruang yang diberikan, dengan asumsi lingkungan dapat diprediksi. Dalam lingkungan stokastik, variabel tambahan, R_1 dan R_2 , digunakan oleh boid untuk membantu beradaptasi dengan perubahan keadaan lingkungan. Parameter ini menambahkan keacakan yang diperlukan untuk meningkatkan eksplorasi ruang pencarian secara efisien. Relevansi posisi optimal boid individu dalam kaitannya dengan posisi optimal kawanan ditunjukkan oleh parameter c_1 dan c_2 . Di sini, c_1 adalah parameter kognitif dan c_2 adalah parameter sosial. Bahkan ketika lingkungan berubah, nilai parameter ini terus mengarahkan setiap boid menuju lokasi terbaik dalam kawanan.

Di sini, kita akan membahas secara singkat dampak perbedaan relatif antara c_1 dan c_2 . Dalam algoritma optimasi, parameter c_1 dan c_2 memainkan peran penting dalam menyeimbangkan eksplorasi dan eksploitasi, sehingga menentukan perilaku pencarian boid individu. "Tingkat eksplorasi" dapat dipengaruhi secara kritis oleh koefisien ini. Ketika c_1 (komponen kognitif, yang memengaruhi memori partikel tentang posisi terbaiknya sendiri) dan c_2 (komponen sosial, yang memengaruhi daya tarik partikel terhadap posisi terbaik global kelompok) keduanya diatur ke nilai tinggi, sistem menunjukkan tingkat eksplorasi yang tinggi

di wilayah yang jauh. Hal ini mendorong partikel untuk sering menyimpang dari jalur mereka saat ini, didorong kuat oleh keberhasilan individu di masa lalu dan temuan terbaik kolektif, sehingga memfasilitasi pencarian yang luas. Sebaliknya, jika c_1 dan c_2 dikonfigurasi ke nilai kecil, algoritma cenderung ke arah pencarian yang lebih halus dan tingkat eksplorasi yang lebih rendah. Dalam skenario ini, partikel kurang dipengaruhi oleh keberhasilan terbaik mereka di masa lalu atau keberhasilan terbaik global boid, yang menyebabkan pergerakan yang lebih terlokalisasi dan fokus pada penyempurnaan solusi di sekitar mereka. Bias spesifik muncul ketika parameter ini asimetris: c_1 yang tinggi ditambah dengan c_2 yang kecil menghasilkan bias terhadap posisi terbaik global boid tertentu.

Di sini, kinerja historis individu sangat memandu pencarian, berpotensi menyebabkan konvergensi cepat ke optimum lokal jika hasil terbaik individu tidak optimal secara global. Sebaliknya, ketika c_1 kecil dan c_2 tinggi, pencarian menunjukkan bias yang kuat terhadap posisi terbaik global kelompok, membuat partikel sangat rentan untuk mengikuti temuan terbaik kolektif, yang dapat meningkatkan konvergensi tetapi mungkin mengurangi eksplorasi wilayah baru jika hasil terbaik boid tidak benar-benar global. Memahami dinamika ini sangat penting untuk menyetel algoritma tersebut secara efektif untuk mencapai kinerja optimasi yang diinginkan.



Gambar 6.1 Algoritma PSO digambarkan sebagai diagram alir.

Algoritma PSO dapat diringkas menggunakan diagram alir seperti yang ditunjukkan pada Gambar 6.1. Merujuk pada persamaan 6.1 dan 6.2, terdapat tiga komponen berbeda dari fungsi kecepatan seperti yang dibahas di bawah ini.

- **Inersia:** Ketahanan suatu benda terhadap perubahan apa pun dalam keadaan geraknya dikenal sebagai inersia, dinotasikan sebagai (v_{idt}) . Ekspresi ini melacak perubahan dalam arah, kecepatan, atau keadaan diam boid. Waktu t menentukan kecepatan pada waktu $t + 1$. Sebuah boid tidak dapat mengubah kecepatannya secara tiba-tiba.

Sebaliknya, kecepatannya berubah secara bertahap berdasarkan perubahan di lingkungan atau jika boid harus mengubah kecepatannya untuk menelusuri ruang pencarian secara efisien. Karena kecepatan boid pada waktu $t + 1$ bergantung pada kecepatan boid pada waktu t , kita mendapati bahwa komunitas boid bergerak dalam suatu formasi dan sebagian besar tetap berada pada arah yang sama karena inersia ini. Suku ini juga sangat krusial untuk mengubah boid terbaik global di dalam kawanan. Ketika fungsi kebugaran boid lebih optimal dibandingkan dengan kebugaran keseluruhan (global) kawanan tersebut, boid itu mengambil posisi sebagai boid terbaik global di dalam kawanan. Selama transisi ($p_i = x_i = p_g$), ekspresi sosial dalam persamaan tersebut menjadi nol ($p_{id(t)} - x_{id(t)} = p_{ig(t)} - x_{id(t)} = 0$). Pada titik ini, boid baru tersebut menjadi boid terbaik global dengan bergerak menggunakan kecepatan baru dan dengan demikian mengubah posisinya di dalam kawanan.

- **Pengetahuan diri:** Komponen fungsi kecepatan ini mengkarakterisasi individualitas sebuah boid di dalam kawanan dan diberikan oleh ekspresi ($p_{id(t)} - x_{id(t)}$). Akibatnya, sebuah boid lebih tertarik pada nilai terbaik globalnya sendiri, yang memaksimalkan pencarian di seluruh ruang solusi.
- **Pengetahuan Sosial:** Komponen fungsi kecepatan ini mendeskripsikan bagaimana setiap boid menyesuaikan diri dengan aktivitas sosial dan diberikan oleh ekspresi ($p_{ig(t)} - x_{id(t)}$). Pernyataan ini mendefinisikan tingkat berbagi pengalaman dan pembelajaran kelompok di antara anggota-anggota individu.

Untuk mendemonstrasikan penerapan algoritma PSO, kita akan fokus pada rekayasa fitur untuk aplikasi Pembelajaran Mesin/Kecerdasan Buatan (ML/AI). Tujuan kita adalah untuk mengidentifikasi fitur-fitur signifikan dalam dataset ini. Skrip Python yang disediakan di sini (lihat Tabel 6.1) diadopsi dari. Dataset pinjaman memiliki total 614 tuple di 12 fitur. Tidak ada fitur yang memiliki nilai yang hilang. Perhatikan di sini bahwa dalam skrip Python, boids telah disebut sebagai partikel. Metode PSO yang digunakan di sini mencoba menemukan solusi terbaik untuk suatu masalah dengan membuat tebakan yang lebih baik secara bertahap. Sebelumnya dalam bab ini, telah ditunjukkan bahwa PSO memecahkan masalah dengan menggunakan sekelompok solusi yang mungkin, yang disebut boids.

Boid-boid ini bergerak di sekitar ruang solusi, menyesuaikan posisi dan kecepatan mereka menggunakan rumus matematika sederhana untuk menemukan jawaban yang lebih baik. Setiap boid bergerak berdasarkan posisi terbaiknya sendiri yang diketahui sejauh ini dan juga tertarik pada posisi terbaik yang ditemukan oleh boid lain. Posisi terbaik mereka sering diperbarui setiap kali solusi yang lebih baik ditemukan selama pencarian. Garis besar sederhana diberikan dalam untuk menunjukkan bagaimana algoritma PSO dapat digunakan untuk pemilihan fitur. Kita akan melihat bagaimana hyperparameter berikut dapat disetel dan digunakan sebaik-baiknya untuk memungkinkan PSO mengidentifikasi fitur-fitur kunci dalam dataset. Setiap boid dapat digunakan untuk mewakili kumpulan fitur yang dipilih secara acak (subset fitur) dalam dataset. Kita dapat membayangkan boid mewakili vektor biner, di mana

setiap elemen sesuai dengan fitur (1 jika fitur dipilih dan 0 jika tidak). Fungsi objektif dapat digunakan untuk memeriksa seberapa baik kumpulan fitur yang dipilih.

Sebagai permulaan, kita secara acak menetapkan posisi dan kecepatan awal semua boid dalam swarm. Untuk pemilihan fitur, ini berarti setiap boid mewakili sekumpulan fitur yang dipilih secara acak, yaitu, subset fitur. Di setiap putaran, setiap boid memperbarui kecepatan dan posisinya berdasarkan dua hal: solusi terbaik yang telah ditemukannya sejauh ini, dan solusi terbaik yang ditemukan oleh seluruh kelompok. Setelah bergerak, fitur setiap boid diuji menggunakan fungsi objektif. Jika hasilnya lebih baik, boid memperbarui solusi terbaik yang diketahuinya. Boid lain juga memperbarui posisi dan kecepatannya berdasarkan solusi terbaik secara keseluruhan jika diperlukan. Langkah-langkah pembaruan dan evaluasi diulang hingga batas yang ditetapkan tercapai atau hasilnya meningkat secara signifikan pada putaran saat ini dibandingkan dengan putaran sebelumnya. Setelah algoritma PSO selesai (setelah menjalankan skrip Python pada Tabel 6.1), fitur yang dipilih adalah *tanggungan*, *pendidikan*, *wiraswasta*, dan *riwayat kredit*.

TABEL 6.1 Skrip Python untuk Mengidentifikasi Fitur Signifikan Menggunakan Algoritma

```
# install the package pyswarms as an initial step
pip install pyswarms

import pandas as pd
# Define the file path
file_path = '/kaggle/input/loan-data-set/loan_train.csv'
# Load the CSV file into a DataFrame
df = pd.read_csv(file_path)
df

# Drop rows with null values
df_cleaned = df.dropna()
# Check for null values
null_values = df_cleaned.isnull().sum()
# Display the null value counts
print("Null value counts:")
print(null_values)

from sklearn.preprocessing import LabelEncoder
# Create a LabelEncoder object
label_encoder = LabelEncoder()
# Apply label encoding to categorical columns
categorical_columns = ['Gender', 'Married', 'Dependents',
                        'Education', 'Self_Employed', 'Area', 'Status']
for column in categorical_columns:
    df_cleaned[column] = label_encoder.fit_transform(df_cleaned[column])
# Display the DataFrame after label encoding
print(df_cleaned.head())

import numpy as np
import pyswarms as ps
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```
X = df_cleaned.drop('Status', axis=1).values # Feature matrix
y = df_cleaned['Status'].values # Target variable
# Define binary PSO
options = {'c1': 0.5, 'c2': 0.3, 'w':0.9}
# Define bounds
max_bound = 1.0 * np.ones(X.shape[1])
min_bound = 0.0 * np.ones(X.shape[1])
bounds = (min_bound, max_bound)
# Objective function
def f_per_particle(m, alpha):
    """Computes for the objective function per particle
    Inputs
    -----
    m : numpy.ndarray
        Binary mask that can be obtained from BinaryPSO, will
        be used to mask features.
    alpha: float
        Penalty factor to control the number of features selected.
    Returns
    -----
    numpy.ndarray
    """
    total_features = X.shape
    # Apply mask to features
    X_subset = X[:,m>0.5]
    if X_subset.shape[1] == 0:
        return float('inf')
    # Split the dataset
    X_train, X_test, y_train, y_test = train_test_split(X_subset, y,
test_size=0.3, random_state=42)
    # Fit the model
    clf = RandomForestClassifier(n_estimators=50, random_state=42)
    clf.fit(X_train, y_train) # Predict and calculate accuracy
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    # Calculate objective
    j = (alpha * (1.0 - accuracy) + (1.0 - alpha) * (1 - (X_subset.
shape[1] / total_features)))
    return j
# Define objective function
def f(x, alpha=0.5):
    """Higher-level method to do classification in the
    whole swarm.
    Inputs
    -----
    x: numpy.ndarray of shape (n_particles, dimensions)
        The swarm that will perform the search
    Returns
    -----
    numpy.ndarray of shape (n_particles, )
        The computed loss for each particle
    """
    n_particles = x.shape
    j = [f_per_particle(x[i], alpha) for i in range(n_particles)]
    return np.array(j)
# Initialize swarm
optimizer = ps.single.GlobalBestPSO(n_particles=50, dimensions=X.
shape, options=options, bounds=bounds)
```

```
# Perform optimization
cost, pos = optimizer.optimize(f, iters=100)

#'pos' is the variable containing the best position returned by
optimizer.optimize()
best_pos = np.array(pos)
print("Position as NumPy Array:", best_pos)

# Define a threshold to determine whether a feature is selected
threshold = 0.6
# Create a boolean mask from 'best_pos' based on the threshold
selected_features_mask = best_pos > threshold
# df_cleaned.columns[:-1] gives you all feature names excluding the
target variable
feature_names = df_cleaned.columns[:-1] # Modify this as necessary
# Use the mask to select the names of the features
selected_features = feature_names[selected_features_mask]
print("Selected Features:", selected_features)
```

SI (Self-Organizing Systems) adalah konsep yang ampuh dan memiliki beberapa aplikasi di bidang ML/AI, menawarkan solusi unik untuk masalah kompleks. Prinsip-prinsip desentralisasi, SO (Self-Organizing), dan kemunculannya menjadikannya alat yang tangguh, terukur, dan fleksibel untuk memecahkan masalah dunia nyata.

6.5 RINGKASAN

Dalam bab ini, kita telah mengeksplorasi SI – sebuah paradigma yang terinspirasi oleh perilaku kolektif dari sistem yang terdesentralisasi dan mengatur diri sendiri – dan aplikasinya dalam ML dan AI. Kita mulai dengan memperkenalkan konsep SI dan inspirasi biologisnya dari serangga sosial seperti semut dan lebah. Sistem SI bergantung pada tiga prinsip dasar: *pembagian kerja*, *stigmergi*, dan *SO*, yang memungkinkan agen sederhana (*boid*) untuk berkolaborasi dan menghasilkan perilaku global yang kompleks tanpa kendali terpusat. Selanjutnya, kita meneliti keuntungan SI – ketangguhan, skalabilitas, fleksibilitas, dan adaptabilitas – bersama dengan tantangan utama, termasuk kompleksitas desain, kerentanan terhadap optimasi lokal, dan biaya komputasi. Kita juga membahas pedoman desain seperti kedekatan, kualitas, respons yang beragam, dan prinsip adaptabilitas, yang memastikan sistem SI yang andal dan efisien.

Bab ini kemudian berfokus pada PSO, salah satu algoritma SI yang paling banyak digunakan. Kami menjelaskan fitur-fitur utamanya, persamaan pembaruan kecepatan dan posisi, dan peran parameter kognitif dan sosial dalam menyeimbangkan eksplorasi dan eksploitasi. Terakhir, kami mendemonstrasikan aplikasi praktis PSO dalam *pemilihan fitur untuk model ML*, menggunakan implementasi Python pada dataset dunia nyata. Dengan memanfaatkan prinsip-prinsip SI, algoritma seperti PSO menawarkan solusi yang ampuh untuk optimasi dan pengambilan keputusan di lingkungan terdistribusi. Konsep-konsep ini memberikan dasar untuk membangun sistem adaptif dan terukur yang mampu mengatasi tantangan kompleks dalam aplikasi AI dan Big Data modern.

6.6 PERTANYAAN

1. Jelaskan konsep Kecerdasan Kawan (Swarm Intelligence/SI) dan analisis prinsip-prinsip dasarnya, berikan contoh yang tepat untuk mendukung penjelasan Anda.
2. Evaluasi secara kritis keuntungan dan tantangan yang terkait dengan sistem SI. Sajikan minimal dua keuntungan dan dua tantangan dalam jawaban Anda.
3. Jelaskan konsep stigmergi dalam sistem SI, ilustrasikan dengan contoh yang relevan. Analisis bagaimana stigmergi berkontribusi pada munculnya kecerdasan kolektif.
4. Manakah dari berikut ini yang BUKAN dianggap sebagai karakteristik mendasar dari sistem SI?
 - a. Pembagian Kerja
 - b. Stigmergi
 - c. Pengorganisasian Diri
 - d. Pengawasan Terpusat
5. Dalam konteks Optimasi Swarm Partikel (PSO), parameter c_1 dan c_2 mewakili:
 - a. Parameter Kognitif dan Sosial
 - b. Parameter Kecepatan dan Posisi
 - c. Parameter Kebugaran dan Keacakan
 - d. Parameter Inersia dan Penyelesaian
6. [Latihan Pemrograman] Ubah skrip Python PSO untuk menampilkan skor akurasi RandomForestClassifier setelah setiap iterasi.
7. [Latihan Pemrograman Tingkat Lanjut] Dengan menggunakan pendekatan PSO yang dijelaskan dalam bab ini, kembangkan program untuk melakukan pemilihan fitur pada dataset pilihan Anda. Bandingkan dan analisis akurasi klasifikasi sebelum dan sesudah proses pemilihan fitur.

BAB 7

PEMROGRAMAN GENETIK

7.1 DEFINISI PEMROGRAMAN GENETIK

Pemrograman Genetik (GP) adalah paradigma pemrograman evolusioner yang banyak digunakan dalam kecerdasan buatan dan penambangan data. Di era Big Data, teknik analisis tradisional seringkali kesulitan dengan efisiensi dan skalabilitas, yang menyebabkan pola yang terlewatkan dan kinerja yang menurun. GP mengatasi tantangan ini dengan memanfaatkan *Algoritma Genetik (GA)* – metode optimasi yang terinspirasi dari alam yang mengembangkan solusi selama beberapa generasi berturut-turut. GP meniru *seleksi alam*, mekanisme kunci evolusi biologis di mana individu dengan sifat yang menguntungkan bertahan hidup dan bereproduksi. Prinsip ini mendorong penyempurnaan iteratif dari solusi kandidat dalam GP. Sama seperti jerapah dengan leher yang lebih panjang secara bertahap mendominasi populasi mereka karena akses yang lebih baik ke makanan, GP mengembangkan program dengan mengutamakan program dengan *kebugaran* yang lebih tinggi – ukuran kinerja relatif terhadap masalah yang sedang dipecahkan. Proses dimulai dengan populasi program acak dan memperbaikinya melalui *operator genetik* seperti *crossover* (menggabungkan bagian dari dua program induk) dan *mutasi* (memperkenalkan perubahan acak untuk keragaman). Selama beberapa generasi, operasi ini menghasilkan solusi yang semakin efektif. Pada bagian berikut, kita akan mengeksplorasi prinsip-prinsip inti GP, terminologinya, dan aplikasi praktis dalam optimasi, pembelajaran mesin, dan sistem dunia nyata.

7.2 KONSEP DASAR

GP dan GA didasarkan pada prinsip-prinsip evolusi yang meniru seleksi alam. Konsep-konsep ini memungkinkan algoritma untuk mengembangkan solusi secara iteratif, meningkatkan kinerja dari generasi ke generasi. Berikut adalah komponen-komponen kuncinya :

- **Pendekatan Evolusi:** Algoritma GP mensimulasikan evolusi biologis dengan menerapkan seleksi, crossover, dan mutasi pada populasi solusi kandidat. Selama generasi-generasi berikutnya, populasi berevolusi menuju solusi optimal.
- **Induksi Pemrograman:** Tujuan GP adalah untuk secara otomatis menghasilkan program komputer yang mampu memecahkan masalah kompleks tanpa pemrograman eksplisit. Hal ini dicapai melalui penyempurnaan iteratif yang dipandu oleh fungsi kebugaran.
- **Operator Genetika:** GP beroperasi berdasarkan dua operasi utama:
 - **Crossover:** Menggabungkan segmen dari dua program induk untuk menciptakan keturunan, mendorong keragaman dan rekombinasi sifat.
 - **Mutasi:** Memperkenalkan perubahan acak pada suatu program, memastikan variabilitas dan mencegah konvergensi prematur.

- **Fungsi Kebugaran:** Fungsi kebugaran mengevaluasi seberapa baik suatu program memecahkan masalah target. Program dengan kebugaran yang lebih tinggi lebih mungkin dipilih untuk reproduksi, mengikuti prinsip "bertahan hidup yang terkuat."
- **Proses Iteratif:** GP mengulangi seleksi, persilangan, dan mutasi di beberapa generasi, secara bertahap meningkatkan kualitas solusi. Siklus iteratif ini berlanjut hingga kondisi penghentian – seperti ambang kebugaran yang telah ditentukan sebelumnya atau jumlah generasi maksimum – terpenuhi.
- **Pewarisan:** Menurut prinsip-prinsip dasar sistem biologis, keturunan mewarisi sifat dari kedua orang tua. Misalnya, orang tua buta warna lebih mungkin memiliki anak buta warna.
- **Seleksi:** Prinsip seleksi alam menyatakan bahwa alam memilih individu dengan sifat yang lebih baik untuk bereproduksi dan menciptakan generasi berikutnya. Proses ini bervariasi di berbagai spesies dan bergantung pada konteks.
- **Reproduksi:** Sifat dari kedua orang tua diturunkan ke generasi berikutnya. Beberapa sifat dari setiap orang tua yang tidak dorman dipilih dan diturunkan, sementara yang lain tetap dorman.
- **Variasi:** Evolusi membutuhkan variasi sifat antara pasangan yang bereproduksi. Tanpa variasi, sifat dan kombinasi baru tidak dapat muncul.

Istilah kunci tambahan yang akan berguna untuk memahami konsep GP meliputi :

- **Generasi:** Generasi mengacu pada iterasi dalam GA. Generasi acak pertama disebut generasi nol.
- **Genotipe:** Genotipe adalah struktur solusi yang dihasilkan oleh algoritma.
- **Fenotipe:** Fenotipe adalah hasil yang terlihat atau nyata dari genotipe.
- **Dekode:** Ini adalah proses translasi di mana kita menerjemahkan solusi dari ruang genotipe ke ruang fenotipe.
- **Encode:** Ini adalah proses translasi di mana kita menerjemahkan solusi dari ruang fenotipe ke ruang genotipe.
- **Populasi:** Populasi adalah himpunan solusi yang mungkin untuk masalah tersebut.
- **Kromosom:** Dalam ruang domain GA/GP, kromosom mewakili solusi potensial untuk suatu masalah. Kromosom dikodekan sebagai rangkaian nilai. Nilai sering direpresentasikan sebagai biner tetapi juga dapat memiliki struktur data lain. Ini analog dengan kromosom biologis, dengan gen yang mewakili parameter spesifik dari solusi. Algoritma mengembangkan populasi kromosom ini melalui seleksi, crossover, dan mutasi untuk menemukan solusi yang paling sesuai.
- **Keragaman:** Keragaman adalah keunikan setiap elemen dalam populasi yang dipilih. Semakin banyak variasi dalam keragaman umumnya baik untuk konvergensi GA.
- **Algoritma Evolusi (EA):** EA adalah optimasi metaheuristik yang memanfaatkan teknik yang berasal dari evolusi alami untuk menemukan solusi optimal untuk suatu masalah. GA didasarkan pada prinsip evolusi dan seleksi alami yang ditemukan di alam, di mana spesies yang paling sesuai beradaptasi dan bertahan hidup selama beberapa generasi.

GA berupaya mengoptimalkan solusi untuk masalah tertentu dengan mengembangkan populasi solusi kandidat melalui beberapa iterasi, atau generasi.

GP sering menggunakan pohon sintaks untuk merepresentasikan program di mana node internal adalah operator, dan daunnya adalah terminal. Pada bagian selanjutnya, kita akan membahas secara singkat aplikasi GP.

7.3 PENERAPAN ALGORITMA GENETIK

GA dan GP banyak diterapkan di berbagai domain yang membutuhkan pencarian dan optimasi yang efisien dalam ruang solusi yang besar dan kompleks. Berikut adalah area aplikasi utama :

- **Masalah Optimasi:** GA adalah teknik yang sangat baik untuk menyelesaikan masalah optimasi. Salah satu masalah optimasi yang terkenal adalah menemukan solusi terbaik dari sejumlah besar kemungkinan. Misalnya, masalah seperti optimasi fungsi matematika, penyetelan parameter, optimasi portofolio, alokasi sumber daya, dan lain-lain.
- **Pembelajaran Mesin:** GA dapat digunakan untuk ML dalam tugas yang melibatkan pengaturan model dan optimasi parameter. Lebih tepatnya, GA dapat diterapkan untuk mengoptimalkan setiap hyperparameter dari algoritma ML. Sebagai contoh, GA dapat digunakan untuk menentukan laju pembelajaran, parameter regularisasi apa pun yang terkait dengan algoritma, desain jaringan, dan lain-lain. GA juga dapat diterapkan untuk tujuan pemilihan fitur, di mana algoritma akan mengulangi populasi subset fitur, mencari fitur terpenting dari model.
- **Pemrosesan Sinyal dan Citra:** GA dapat diterapkan untuk aplikasi pemrosesan citra dan sinyal, termasuk rekonstruksi, pengurangan noise, ekstraksi fitur, dan identifikasi pola. GA dapat membantu mengoptimalkan pengaturan algoritma rekonstruksi, sehingga membantu meningkatkan kualitas citra. Dalam pemrosesan sinyal, GA juga dapat digunakan untuk mengoptimalkan pengaturan filter untuk mengurangi noise sinyal sambil mempertahankan informasi penting. GA juga dapat diterapkan pada ekstraksi fitur otomatis untuk mengembangkan algoritma ekstraksi fitur yang menemukan fitur dan pola penting dalam gambar atau sinyal.

Beberapa aplikasi dunia nyata dari GA dan GP meliputi:

- **Mobil Otonom Tesla:** Mobil Tesla menggunakan GA untuk meningkatkan teknologi pengemudian otonom mereka. Jaringan saraf yang digunakan dalam mobil-mobil ini berevolusi dan menyempurnakan parameternya menggunakan GA. GA memungkinkan evolusi berkelanjutan dari sistem dalam mobil untuk beradaptasi dengan kondisi mengemudi yang kompleks dari waktu ke waktu. Ini menjamin peningkatan kinerja, keselamatan, dan ekonomi sistem otonomnya.
- **Operasi Logistik Amazon:** Amazon menggunakan GA untuk pemenuhan pesanan dan optimasi operasi logistiknya. Algoritma ini mengatasi masalah perutean dan penjadwalan yang rumit, sehingga membuat rantai pasokan lebih efektif dan mengurangi waktu pengiriman. Evolusi solusi melalui data waktu nyata dengan GA

memungkinkan Amazon untuk merespons secara dinamis terhadap perubahan permintaan.

- **Optimasi Uber:** Uber menggunakan Evolutionary Optimizer, berdasarkan GA, untuk optimasi penetapan harga dinamisnya. Kerangka kerja ini mengembangkan strategi penetapan harga berdasarkan data historis dan pola permintaan waktu nyata, memilih strategi yang paling efektif melalui proses seleksi alam untuk memaksimalkan pendapatan sekaligus memastikan keadilan bagi pelanggan.

Sekarang setelah kita melihat aplikasi GP/GA, mari kita alihkan fokus kita ke bagaimana GA, yang terinspirasi oleh evolusi alam, dapat membantu memecahkan masalah yang kompleks. Sebelumnya, kami telah menunjukkan bahwa ada kebutuhan akan program komputer cerdas yang dapat berevolusi dan mencari solusi potensial dalam ruang yang luas secara efisien. Karena ruang pencarian seringkali sangat besar, metode brute-force tidak layak. GA dapat membantu menemukan solusi dengan cepat, seperti halnya evolusi alami yang bekerja untuk menemukan sifat-sifat baru. Di sini, kita akan mengeksplorasi bagaimana GA terstruktur dan bagaimana GA dapat menyederhanakan dan mengoptimalkan pencarian solusi.

7.4 STRUKTUR ALGORITMA GENETIK

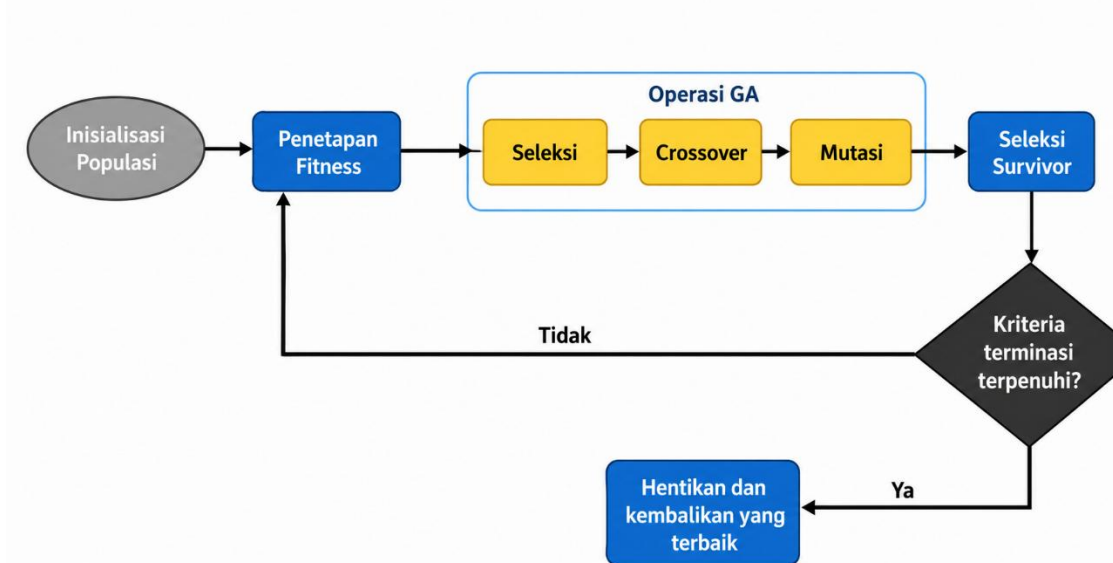
GA dan GP mengikuti proses iteratif yang terinspirasi oleh evolusi biologis. Tujuannya adalah untuk secara efisien mencari ruang solusi yang luas di mana metode brute-force tidak praktis. Alur kerja umum yang diikuti GA/GP dapat dijelaskan sebagai berikut:

1. **Inisialisasi:** Menghasilkan populasi awal solusi kandidat secara acak.
2. **Evaluasi Kebugaran:** Menilai setiap individu menggunakan fungsi kebugaran yang mengukur kualitas solusi.
3. **Seleksi:** Memilih individu dengan skor kebugaran yang lebih tinggi untuk reproduksi.
4. **Crossover (Rekombinasi):** Menggabungkan sifat dari dua atau lebih induk untuk menciptakan keturunan.
5. **Mutasi:** Memperkenalkan perubahan acak kecil untuk mempertahankan keragaman dan menghindari konvergensi prematur.
6. **Penggantian:** Membentuk populasi baru dengan mengganti individu yang lebih lemah dengan keturunan.
7. **Terminasi:** Mengulangi siklus hingga kriteria penghentian terpenuhi (misalnya, generasi maksimum atau tingkat kebugaran yang diinginkan).

Beberapa karakteristik utama dari alur kerja ini adalah:

- **Pencarian Berbasis Populasi:** Mempertahankan serangkaian solusi yang beragam daripada hanya satu kandidat, memungkinkan eksplorasi berbagai wilayah dalam ruang pencarian.
- **Peran Keacakan:** Seleksi dan mutasi memperkenalkan stokastisitas, membuat proses menjadi non-deterministik. Beberapa kali dijalankan dapat menghasilkan hasil yang berbeda.
- **Pelestarian Keragaman:** Memastikan algoritma tidak terjebak dalam optimasi lokal dengan mempertahankan variasi antar generasi.

Solusi sering dikodekan sebagai *kromosom*, biasanya dalam format biner atau format terstruktur lainnya. GP sering menggunakan *pohon sintaksis*, di mana simpul internal mewakili operator dan daun mewakili operan atau konstanta.



Gambar 7.1 Prinsip kerja GA/GP.

Proses umum bagaimana GA/GP bekerja ditunjukkan pada Gambar 7.1. Setelah menetapkan struktur dan alur kerja GA dan GP, kita sekarang beralih ke contoh praktis yang menunjukkan bagaimana konsep-konsep ini diimplementasikan. Contoh ini mengilustrasikan proses iteratif seleksi, crossover, dan mutasi dalam memecahkan masalah optimasi klasik – Masalah Penjual Keliling (*Traveling Salesman Problem/TSP*) – menggunakan Python.

7.5 CONTOH ALGORITMA GENETIK

TSP melibatkan pencarian rute terpendek yang mengunjungi sekumpulan kota tepat sekali dan kembali ke titik awal. Masalah ini menantang secara komputasi untuk dataset besar, menjadikannya kandidat ideal untuk GA.

Langkah 1: Pengaturan dan Persiapan Data: Pertama, kita akan mengimpor paket Python yang diperlukan. Perhatikan skrip Python yang disediakan pada Tabel 7.1

Di sini, kita mengimpor dua pustaka Python, yaitu *numpy* dan *random*. Kita juga mendefinisikan jarak antar kota. Sebuah matriks jarak dibuat di mana setiap elemen mewakili jarak antara dua kota. Diagonalnya diatur ke nol, karena jarak dari sebuah kota ke dirinya sendiri adalah nol.

Langkah 2: Membuat Populasi Awal dan Fungsi Kebugaran: Selanjutnya, kita membuat populasi awal. Seperti yang telah dibahas sebelumnya, populasi awal mewakili sekumpulan solusi acak yang mungkin. Di sini, setiap solusi adalah daftar yang berisi urutan kota dalam suatu rute. Kita dapat menyebutnya sebagai kromosom. Ekspresi `np.random.permutation(city_count)` menghasilkan urutan kota acak untuk setiap individu. Selanjutnya, kita mendefinisikan fungsi kebugaran. Fungsi kebugaran

mengevaluasi seberapa "baik" suatu solusi atau rute. Untuk TSP, kebugaran berbanding terbalik dengan total jarak rute. Semakin pendek rute, semakin baik kebugarannya (lihat Tabel 7.2).

Langkah 3: Operator Genetika – Seleksi, Crossover, dan Mutasi: Sekarang kita harus melakukan proses seleksi, crossover, dan mutasi (lihat Tabel 7.3). Seleksi adalah proses memilih individu induk untuk reproduksi. Di sini, individu dipilih berdasarkan kebugarannya. Dalam proses seleksi, asumsinya adalah semakin bugar suatu individu (yaitu, semakin pendek rutenya), semakin tinggi probabilitasnya untuk dipilih. Crossover menggabungkan dua rute induk untuk menciptakan keturunan. Metode *crossover berurutan* diimplementasikan di sini, di mana sebagian kota dari satu induk disalin langsung ke keturunan, dan kota-kota yang tersisa dari induk lainnya ditempatkan di posisi yang tersisa, memastikan tidak ada duplikasi. Fungsi mutasi memperkenalkan keacakan dengan mengubah keturunan setelah crossover. Di sini, kita mengimplementasikan *mutasi pertukaran*. Dalam implementasi ini, kita secara acak menukar dua kota dalam rute. Ini memastikan keragaman dalam populasi dan mencegah algoritma terjebak dalam optimasi lokal.

Langkah 4: Loop GA Utama: Sekarang, kita siap untuk mengimplementasikan GA/GP. Ini adalah loop utama. Dalam perulangan ini, kita berulang kali memilih individu, menerapkan crossover dan mutasi, dan mengganti populasi lama dengan generasi baru. Di sini, kita juga melacak solusi terbaik yang telah kita peroleh sejauh ini (lihat Tabel 7.4). Setelah menjalankan skrip pada Tabel 7.4, kita memperoleh *rute terbaik dan jarak totalnya*.

TABEL 7.1 Mengimpor Pustaka dan Dataset yang Diperlukan

```
import numpy as np
import random

city_count = 6
np.random.seed(42)

distance_matrix = np.random.randint(10, 100, size=(city_count,
city_count))
np.fill_diagonal(distance_matrix, 0)
```

TABEL 7.2 Skrip Python untuk Membuat Populasi Awal dan Fungsi Kebugaran

```
def create_population(pop_size, city_count):
    population = []
    for _ in range(pop_size):
        individual = list(np.random.permutation(city_count))
        population.append(individual)
    return population

def fitness(route):
    return sum(distance_matrix[route[i], route[i + 1]] for i
in range(len(route) - 1)) + distance_matrix[route, route[0]]
```

TABEL 7.3 Skrip Python untuk Seleksi, Crossover, dan Mutasi

```
def selection(population, fitness_scores):
    total_fitness = sum(fitness_scores)
    probabilities = [1 - (f / total_fitness) for f in
        fitness_scores]
    probabilities = np.array(probabilities) / sum(probabilities)
    selected_index = np.random.choice(range(len(population)),
        size=2, p=probabilities)
    return population[selected_index[0]], population[selected_index[1]]

def crossover(parent1, parent2):
    size = len(parent1)
    start, end = sorted(random.sample(range(size), 2))
    child = [-1] * size
    child[start:end] = parent1[start:end]
    pointer = 0
    for gene in parent2:
        if gene not in child:
            while child[pointer] != -1:
                pointer += 1
            child[pointer] = gene
    return child

def mutate(individual, mutation_rate=0.1):
    if random.random() < mutation_rate:
        i, j = random.sample(range(len(individual)), 2)
        individual[i], individual[j] = individual[j], individual[i]
```

TABEL 7.4 Skrip Python untuk Mengimplementasikan GP/GA untuk Masalah TSP

```
def genetic_algorithm(city_count, distance_matrix, pop_size=10,
    generations=100, mutation_rate=0.1):
    population = create_population(pop_size, city_count)
    best_route = None
    best_fitness = float('inf')
    for gen in range(generations):
        fitness_scores = [fitness(ind) for ind in population]
        new_population = []

        # Update best solution
        for ind, score in zip(population, fitness_scores):
            if score < best_fitness:
                best_fitness = score
                best_route = ind
        while len(new_population) < pop_size:
            parent1, parent2 = selection(population, fitness_scores)
            child = crossover(parent1, parent2)
            mutate(child, mutation_rate)
            new_population.append(child)
        population = new_population
        print(f"Generation {gen + 1}: Best Fitness =
{best_fitness}")
    return best_route, best_fitness

best_route, best_fitness = genetic_algorithm(city_count,
    distance_matrix)
```

```
print("\nBest Route:", best_route)
print("Best Distance:", best_fitness)
```

Algoritma ini menghasilkan rute terbaik dan jarak totalnya. Karena sifat acaknya, hasil dapat bervariasi antar eksekusi, mencerminkan sifat non-deterministik dari GA.

7.6 RINGKASAN

Dalam bab ini, kita telah mengeksplorasi GA dan GP – teknik optimasi yang terinspirasi dari alam yang menerapkan prinsip-prinsip evolusi untuk memecahkan masalah kompleks. Algoritma ini meniru proses biologis seperti *seleksi*, *crossover*, dan *mutasi*, memungkinkan penyempurnaan iteratif dari solusi kandidat lintas generasi. Kita mulai dengan memperkenalkan konsep GP dan fondasinya dalam *seleksi alam*, dengan menarik paralel antara evolusi biologis dan pemecahan masalah komputasi. Komponen kunci seperti *fungsi kebugaran*, *operator genetik*, dan *keragaman populasi* dibahas, bersama dengan terminologi penting seperti *genotipe*, *fenotipe*, dan *representasi kromosom*. Bab ini menyoroti aplikasi praktis GA/GP dalam *optimasi*, *pembelajaran mesin* (penyetelan hiperparameter dan pemilihan fitur), *pemrosesan sinyal dan gambar*, dan skenario dunia nyata seperti *mengemudi otonom*, *optimasi logistik*, dan penetapan *harga dinamis*.

Kemudian, kami meneliti *struktur dan alur kerja* GA/GP, menekankan siklus iteratif inisialisasi, evaluasi kebugaran, seleksi, crossover, mutasi, dan terminasi. Terakhir, kami mendemonstrasikan implementasi GA berbasis Python untuk *TSP*, mengilustrasikan bagaimana prinsip-prinsip ini diterjemahkan ke dalam kode untuk menyelesaikan tantangan optimasi skala besar. Dengan memanfaatkan prinsip-prinsip evolusi, GA dan GP menyediakan solusi yang kuat, terukur, dan adaptif untuk masalah di mana metode tradisional gagal karena kompleksitas atau kendala komputasi. Teknik-teknik ini membentuk fondasi penting untuk membangun sistem cerdas yang mampu menangani tuntutan Big Data dan aplikasi AI tingkat lanjut.

PERTANYAAN

1. Bandingkan dan kontraskan Algoritma Genetika (GA) dan Pemrograman Genetika (GP). Analisis bagaimana elemen-elemen berikut membentuk dinamika pencarian:
 - a. Representasi/pengkodean (misalnya, kromosom vs. pohon sintaksis)
 - b. Genotipe vs. fenotipe dan proses pengodean/dekode
 - c. Peran seleksi, crossover, mutasi, dan evaluasi kebugaran. Sertakan diagram ringkas atau alur kerja poin-poin dalam jawaban Anda.
2. Alur Kerja dan Konvergensi GA/GP
 - a. Jelaskan proses GA/GP dari awal hingga akhir, mulai dari inisialisasi populasi hingga seleksi, crossover, mutasi, penggantian, dan terminasi.
 - b. Diskusikan:
 - i. Pentingnya keacakan dalam proses
 - ii. Fenomena konvergensi prematur dan implikasinya

iii. Setidaknya dua strategi untuk mempertahankan keragaman populasi

3. Aplikasi

Pilih tiga aplikasi yang disebutkan dalam bab ini (misalnya, penyetelan hiperparameter/seleksi fitur ML, pemrosesan gambar/sinyal, kendaraan otonom Tesla, perutean logistik Amazon, dan penetapan harga Uber). Untuk masing-masing

- a. Jelaskan tujuannya
 - b. Jelaskan pengkodean solusi kandidat
 - c. Berikan alasan mengapa GA/GP lebih disukai daripada pencarian brute-force dalam konteks ini
4. GP biasanya mewakili program seperti
- a. Kueri SQL atas skema yang dinormalisasi
 - b. Pohon sintaksis yang simpul internalnya adalah operator dan daunnya adalah terminal
 - c. Peta hash dari fitur ke bobot
 - d. Hanya array bit dengan panjang tetap
5. Dalam GA/GP, fungsi utama dari fungsi fitness adalah untuk
- a. Menjamin hasil yang sama pada setiap eksekusi
 - b. Secara acak memutasi individu untuk meningkatkan keragaman
 - c. Mengukur kualitas solusi dan mendorong pemilihan induk
 - d. Paksa evaluasi tepat satu kali untuk setiap kromosom
6. [Latihan Pemrograman – Menengah] Perluas contoh GA Traveling Salesman Problem (TSP) dari bab ini
- a. Ganti seleksi proporsional (gaya roulette) dengan seleksi turnamen (ukuran $k = 3$)
 - b. Implementasikan pelacakan dan pencetakan kebugaran terbaik di setiap generasi
 - c. Implementasikan penghentian dini jika kebugaran terbaik tidak meningkat selama 20 generasi berturut-turut
7. [Latihan Pemrograman – Tingkat Lanjut] Implementasikan sistem Pemrograman Genetika dasar untuk regresi simbolik guna mendekati fungsi $y = x^2 + x + 1$.
- Himpunan primitif: {+, -, *, protected_div}
 - Terminal: $x \cup$ konstanta sementara dalam $[-5, 5]$
 - Representasi: Pohon ekspresi dengan inisialisasi terbatas kedalaman
 - Operator: Persilangan subpohon; mutasi titik (mengganti subpohon acak)
 - Kebugaran: Kesalahan Kuadrat Rata-rata (MSE) pada 50 titik yang diambil sampelnya dari $x \in [-3, 3]$
 - Keluaran: Ekspresi terbaik dan MSE-nya; plot nilai prediksi vs. nilai sebenarnya

BAB 8

DASAR PEMBELAJARAN MESIN DAN AI

8.1 DEFINISI PEMBELAJARAN MESIN

Dalam bab ini, kita meletakkan dasar untuk memahami konsep-konsep kunci dalam pembelajaran mesin (ML) dan kecerdasan buatan (AI), dengan fokus pada jaringan saraf (NN) dan varian arsitekturnya. Diskusi ini membuka jalan untuk mengeksplorasi topik AI tingkat lanjut seperti pembelajaran penguatan (RL) dan pemrosesan bahasa alami (NLP) di bab-bab selanjutnya. Kita mulai dengan pengantar singkat tentang NN dan jaringan saraf buatan (ANN), diikuti dengan tinjauan arsitektur NN dan konsep inti seperti perseptron, fungsi aktivasi, dan blok bangunan model transformer. ML banyak digunakan di berbagai bidang seperti kedokteran, keuangan, dan periklanan, di mana ia mengungkap pola dan hubungan dalam kumpulan data kompleks berdimensi tinggi. Teknik-teknik ini, terutama pembelajaran mendalam, memperoleh kekuatan prediktif dari kumpulan data pelatihan skala besar dan menunjukkan peningkatan kinerja seiring tersedianya lebih banyak data.

ML juga berfungsi sebagai inti kognitif dari banyak sistem AI. Secara umum, sistem AI merasakan lingkungannya melalui sensor dan bertindak berdasarkan sensor tersebut melalui efektor. ML menginterpretasikan data sensor dan menghasilkan keputusan atau tindakan. Misalnya, kendaraan otonom mengintegrasikan input dari kamera, GPS, sonar, dan LiDAR untuk memutuskan apakah akan berakselerasi, mengerem, atau berbelok. ML dapat dilihat sebagai mekanisme pemrosesan informasi yang mengubah data sensor mentah menjadi output yang dapat ditindaklanjuti. Supervised ML, yang menjadi fokus bab ini, melibatkan pelatihan model pada dataset berlabel untuk mempelajari hubungan antara input dan output. Selama pelatihan, model membuat prediksi, membandingkannya dengan label yang diketahui, dan menyesuaikan parameter internalnya untuk mengurangi kesalahan dan meningkatkan akurasi. Tujuannya adalah untuk mengembangkan model yang mampu membuat prediksi yang tepat pada data baru yang belum pernah dilihat sebelumnya.

Bagaimana cara kerja supervised ML? Proses dimulai dengan dataset pelatihan di mana setiap contoh input dihubungkan dengan label output yang benar. Misalnya, dalam dataset pengenalan gambar, gambar hewan diberi label dengan nama mereka, seperti *kucing*, *anjing*, atau *gajah*. Algoritma ML mempelajari hubungan antara fitur input dan label output dengan menganalisis data berlabel ini. Algoritma ini mendeteksi pola dan membuat fungsi pemetaan yang menghubungkan keduanya. Setelah dilatih, model diuji dengan data baru yang belum pernah dilihat sebelumnya. Model tersebut membuat prediksi, yang kemudian dibandingkan dengan label sebenarnya. Perbedaan atau kesalahan antara prediksi dan label sebenarnya digunakan untuk menyempurnakan parameter model, sehingga membantu meningkatkan kinerjanya pada setiap iterasi. Setelah menetapkan peran ML terawasi dan NN dalam mengubah data mentah menjadi wawasan yang dapat ditindaklanjuti, sekarang kita akan membahas dasar-dasar matematika yang mendasari model-model ini. Memahami bagaimana

neuron individual beroperasi – melalui input berbobot, istilah bias, dan fungsi aktivasi – sangat penting untuk memahami perilaku arsitektur jaringan yang lebih besar seperti ANN, DNN, dan variannya.

8.2 CARA KERJA JARINGAN SARAF

NN adalah model matematika yang dirancang untuk mempelajari dan mengenali pola kompleks dalam data. Terinspirasi oleh sistem saraf biologis, NN terdiri dari neuron buatan (atau node) yang saling terhubung yang memproses sinyal input dan menghasilkan output melalui serangkaian transformasi. Setiap neuron menerima banyak input, masing-masing dikaitkan dengan bobot. Input ini digabungkan secara linier, disesuaikan dengan istilah bias, dan dilewatkan melalui fungsi aktivasi non-linier. Proses ini memungkinkan jaringan untuk memodelkan hubungan non-linier yang kompleks antara input dan output. Hal ini diilustrasikan pada Gambar 8.1.

- **Perceptron:** Unit fundamental dari NN adalah *perceptron*. Dalam *perceptron*, input bersih dihitung sebagai jumlah tertimbang dari fitur input ditambah bias. Input bersih ini kemudian dilewatkan melalui fungsi aktivasi non-linier untuk menghasilkan output. Misalkan vektor inputnya adalah:

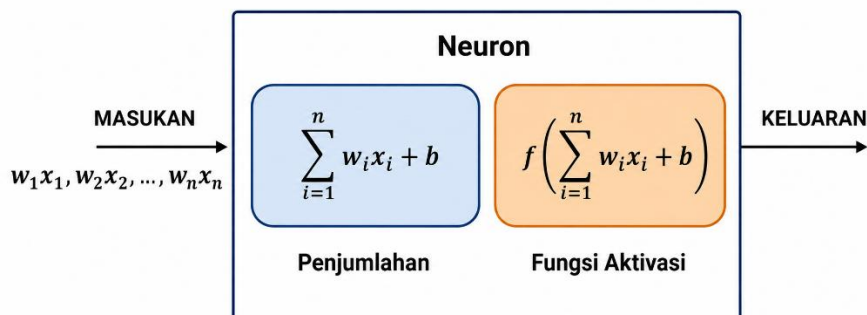
$$\mathbf{x} = [x_1, x_2, \dots, x_n]$$

dan vektor bobot yang sesuai adalah:

$$\mathbf{w} = [w_1, w_2, \dots, w_n]$$

Perceptron menghitung hasil perkalian titik dari vektor-vektor ini dan menambahkan suku bias:

$$z = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^n w_i x_i + b$$



Gambar 8.1 Model neuron buatan.

Input bersih ini kemudian dilewatkan melalui fungsi aktivasi untuk menghasilkan output:

$$y = f(z) = f(\sum_{i=1}^n w_i x_i + b) \quad (8.1)$$

Istilah bias b memungkinkan fungsi aktivasi untuk bergeser ke kiri atau ke kanan, sehingga model dapat lebih sesuai dengan data. Fungsi aktivasi f memperkenalkan non-linearitas, yang sangat penting untuk mempelajari pola-pola kompleks.

- **Fungsi Aktivasi:** *Fungsi aktivasi* adalah transformasi matematis yang diterapkan pada output neuron. Fungsi ini menentukan apakah neuron harus "aktif" dan bagaimana outputnya menyebar melalui jaringan. *Fungsi aktivasi* non-linear lebih disukai karena memungkinkan NN untuk mendekati pemetaan non-linear yang kompleks antara input dan output. Di sini, kami menyajikan beberapa *fungsi aktivasi* yang umum digunakan :
- Fungsi aktivasi *sigmoid*, atau *logistik*, mengubah bilangan riil menjadi nilai antara 0 dan 1. Fungsi ini sangat berguna dalam model yang membutuhkan prediksi berbasis probabilitas sebagai output. Bentuk fungsi *sigmoid* menyerupai kurva S, dan secara matematis dinyatakan sebagai:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Fungsi sigmoid dapat didiferensiasi, sehingga memungkinkan kita untuk menghitung kemiringannya di titik mana pun pada kurva. Selain itu, fungsi ini monoton, artinya secara konsisten tidak menurun (atau tidak meningkat), yang membantu dalam pembelajaran yang stabil selama pelatihan NN.

- Fungsi aktivasi *tanh*, atau *tangen hiperbolik*, mirip dengan fungsi sigmoid dengan kurva berbentuk S, tetapi memetakan bilangan riil ke rentang -1 hingga $+1$. Outputnya berpusat pada nol, dan sifat nonliniernya umumnya lebih disukai daripada fungsi sigmoid karena sering membantu NN konvergen lebih cepat. Secara matematis, fungsi *tanh* dinyatakan sebagai:

$$\tanh(x) = 2\sigma(2x) - 1$$

Fungsi tanh adalah versi skala dari sigmoid dan dapat didiferensiasi. Seperti sigmoid, fungsi ini monoton – baik tidak menurun atau tidak meningkat – dan banyak digunakan dalam jaringan saraf feedforward karena sifat-sifatnya yang menguntungkan. Jaringan saraf feedforward akan dijelaskan lebih detail di bab ini.

- *Rectified Linear Unit* atau *ReLU* adalah salah satu fungsi aktivasi yang paling umum digunakan dalam jaringan saraf dan pembelajaran mendalam. Fungsi ini didefinisikan sebagai $f(x) = \max(0, x)$. Ini berarti bahwa jika input x kurang dari nol, outputnya adalah nol; jika tidak, outputnya sama dengan input itu sendiri. ReLU mengubah bilangan riil menjadi rentang dari 0 hingga tak terhingga. Fungsi ini monoton, terus meningkat untuk input non-negatif. Namun, untuk input negatif, output langsung menjadi nol, yang terkadang dapat membatasi kemampuan model untuk belajar dari data secara efektif, terutama jika banyak input berada di wilayah ini.

- Fungsi aktivasi *softmax* mengubah vektor K input bernilai riil menjadi vektor K nilai yang jumlahnya sama dengan 1. Nilai input ini dapat berupa positif, negatif, nol, atau lebih besar dari satu, tetapi fungsi *softmax* mengubahnya menjadi probabilitas dalam rentang, sehingga dapat diinterpretasikan sebagai probabilitas kelas. *Softmax* sering disebut sebagai regresi logistik multikelas karena memperluas konsep regresi logistik untuk menangani banyak kelas, dan bentuk matematisnya sangat mirip dengan fungsi sigmoid. Secara formal, fungsi *softmax* dinyatakan sebagai:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}$$

di mana $\vec{z}$ adalah vektor input fungsi softmax yang terdiri atas k elemen $(z_0, z_1, \dots, z_k)$, z_i merupakan elemen dari vektor input, dan k adalah jumlah kelas dalam pengklasifikasi multikelas. Setiap fungsi aktivasi memiliki kekuatan masing-masing dan dipilih berdasarkan tugas spesifik dan arsitektur jaringan. Misalnya, ReLU biasanya digunakan di lapisan tersembunyi jaringan dalam, sedangkan softmax digunakan di lapisan keluaran untuk klasifikasi. Sekarang kita memiliki pemahaman yang baik tentang model neuron buatan dan fungsi aktivasi, mari kita perkenalkan ANN secara formal.

8.3 JARINGAN SARAF BUATAN

ANN adalah model komputasi yang terinspirasi oleh pemrosesan saraf biologis. Mereka terdiri dari lapisan neuron buatan (atau node) yang saling terhubung, yang masing-masing mengubah sinyal input menjadi output melalui penjumlahan berbobot dan aktivasi non-linear. Jaringan ini sangat terparalelisasi dan dinamis, sering direpresentasikan sebagai grafik berarah yang memproses informasi berdasarkan respons keadaan internalnya terhadap rangsangan eksternal. Dalam aplikasi ML, ANN digunakan untuk memodelkan hubungan antara variabel input (fitur) dan variabel output (target). Arsitektur ANN yang umum meliputi:

- **Lapisan Input:** Menerima input numerik yang dinormalisasi yang sesuai dengan fitur dataset.
- **Lapisan Tersembunyi:** Memproses input menggunakan koneksi berbobot dan fungsi aktivasi.
- **Lapisan Output:** Menghasilkan prediksi atau klasifikasi berdasarkan sinyal yang diproses.

Setiap neuron dalam jaringan menerima serangkaian input, mengalikannya dengan bobot yang sesuai, menambahkan suku bias, dan menerapkan fungsi aktivasi (misalnya, sigmoid, tanh, dan ReLU) untuk menghasilkan sinyal output. Output ini kemudian diteruskan ke neuron di lapisan berikutnya, membentuk aliran informasi umpan maju. Arsitektur ANN yang banyak digunakan adalah *multilayer perceptron (MLP)*, yang biasanya terdiri dari tiga jenis lapisan:

- **Lapisan Masukan:** Satu neuron per fitur.
- **Lapisan Tersembunyi:** Jumlah neuron sering ditentukan melalui eksperimen.
- **Lapisan Keluaran:** Satu neuron per kelas (dalam tugas klasifikasi).

MLP dilatih menggunakan algoritma backpropagation, yang menyesuaikan bobot dan bias untuk meminimalkan kesalahan prediksi. Selama pelatihan, model melakukan:

- **Propagasi Maju:** Masukan dilewatkan melalui jaringan untuk menghitung keluaran.
- **Perhitungan Kesalahan:** Perbedaan antara keluaran yang diprediksi dan keluaran aktual dihitung.
- **Propagasi Mundur:** Gradien kesalahan dipropagasi mundur untuk memperbarui bobot.

Proses iteratif ini berlanjut hingga model konvergen atau mencapai jumlah iterasi pelatihan yang telah ditentukan. Setelah kita membahas struktur dan komponen ANN, khususnya MLP, kita akan beralih ke mekanisme bagaimana jaringan ini belajar. Ini melibatkan dua proses kunci – *propagasi maju* dan *propagasi mundur* – yang bersama-sama memungkinkan jaringan untuk menyesuaikan parameter internalnya dan meningkatkan akurasi prediksi dari waktu ke waktu.

8.4 CARA MESIN BELAJAR DARI KESALAHAN

Dalam ANN, jumlah neuron lapisan input sama dengan jumlah fitur input, sedangkan jumlah neuron lapisan output sesuai dengan jumlah kelas yang akan diprediksi. Dalam propagasi maju, fitur input dilewatkan melalui jaringan lapis demi lapis untuk menghitung aktivasi. Setiap neuron di lapisan tersembunyi melakukan langkah-langkah berikut:

- Menghitung jumlah tertimbang dari inputnya ditambah bias.
- Menerapkan fungsi aktivasi (misalnya, ReLU, sigmoid, dan tanh).
- Meneruskan output ke lapisan berikutnya.

Proses ini berlanjut melalui semua lapisan tersembunyi dan berpuncak pada lapisan output, yang menghasilkan prediksi akhir. Istilah bias membantu menggeser ambang aktivasi, memungkinkan neuron untuk aktif bahkan ketika sinyal input lemah. Setelah jaringan menghasilkan output, output tersebut dibandingkan dengan label sebenarnya untuk menghitung kesalahan. Kemudian, propagasi mundur dimulai, menggunakan kesalahan ini untuk memperbarui parameter jaringan:

1. Turunan dari fungsi aktivasi dikalikan dengan kesalahan keluaran untuk menghitung gradien.
2. Gradien ini dipropagasi mundur melalui jaringan untuk menghitung kontribusi kesalahan untuk setiap lapisan tersembunyi.
3. Bobot dan bias diperbarui menggunakan teknik optimasi (misalnya, penurunan gradien).

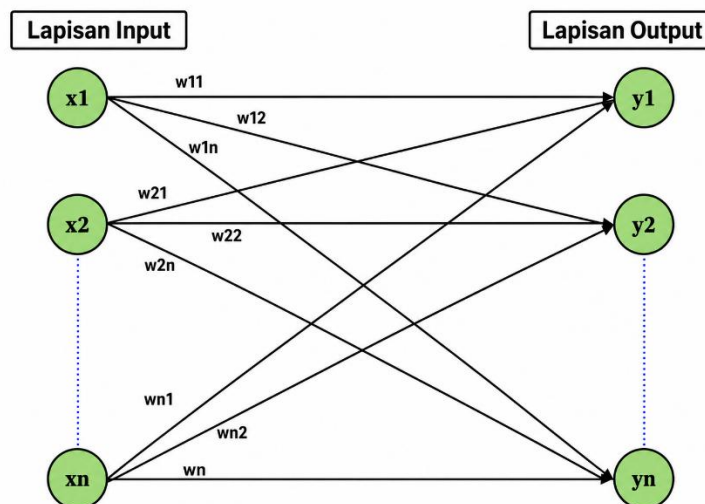
Proses ini berlanjut hingga bobot input ke lapisan tersembunyi pertama diperbarui. Pelatihan melibatkan pengulangan proses maju dan mundur selama beberapa iterasi (epoch) hingga model konvergen atau mencapai kriteria penghentian yang telah ditentukan. Dengan pemahaman tentang bagaimana NN belajar melalui propagasi maju dan mundur, kita sekarang beralih ke desain arsitektur ANN. Cara neuron saling terhubung – khususnya di lapisan tersembunyi – memainkan peran penting dalam menentukan daya komputasi jaringan, kapasitas representasional, dan kesesuaian untuk berbagai tugas.

8.5 VARIAN ARSITEKTUR ANN

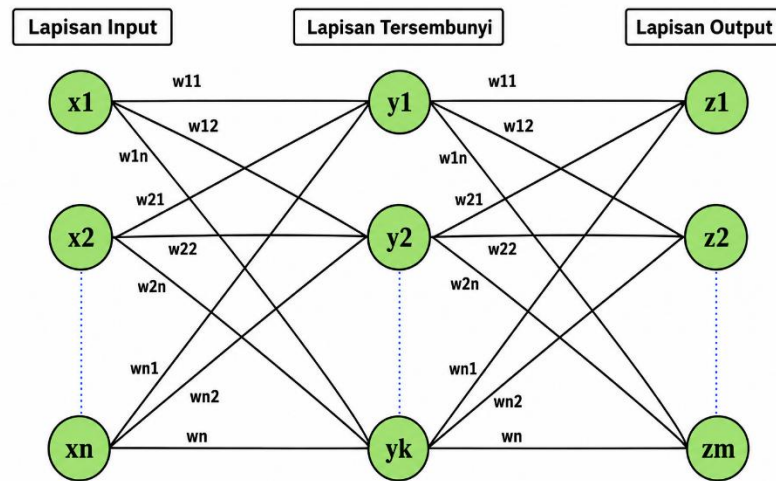
Interkoneksi merupakan elemen kunci dari arsitektur ANN. Interkoneksi menentukan bagaimana elemen pemrosesan (neuron) diatur dan dihubungkan. Semua jaringan memiliki lapisan input dan output; yang membedakan arsitektur adalah lapisan tersembunyi, yang tidak dapat diamati secara langsung dan bertindak sebagai *kotak hitam*. Penambahan lapisan tersembunyi meningkatkan daya komputasi dan kompleksitas model. Di sini, kita akan secara singkat menyortir lima jenis interkoneksi dasar :

- *Jaringan feed-forward lapisan tunggal* (lihat Gambar 8.2) hanya memiliki lapisan input dan output. Lapisan input diberi bobot dan digabungkan untuk menghasilkan output.
- *Jaringan feed-forward multilayer* (lihat Gambar 8.3) memiliki satu atau lebih lapisan tersembunyi di antara lapisan input dan output, meningkatkan kapasitas representasi jaringan.
- *Jaringan node tunggal dengan umpan balik* (lihat Gambar 8.4) adalah node pemrosesan tunggal yang menerima outputnya sendiri sebagai input. Representasi ini adalah loop umpan balik lokal.
- *Jaringan berulang lapisan tunggal* (lihat Gambar 8.5) adalah lapisan yang keluarannya diumpankan kembali ke lapisan yang sama atau lapisan sebelumnya, sehingga membentuk koneksi umpan balik.
- Terakhir, *jaringan berulang multilapisan* (lihat Gambar 8.6) memiliki beberapa lapisan dengan koneksi berulang (loop tertutup), sehingga memungkinkan perilaku temporal atau bergantung pada keadaan.

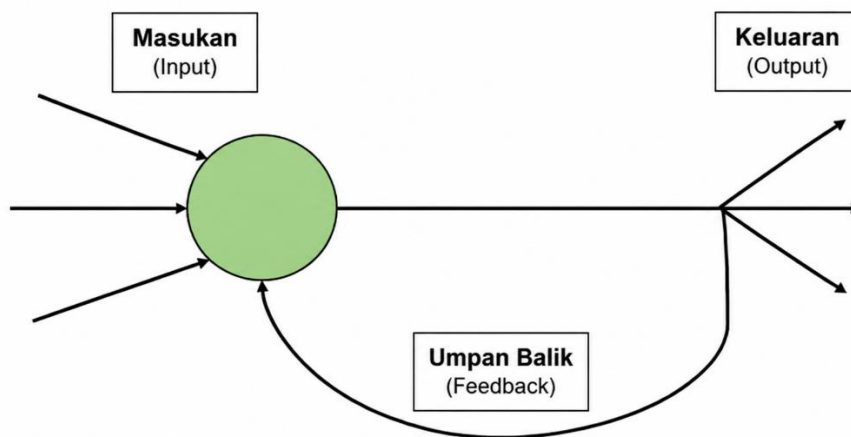
Penting untuk dicatat bahwa *jaringan umpan balik* mengarahkan keluaran kembali ke node sebelumnya, sedangkan *jaringan berulang* adalah jaringan umpan balik dengan loop tertutup yang mendukung perilaku temporal dinamis.



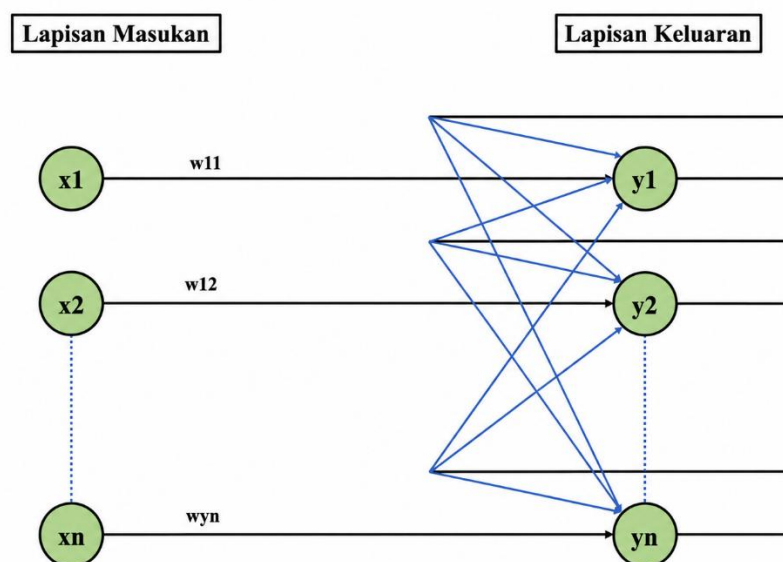
Gambar 8.2 Jaringan feed-forward satu lapis.



Gambar 8.3 Jaringan feed-forward multi-lapis.



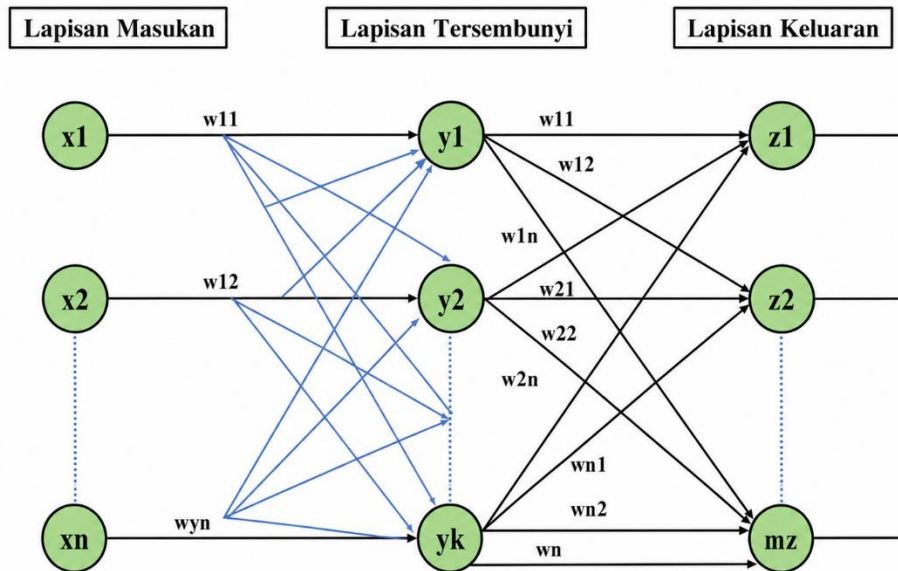
Gambar 8.4 Simpul tunggal dengan umpan baliknya sendiri.



Gambar 8.5 Jaringan berulang lapisan tunggal.

8.6 PENGATURAN PARAMETER MODEL

Setelah desain arsitektur jaringan saraf tiruan ditetapkan, langkah penting selanjutnya adalah mengonfigurasi hiperparameternya – pengaturan yang mengatur struktur dan perilaku pembelajaran model. Parameter-parameter ini tidak dipelajari selama pelatihan tetapi harus ditentukan sebelumnya, dan nilainya dapat secara signifikan memengaruhi kinerja jaringan, kecepatan konvergensi, dan kemampuan generalisasi. Parameter umum untuk ANN adalah:



Gambar 8.6 Jaringan berulang multilayer.

- **Jumlah Lapisan Tersembunyi dan Neuron per Lapisan:** Menentukan kedalaman dan kapasitas jaringan. Lebih banyak lapisan dan neuron dapat memodelkan fungsi kompleks tetapi dapat meningkatkan risiko overfitting.
- **Dropout:** Teknik regularisasi yang secara acak menonaktifkan neuron selama pelatihan untuk mencegah overfitting.
- **Inisialisasi Bobot:** Mempengaruhi seberapa cepat dan efektif jaringan belajar. Inisialisasi harus kompatibel dengan fungsi aktivasi yang dipilih.
- **Fungsi Aktivasi:** Menentukan bagaimana neuron mentransformasikan masukan. Pilihan umum meliputi ReLU, sigmoid, dan tanh.
- **Tingkat Pembelajaran:** Mengontrol ukuran pembaruan bobot selama pelatihan. Tingkat yang terlalu tinggi dapat menyebabkan ketidakstabilan; terlalu rendah dapat memperlambat konvergensi.
- **Jumlah Epoch:** Menentukan berapa banyak lintasan penuh yang dilakukan model pada kumpulan data pelatihan.
- **Ukuran Batch:** Menentukan berapa banyak sampel yang diproses sebelum memperbarui parameter. Batch yang lebih kecil menawarkan pembaruan yang lebih sering; batch yang lebih besar meningkatkan efisiensi komputasi.

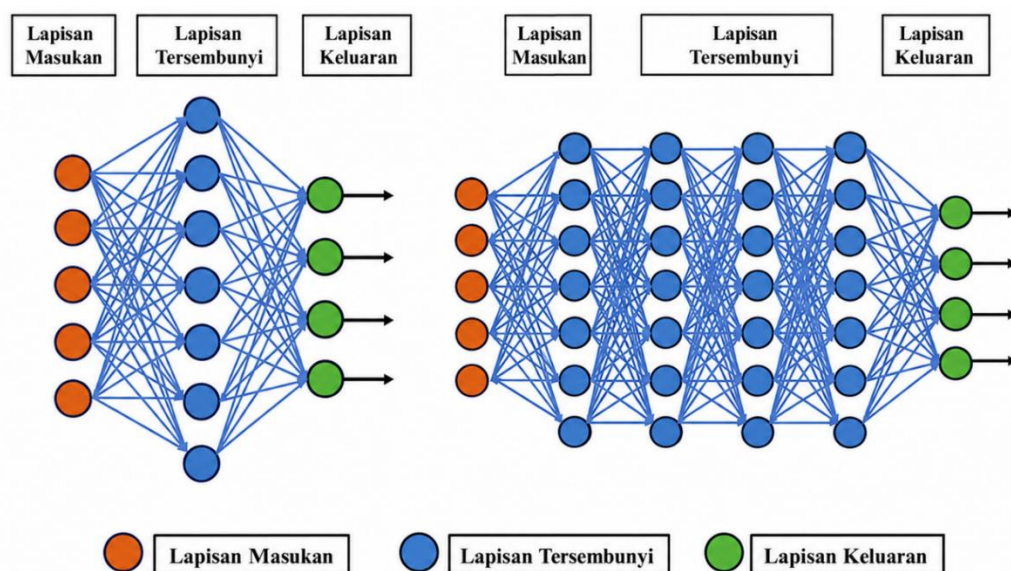
Terdapat beberapa strategi penyyetelan hiperparameter, termasuk:

- **Pencarian Manual:** Melibatkan penyesuaian hiperparameter secara iteratif berdasarkan intuisi dan pengalaman. Biasanya dimulai dengan perubahan yang luas dan dipersempit pada rentang yang menjanjikan.
- **Pencarian Grid:** Mengevaluasi secara menyeluruh semua kombinasi hiperparameter pada grid yang telah ditentukan. Sederhana tetapi mahal secara komputasi, terutama dengan banyak parameter.
- **Pencarian Acak:** Mengambil sampel nilai hiperparameter dari distribusi yang ditentukan. Seringkali lebih efisien daripada pencarian grid di ruang berdimensi tinggi.

Sekarang kita akan mengeksplorasi perluasan yang ampuh dari model ANN – *jaringan saraf dalam* (DNN). DNN dibangun di atas fondasi ANN dengan menambahkan beberapa lapisan tersembunyi, memungkinkan jaringan untuk mempelajari representasi hierarkis dan menyelesaikan tugas yang lebih kompleks di berbagai domain seperti pengenalan gambar, NLP, dan peramalan deret waktu.

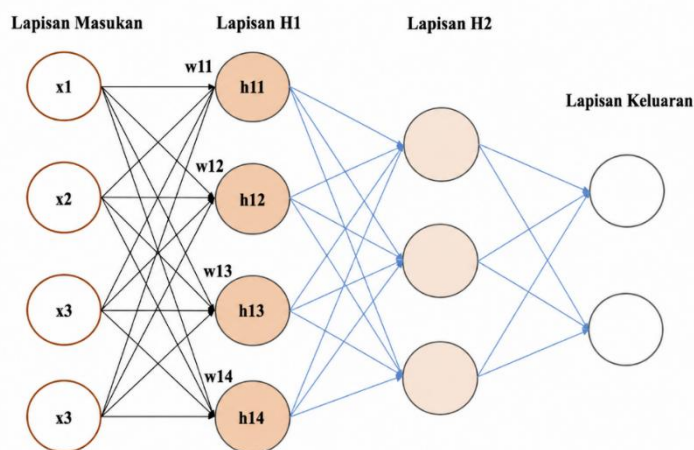
8.7 JARINGAN SARAF DALAM

Jaringan saraf dalam (DNN) adalah jaringan saraf tiruan (ANN) dengan beberapa lapisan unit pemrosesan nonlinier (lapisan tersembunyi) yang tersusun di antara lapisan input dan output. Istilah "*dalam*" mengacu pada jumlah lapisan, yang memungkinkan jaringan untuk mengekstrak fitur pada beberapa tingkat abstraksi. Lapisan awal mempelajari fitur tingkat rendah (misalnya, tepi dalam gambar), sementara lapisan yang lebih dalam menggabungkan fitur-fitur ini menjadi representasi tingkat yang lebih tinggi (misalnya, bentuk atau objek). Lihat Gambar 8.7 untuk perbandingan visual antara DNN dan ANN tradisional. Elemen kunci dari DNN adalah neuron (unit), lapisan (input, beberapa lapisan tersembunyi, dan output), bobot dan bias, fungsi aktivasi, fungsi kerugian, pengoptimal, regularisasi, dan motif arsitektural (konvolusi, rekurensi, perhatian, koneksi residual, dan normalisasi).



Gambar 8.7 DNN (gambar kiri) dan ANN (gambar kanan).

Meskipun memiliki kekuatan yang besar, DNN lebih sulit dilatih dan lebih rentan terhadap overfitting. Jaringan saraf dalam (DNN) mengandung parameter yang jauh lebih banyak daripada jaringan saraf dangkal, sehingga meningkatkan biaya komputasi dan kebutuhan akan kumpulan data besar serta regularisasi. Sebagian besar DNN adalah jaringan feedforward, di mana data mengalir dalam satu arah – dari input ke output. Kedalaman tambahan memungkinkan komposisi fitur hierarkis, memungkinkan model dalam untuk merepresentasikan hubungan non-linear yang kompleks dengan lebih sedikit neuron daripada jaringan saraf dangkal, mencapai kinerja yang serupa. Lihat Gambar 8.8 untuk arsitektur DNN 2 lapis yang khas. Terdapat beberapa varian arsitektur DNN yang berbeda. Pada bagian ini, kita akan membahas secara singkat MLP, jaringan saraf konvolusional (CNN), jaringan saraf berulang (RNN), jaringan memori jangka pendek panjang (LSTM), dan Transformer.



Gambar 8.8 Arsitektur DNN 2 lapis yang khas.

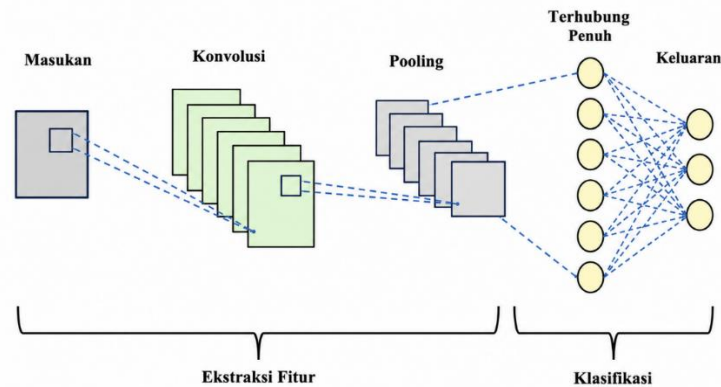
8.8 PERCEPTRON MULTI-LAPIS

MLP adalah jaringan saraf umpan-maju (feed-forward) yang terhubung penuh dan memetakan vektor masukan ke keluaran melalui beberapa lapisan bertumpuk. Setiap lapisan, kecuali lapisan masukan, terhubung secara padat ke lapisan berikutnya. Setiap simpul merupakan neuron dengan fungsi aktivasi nonlinier. MLP umumnya dilatih secara terawasi menggunakan metode propagasi balik (backpropagation). Jaringan melakukan propagasi maju untuk menghitung keluaran dan fungsi kerugian; gradien kemudian dipropagasikan ke belakang, dan bobot diperbarui, misalnya melalui penurunan gradien, untuk mengurangi kesalahan. Berbeda dengan perceptron linier, MLP dapat mempelajari batas keputusan nonlinier dan menangani data yang tidak dapat dipisahkan secara linier.

8.9 JARINGAN SARAF KONVOLUSIONAL

CNN adalah model deep learning yang dikhususkan untuk data visual seperti gambar dan video. Mereka mempelajari hierarki fitur secara otomatis melalui lapisan bertumpuk, termasuk lapisan konvolusi, aktivasi, pooling, dan akhirnya lapisan yang terhubung penuh

untuk melakukan tugas-tugas seperti klasifikasi gambar dan deteksi objek (lihat Gambar 8.9). Dengan berbagi parameter dan menggunakan teknik pengurangan spasial, CNN mendeteksi fitur tanpa memperhatikan posisi sambil menjaga ukuran model tetap terkendali.



Gambar 8.9 Arsitektur CNN.

Lapisan inti CNN adalah:

- **Lapisan Konvolusi:** Lapisan ini menerapkan filter yang dapat dipelajari (kernel) di seluruh input untuk menghasilkan peta fitur yang menekankan pola spesifik (tepi, tekstur, dan lain-lain.).
- **Lapisan Aktivasi:** Lapisan aktivasi memperkenalkan non-linearitas dengan menerapkan fungsi seperti ReLU ke peta fitur, memungkinkan jaringan untuk memodelkan hubungan yang kompleks.
- **Lapisan Pooling:** Lapisan pooling menurunkan resolusi peta fitur untuk mengurangi dimensi spasial dan komputasi sambil mempertahankan informasi penting dan meningkatkan ketahanan terhadap pergeseran kecil.
- **Lapisan Terhubung Penuh:** Lapisan terhubung penuh meratakan dan menggabungkan fitur tingkat tinggi dari lapisan sebelumnya untuk menghasilkan prediksi akhir (misalnya, skor kelas).

Karakteristik utama CNN adalah:

- **Ekstraksi Fitur Otomatis:** Filter dipelajari langsung dari data mentah, menghilangkan kebutuhan akan rekayasa fitur manual.
- **Pembelajaran Fitur Hierarkis:** Lapisan bawah menangkap pola sederhana; lapisan yang lebih dalam menyusun pola-pola ini menjadi representasi yang semakin kompleks.
- **Invariansi Spasial:** Pembagian parameter dan penggabungan memungkinkan pengenalan objek meskipun terjadi perubahan posisi atau orientasi.
- **Pembagian Parameter:** Filter yang sama digunakan kembali di seluruh input, sangat mengurangi jumlah parameter dan meningkatkan efisiensi.

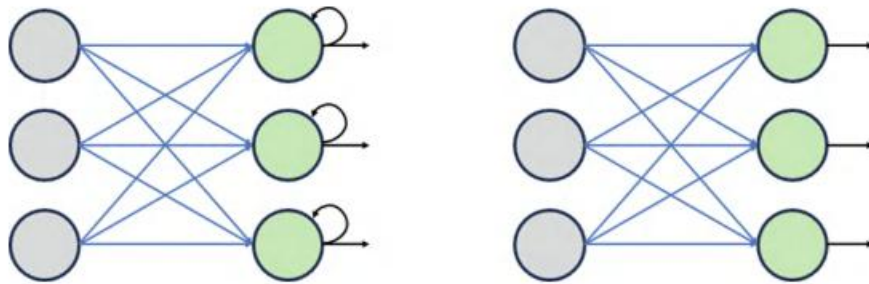
Singkatnya, CNN adalah arsitektur feed-forward yang secara spasial berisi filter konvolusional lokal yang berbagi bobot untuk mendeteksi pola di wilayah input yang tumpang tindih. Struktur ini memanfaatkan korelasi lokal dan ekuivalen translasi, memungkinkan CNN untuk

mempelajari peta fitur hierarkis dengan prapemrosesan minimal. CNN banyak digunakan untuk tugas penambangan gambar dan video, serta banyak aplikasi dalam audio, rekomendasi, dan NLP.

8.10 JARINGAN BERULANG

RNN adalah model pembelajaran mendalam yang dibangun untuk data sekuensial. Mereka mempertahankan keadaan tersembunyi internal yang membawa informasi dari input sebelumnya, memungkinkan konteks masa lalu untuk memengaruhi prediksi saat ini (lihat Gambar 8.10). Lingkaran umpan balik ini, di mana output sebelumnya (atau keadaan tersembunyi) digunakan sebagai input untuk langkah waktu berikutnya, membuat RNN cocok untuk tugas-tugas seperti pemodelan bahasa, pengenalan ucapan, dan peramalan deret waktu. Karakteristik utama RNN adalah:

- **Pemrosesan Sekuensial:** Input diproses satu elemen pada satu waktu, menjaga urutan dan konteks.
- **Memori (Keadaan Tersembunyi):** Keadaan tersembunyi menyimpan informasi tentang input sebelumnya, memungkinkan komputasi yang sadar konteks.
- **Umpan Balik:** Informasi mengalir dari satu langkah waktu ke langkah waktu berikutnya, menghubungkan komputasi masa lalu dan masa kini.
- **Bobot Bersama:** Parameter yang sama digunakan kembali pada setiap langkah waktu, memungkinkan model untuk menangani urutan dengan panjang variabel secara efisien.



Gambar 8.10 Perbandingan RNN (kiri) dan jaringan saraf umpan-maju (kanan).

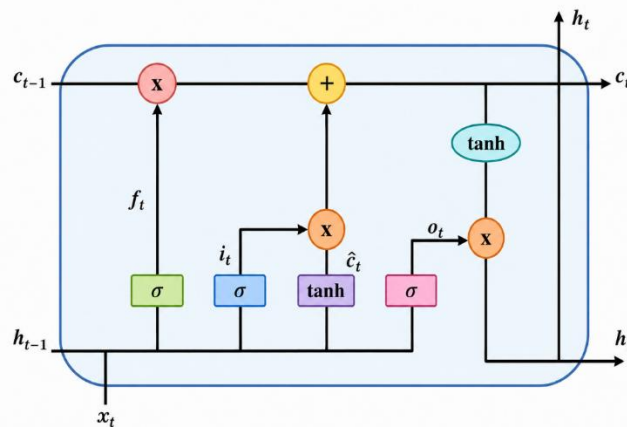
Dibangun untuk data sekuensial, RNN mempertahankan keadaan tersembunyi yang membawa informasi di seluruh langkah waktu, memungkinkan prediksi yang peka terhadap konteks. Mereka digunakan dalam pemodelan bahasa, pengenalan ucapan, dan peramalan deret waktu. Namun, mereka mengalami masalah gradien yang menghilang/meledak ketika mempelajari ketergantungan jangka panjang.

8.11 MEMORI JANGKA PANJANG (LSTM)

LSTM adalah varian RNN yang dirancang untuk mempertahankan informasi dalam urutan panjang, sehingga sangat cocok untuk tugas-tugas seperti pemodelan bahasa dan peramalan

deret waktu. Strukturnya berpusat pada sel memori ditambah tiga gerbang yang mengatur aliran informasi, yaitu lupa, masukan, dan keluaran (lihat Gambar 8.11):

- **Sel Memori:** Keadaan inti yang membawa informasi jangka panjang melintasi langkah waktu. Gerbang mengontrol apa yang ditambahkan atau dihapus.
- **Gerbang Lupa:** Menggunakan aktivasi sigmoid untuk memutuskan bagian mana dari keadaan sel sebelumnya yang akan dibuang (nilai mendekati 0 = lupakan, mendekati 1 = simpan).
- **Gerbang Masukan:** Menentukan informasi baru apa yang akan ditulis ke sel. Lapisan sigmoid memilih nilai mana yang akan diperbarui, sementara lapisan tanh menghasilkan nilai kandidat untuk ditambahkan.
- **Gerbang Keluaran:** Mengontrol bagian mana dari keadaan sel yang diekspos sebagai keluaran unit. Filter sigmoid memilih komponen, dan transformasi tanh dari keadaan sel diskalakan untuk menghasilkan vektor tersembunyi/keluaran.
- **Keadaan Tersembunyi vs. Keadaan Sel:** Keadaan sel adalah memori jangka panjang; keadaan tersembunyi adalah keluaran jangka pendek pada setiap langkah waktu dan digunakan, bersama dengan keadaan sel, untuk menghitung langkah selanjutnya.



Gambar 8.11 Struktur internal LSTM.

RNN standar kesulitan dengan ketergantungan jangka panjang karena jarak antara input dan output terkaitnya dalam suatu urutan semakin besar. Hal ini mengakibatkan jaringan sering gagal mempertahankan atau menghubungkan informasi tersebut. Ini terutama bermasalah untuk urutan teks yang panjang. Arsitektur LSTM dirancang untuk mengatasi keterbatasan ini dan mempertahankan informasi dalam rentang waktu yang jauh lebih lama.

8.12 TRANSFORMER

Transformer akan dibahas secara detail di Bab 11. Di sini, kami memberikan perbandingan singkat tentang bagaimana blok bangunan transformer berhubungan dengan RNN, CNN, dan LSTM. Transformer adalah arsitektur jaringan saraf untuk memproses data sekuensial, terutama teks. Arsitektur ini menggunakan mekanisme self-attention (dibahas pada Bab 11) untuk menentukan kepentingan relatif setiap elemen dalam masukan. Desain

tersebut memungkinkan pemodelan ketergantungan jarak jauh sehingga efektif untuk tugas seperti penerjemahan dan pembangkitan teks. Komponen utama Transformer meliputi:

- **Encoder dan Decoder:** Transformer asli adalah model seq-to-seq di mana encoder mengkodekan urutan input, dan decoder menghasilkan output. Beberapa model (misalnya, BERT) hanya menggunakan encoder; yang lain (misalnya, GPT) hanya menggunakan decoder.
- **Self-attention:** Mekanisme utama yang memungkinkan model membandingkan setiap token dengan setiap token lainnya dan menetapkan bobot relevansi, sehingga dapat menangkap hubungan jarak jauh.
- **Multi-head Attention:** Beberapa operasi self-attention berjalan secara paralel dengan proyeksi yang dipelajari berbeda, memungkinkan model untuk memperhatikan berbagai jenis hubungan secara bersamaan.
- **Positional Encoding:** Modul ini menambahkan informasi posisi ke embedding token sehingga model dapat merepresentasikan urutan token meskipun memproses urutan secara paralel.
- **Feed-Forward Networks:** Jaringan saraf posisi-per-posisi yang diterapkan setelah lapisan perhatian untuk mengubah dan menggabungkan fitur secara independen di setiap posisi.
- **Layer Normalization and Residual Connections:** Modul ini menstabilkan dan mempercepat pelatihan, dengan koneksi residual yang memudahkan aliran gradien dan normalisasi lapisan untuk menjaga aktivasi tetap terukur dengan baik.

Setelah mengeksplorasi varian arsitektur utama DNN, sekarang kita membandingkan model-model ini untuk menyoroti kesamaan, perbedaan, dan trade-off-nya. Analisis komparatif ini memberikan pemahaman yang lebih dalam tentang bagaimana setiap arsitektur selaras dengan tipe data, tugas, dan batasan komputasi tertentu.

8.13 ANALISIS KOMPARATIF ARSITEKTUR

Terlepas dari perbedaan strukturalnya, CNN, RNN, LSTM, dan Transformer memiliki tujuan yang sama: mempelajari representasi dari data. Keempat arsitektur tersebut menggunakan fungsi berlapis dan berparameter yang dilatih melalui optimasi berbasis gradien. Mereka juga memanfaatkan pembagian parameter untuk meningkatkan efisiensi dan generalisasi. Beberapa karakteristik bersama mereka meliputi:

- **Arsitektur yang Dapat Didiferensiasi:** Semua model dilatih menggunakan backpropagation dan penurunan gradien.
- **Struktur Berlapis:** Setiap arsitektur membangun representasi hierarkis melalui beberapa lapisan.
- **Pembagian Parameter:** Mengurangi jumlah parameter yang dapat dipelajari dan meningkatkan generalisasi.
- **Teknik Optimasi:** Penggunaan umum regularisasi, jadwal laju pembelajaran, dan pelatihan batch.

RNN memproses urutan secara bertahap menggunakan keadaan tersembunyi yang menyebarkan informasi melalui waktu, secara alami mengkodekan urutan temporal tetapi mengalami gradien yang menghilang/meledak seperti yang disebutkan sebelumnya. Oleh karena itu, kemampuan mereka untuk mempelajari ketergantungan jarak jauh sangat terbatas. LSTM memperluas RNN dengan sel gerbang yang mengatur aliran informasi dan mengurangi masalah pelatihan ini, memungkinkan retensi konteks jangka panjang yang jauh lebih baik sambil mempertahankan bias induktif komputasi sekuensial.

CNN, di sisi lain, mahir dalam lokalitas dan kesetaraan translasi, menggunakan filter konvolusi yang dipelajari dan pooling untuk membangun fitur spasial hierarkis yang sangat paralel dan efisien untuk data spasial terstruktur seperti gambar dan audio. Transformer mengabaikan rekurensi dan bidang reseptif lokal tetap demi perhatian diri, yang menghitung interaksi berpasangan antara semua posisi dalam suatu urutan dan oleh karena itu memodelkan ketergantungan jarak jauh secara langsung dan fleksibel sambil memungkinkan paralelisme masif selama pelatihan. Dalam praktiknya, pilihan di antara arsitektur ini melibatkan pertimbangan antara bias induktif, efisiensi komputasi, paralelisasi, dan jumlah data serta daya komputasi yang tersedia. LSTM lebih unggul daripada RNN untuk urutan panjang, CNN tetap efisien untuk pola spasial lokal, dan Transformer mendominasi banyak tugas urutan skala besar.

8.14 RINGKASAN

Dalam bab ini, dibahas konsep-konsep dasar jaringan saraf, dengan fokus pada komponen inti, fungsi aktivasi, varian arsitektur, dan dinamika pelatihan. Jaringan saraf merupakan aproksimator fungsi berlapis; fungsi aktivasi memperkenalkan nonlinieritas dan memengaruhi aliran gradien sehingga jaringan mampu memodelkan hubungan yang kompleks dalam data. Pembahasan diawali dengan struktur matematis neuron individual serta peran fungsi aktivasi sigmoid, tanh, ReLU, dan softmax. Fungsi-fungsi tersebut menentukan cara neuron merespons masukan dan meneruskan sinyal di dalam jaringan. Berlandaskan fondasi tersebut, bab ini memperkenalkan ANN, khususnya MLP, serta menjelaskan bagaimana propagasi maju dan propagasi balik memungkinkan pembelajaran melalui pembaruan bobot secara iteratif. Selanjutnya, dibahas varian arsitektur ANN, termasuk jaringan umpan-maju, rekuren, dan umpan balik, dengan menekankan pengaruh pola interkoneksi terhadap perilaku komputasi.

Selanjutnya, dibahas hiperparameter, seperti laju pembelajaran, ukuran batch, jumlah lapisan, dan fungsi aktivasi, beserta strategi penalaannya untuk mengoptimalkan kinerja model. Terakhir, kami memperkenalkan DNN dan varian arsitekturnya: MLP, CNN, RNN, LSTM, dan Transformer. Setiap arsitektur cocok untuk tipe data dan tugas tertentu, dengan pertimbangan dalam hal efisiensi komputasi, paralelisasi, dan bias induktif. Sementara CNN unggul dalam pengenalan pola spasial, LSTM dan Transformer lebih cocok untuk data sekuensial dan ketergantungan jarak jauh. Bersama-sama, konsep-konsep ini menyediakan kerangka kerja komprehensif untuk memahami dan merancang sistem AI berbasis NN.

Landasan ini mempersiapkan kita untuk mengeksplorasi topik yang lebih lanjut seperti RL dan NLP di bab-bab selanjutnya.

8.15 PERTANYAAN

1. Jelaskan definisi perceptron dan uraikan peran bobot, bias, dan fungsi aktivasi dalam operasinya.
2. Bandingkan fungsi aktivasi sigmoid, tanh, ReLU, dan softmax. Jelaskan rentang keluaran masing-masing serta kasus penggunaan yang lazim.
3. Uraikan proses propagasi maju dan propagasi mundur (backpropagation) dalam Multilayer Perceptron (MLP).
4. Identifikasi hiperparameter utama pada jaringan saraf tiruan (ANN) dan jaringan saraf dalam (DNN). Ringkas strategi yang efektif untuk melakukan penalaan hiperparameter.
5. Kontraskan arsitektur jaringan saraf berikut: MLP, jaringan saraf konvolusional (CNN), jaringan saraf berulang (RNN), memori jangka pendek panjang (LSTM), dan Transformer. Untuk masing-masing, jelaskan ide inti dan kasus penggunaan yang paling sesuai.
6. [Latihan Pemrograman - Dasar] Implementasikan langkah propagasi maju sebuah perceptron menggunakan fungsi aktivasi sigmoid, tanh, ReLU, dan softmax vektor. Tunjukkan bagaimana komponen bias memengaruhi batas keputusan.
7. [Latihan Pemrograman - Lanjutan] Kembangkan MLP dua lapis minimal (masukan -> lapisan tersembunyi ReLU -> keluaran sigmoid), kemudian latih menggunakan fungsi kerugian entropi silang dengan perhitungan gradien secara eksplisit.
8. Fungsi aktivasi mana yang berpusat pada nol dan berkisar dari -1 hingga 1?
 - A. Sigmoid
 - B. tanh
 - C. ReLU
 - D. Softmax
9. Keterbatasan utama RNN vanilla yang disorot adalah
 - A. Ketidakmampuan untuk menangani urutan dengan panjang variabel
 - B. Gradien yang menghilang/meledak pada ketergantungan yang panjang
 - C. Kurangnya pembagian parameter antar waktu
 - D. Ketidakcocokan dengan data teks
10. Metode pencarian hiperparameter mana yang menyeluruh pada grid yang telah ditentukan dan skalanya buruk dengan banyak dimensi?
 - A. Pencarian manual
 - B. Pencarian acak
 - C. Pencarian grid
 - D. Optimasi Bayesian

BAB 9

PEMBELAJARAN PENGUATAN

9.1 PEMBELAJARAN PENGUATAN

Pembelajaran Penguatan atau *Reinforcement Learning (RL)* adalah cabang dari Kecerdasan Buatan (Artificial Intelligence atau AI) yang berfokus pada pelatihan *agen* untuk membuat keputusan dengan berinteraksi dengan *lingkungannya*. Dalam RL, agen merasakan lingkungannya melalui *ruang keadaan* dan *ruang tindakan*, melakukan tindakan, bertransisi ke keadaan baru, dan menerima *imbalan* sebagai umpan balik. Tujuan agen adalah mempelajari *kebijakan* optimal yang memaksimalkan imbalan kumulatif dari waktu ke waktu. Untuk lebih memahami RL, akan sangat membantu jika membandingkannya dengan pembelajaran mesin (machine learning atau ML) tradisional. Meskipun keduanya merupakan subbidang AI, keduanya berbeda dalam paradigma dan tujuan pembelajaran mereka:

- Algoritma **ML** belajar dari data untuk meningkatkan kinerja. ML biasanya dikategorikan menjadi:
 - **Pembelajaran Terawasi (Supervised Learning):** Model dilatih pada kumpulan data berlabel untuk memprediksi hasil.
 - **Pembelajaran Semi-Terawasi (Semi-Supervised Learning):** Menggabungkan sejumlah kecil data berlabel dengan sejumlah besar data tidak berlabel.
 - **Pembelajaran Tak Terawasi (Unsupervised Learning):** Model mempelajari pola dan struktur dari data tanpa label, misalnya untuk pengelompokan atau reduksi dimensi.
- Sebaliknya, **RL** tidak bergantung pada kumpulan data berlabel. Sebaliknya, ia melatih agen untuk membuat keputusan dengan berinteraksi dengan lingkungan. Agen menerima *imbalan* berdasarkan tindakannya dan menggunakan umpan balik ini untuk meningkatkan strategi pengambilan keputusannya. Lingkungan dapat berupa nyata atau simulasi, dan tujuan agen adalah mencapai *keadaan tujuan* yang menghasilkan imbalan kumulatif tertinggi.

RL memiliki keunggulan khusus pada tugas yang menuntut pencapaian keadaan tujuan melalui rangkaian keputusan. Kerangka ini dapat digunakan untuk memecahkan beragam masalah pengambilan keputusan sekuensial. RL sangat bermanfaat pada domain yang melibatkan pengambilan keputusan sekuensial dan lingkungan dinamis. Beberapa penerapannya meliputi:

- **Robotika:** RL memungkinkan robot mempelajari tugas seperti manipulasi objek, perakitan, dan navigasi dalam lingkungan kompleks. Melalui interaksi dan umpan balik, robot dapat beradaptasi terhadap rintangan dan perubahan kondisi.
- **Mengemudi Otonom:** Algoritma RL membantu kendaraan mempelajari strategi berkendara, termasuk pengendalian kecepatan, kemudi, dan pengereman, dengan mempertimbangkan lalu lintas, kondisi jalan, dan keselamatan.

- **Pemrosesan Bahasa Alami:** RL digunakan untuk melatih agen percakapan, mengoptimalkan ringkasan teks, dan mempersonalisasi rekomendasi berdasarkan interaksi pengguna.
- **Perawatan Kesehatan:** RL mendukung pembedahan presisi, diagnosis penyakit, dan perencanaan perawatan dengan memungkinkan agen AI untuk membuat keputusan yang tepat dalam lingkungan yang tidak pasti.
- **Permainan:** Dalam aplikasi permainan video, tujuan agen adalah memenangkan permainan dengan mencapai jumlah poin tertinggi yang mungkin. Batasan ini membantu agen dalam menemukan pendekatan baru sambil menavigasi menuju tujuannya.

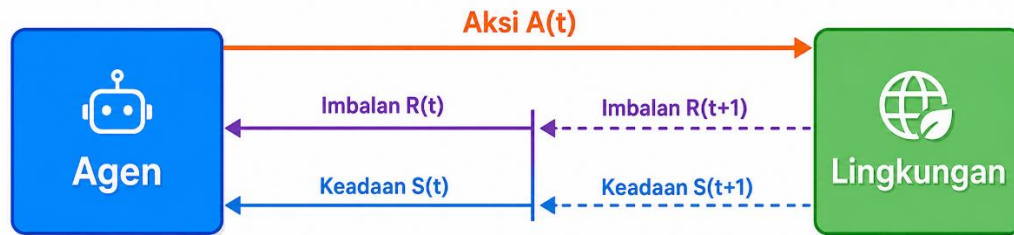
Penting untuk diingat bahwa baik manusia maupun agen tidak dapat merasakan seluruh dunia sekaligus. Apa pun yang dirasakan dari lingkungan oleh agen akan diwujudkan menjadi suatu keadaan di mana agen melakukan tindakan. Hal ini mengakibatkan agen berada dalam keadaan baru yang berbeda dari keadaan sebelumnya. Ini menciptakan serangkaian ruang keadaan yang dapat terbatas atau tak terbatas. Setelah menetapkan konsep dan aplikasi dasar RL, sekarang kita akan beralih ke terminologi dan konvensi penting yang mendasari sistem RL. Pemahaman yang jelas tentang elemen-elemen ini sangat penting untuk memahami mekanisme bagaimana agen berinteraksi dengan lingkungan, membuat keputusan, dan mempelajari perilaku optimal dari waktu ke waktu.

9.2 ISTILAH DASAR RL

RL melibatkan serangkaian konsep inti yang mendefinisikan bagaimana agen beroperasi dalam suatu lingkungan. Konsep-konsep ini merupakan dasar untuk memahami algoritma RL dan akan digunakan secara konsisten di seluruh bab ini dan bab berikutnya.

- **Agen:** Pembelajar atau pengambil keputusan yang berinteraksi dengan lingkungan. Agen mengamati keadaan saat ini, memilih tindakan, dan menerima umpan balik berupa imbalan.
- **Lingkungan:** Sistem eksternal tempat agen berinteraksi. Sistem ini dapat berupa nyata atau simulasi dan berisi aturan yang mengatur transisi keadaan dan mekanisme imbalan.
- **Keadaan:** Representasi lingkungan pada momen tertentu. Keadaan dapat berupa diskrit atau kontinu, terbatas atau tak terbatas, dan digunakan oleh agen untuk menentukan tindakan selanjutnya.
- **Tindakan:** Setiap operasi yang dapat dilakukan agen dalam keadaan tertentu. Himpunan semua tindakan yang mungkin membentuk *ruang tindakan*, yang juga dapat terbatas atau tak terbatas.
- **Imbalan (Reward):** Sinyal umpan balik skalar yang diterima agen setelah melakukan tindakan. Agen berupaya memaksimalkan imbalan kumulatif dengan mempertimbangkan imbalan langsung dan imbalan masa depan.
- **Tripel SAR:** Unit interaksi dasar dalam RL yang terdiri atas *State* (s), *Action* (a), dan *Reward* (r). Tripel ini merekam pengalaman agen pada setiap langkah waktu.

- **Episode:** Urutan interaksi dari keadaan awal ke keadaan akhir (tujuan). Setiap episode mewakili percobaan lengkap dari proses pembelajaran agen.



Gambar 9.1 Terminologi dan konvensi yang digunakan dalam RL.

- **Tujuan:** Keadaan akhir yang menghasilkan imbalan kumulatif tertinggi. Mencapai tujuan menyelesaikan sebuah episode dan memberikan umpan balik untuk peningkatan kebijakan.
- **Kebijakan (π):** Strategi yang memetakan keadaan ke tindakan. Kebijakan dapat berupa:
 - **Deterministik:** Selalu memilih tindakan yang sama untuk keadaan tertentu.
 - **Stokastik:** Memilih tindakan berdasarkan distribusi probabilitas.
- **Fungsi Nilai (V_π):** Perkiraan imbalan kumulatif yang diharapkan dari suatu keadaan dengan asumsi agen mengikuti kebijakan π . Fungsi ini membantu menilai tingkat desirabilitas keadaan dan memandu pengambilan keputusan agen.

Fungsi nilai secara matematis dinyatakan sebagai:

$$V^\pi(s) = \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots | S_t = s]$$

Persamaan ini menangkap jumlah *terdiskon* dari *imbalan masa depan*, di mana:

- R_t adalah imbalan pada waktu t ,
- γ adalah *faktor diskon* ($0 \leq \gamma \leq 1$) yang menentukan bobot imbalan masa depan.

Gambar 9.1 merangkum konsep-konsep ini secara visual, menunjukkan bagaimana suatu agen bertransisi di antara berbagai keadaan, melakukan tindakan, dan menerima imbalan, membentuk serangkaian rangkap tiga SAR. Mari kita berasumsi sebuah agen mulai dari keadaan $S(t)$, melakukan tindakan $A(t)$ dan dengan demikian, menerima imbalan $R(t+1)$, serta mendarat pada keadaan baru $S(t+1)$. Pasangan keadaan dan tindakan saat ini memberikan imbalan untuk langkah berikutnya. Karena $S(t)$ dan $A(t)$ menghasilkan $S(t+1)$, kita memiliki rangkap tiga baru (keadaan saat ini, tindakan, keadaan baru) yaitu $[S(t), A(t), S(t+1)]$ atau (s, a, s') .

9.3 MENENTUKAN STRATEGI TERBAIK

Dalam RL, tujuan agen adalah untuk memaksimalkan imbalan di masa depan melalui pengambilan keputusan yang cerdas. Untuk mengevaluasi seberapa baik kinerja agen, kita menggunakan dua *kriteria optimalitas* utama:

- Fungsi nilai
- Kebijakan

Kedua kriteria tersebut membantu mengukur efektivitas perilaku agen dan memandu proses pembelajarannya.

- **Fungsi Nilai sebagai Kriteria Optimalitas:** *Fungsi nilai* memberikan nilai numerik pada setiap keadaan, yang mewakili imbalan kumulatif yang diharapkan yang dapat diperoleh agen dari keadaan tersebut dan seterusnya. Ini termasuk imbalan langsung dan masa depan, yang didiskon dari waktu ke waktu.

Konsep *imbalan tertunda* sangat penting dalam RL: Agen harus mempertimbangkan tidak hanya imbalan langsung tetapi juga konsekuensi jangka panjang dari tindakan mereka. Fungsi nilai menangkap hal ini dengan memperkirakan total pengembalian yang diharapkan dari keadaan tertentu. Misalkan $V(s)$ menunjukkan nilai keadaan s . Nilainya didefinisikan sebagai:

$$V(s) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right]$$

di mana:

- R_{t+k+1} adalah imbalan yang diterima pada waktu $t + k + 1$,
- γ adalah *faktor diskon* ($0 \leq \gamma \leq 1$), yang menentukan pentingnya imbalan di masa depan.

Nilai $V(s)$ yang lebih tinggi menunjukkan bahwa keadaan tersebut lebih diinginkan oleh agen. Tidak seperti imbalan, yang tetap untuk setiap tindakan, nilai suatu keadaan dapat berubah secara dinamis berdasarkan pengalaman agen dan kebijakan yang diikuti.

- **Kebijakan sebagai Kriteria Optimalitas:** Suatu *kebijakan* mendefinisikan perilaku agen dengan memetakan keadaan ke tindakan. Kebijakan tersebut dapat berupa:
 - **Deterministik:** Selalu memilih tindakan yang sama untuk keadaan tertentu.
 - **Stokastik:** Memilih tindakan berdasarkan distribusi probabilitas.

Kebijakan tersebut dilambangkan dengan π , dan probabilitas mengambil tindakan a dalam keadaan s dinyatakan sebagai:

$$\pi(a \mid s) = P(A_t = a \mid S_t = s)$$

Kebijakan optimal memaksimalkan imbalan kumulatif yang diharapkan dari setiap keadaan. Kebijakan ini bekerja bersamaan dengan fungsi nilai. Sementara fungsi nilai mengevaluasi seberapa baik suatu keadaan, kebijakan memandu agen tentang tindakan apa yang harus diambil dalam keadaan tersebut. Bersama-sama, *fungsi nilai* dan *kebijakan* membentuk dasar untuk memecahkan masalah RL. Keduanya memungkinkan agen untuk mempelajari strategi yang menyeimbangkan keuntungan jangka pendek dengan keberhasilan jangka panjang.

9.4 KELEBIHAN DAN TANTANGAN RL

RL menawarkan kerangka kerja yang ampuh untuk mengembangkan agen cerdas yang mampu mempelajari perilaku optimal melalui interaksi dengan lingkungan dinamis. Meskipun RL telah menunjukkan keberhasilan yang luar biasa di berbagai domain, ia juga menghadirkan tantangan unik yang harus diatasi untuk penerapan yang efektif:

- **Manfaat RL:** RL menawarkan beberapa manfaat:
 - **Maksimisasi Imbalan Jangka Panjang:** RL berfokus pada pengoptimalan imbalan kumulatif dari waktu ke waktu daripada keuntungan jangka pendek. Perspektif jangka panjang ini memungkinkan agen untuk mengembangkan strategi yang lebih berkelanjutan dan efektif.
 - **Tidak Perlu Data Pelatihan yang Dikumpulkan Sebelumnya:** Tidak seperti pembelajaran terawasi, RL tidak memerlukan fase pengumpulan data terpisah. Agen belajar langsung dari pengalaman mereka dengan berinteraksi dengan lingkungan, menjadikan RL sangat cocok untuk tugas-tugas di mana data dihasilkan secara waktu nyata.
 - **Kemampuan Beradaptasi dengan Lingkungan Dinamis dan Tidak Pasti:** Agen RL secara inheren dirancang untuk beradaptasi dengan kondisi yang berubah. Mereka terus-menerus menyempurnakan strategi mereka berdasarkan umpan balik, sehingga sangat cocok untuk lingkungan dengan dinamika yang tidak dapat diprediksi atau terus berkembang.
 - **Keselarasan yang Lebih Dekat dengan Kecerdasan Buatan Umum (AGI):** Agen RL menunjukkan perilaku yang berorientasi pada tujuan dan pembelajaran otonom, karakteristik yang selaras dengan visi AGI yang lebih luas. Kemampuan mereka untuk mengeksplorasi, belajar, dan mengoptimalkan tanpa pengawasan eksplisit menjadikan RL sebagai pendekatan yang menjanjikan untuk sistem cerdas tujuan umum.
- **Tantangan RL:** Meskipun berhasil dalam memecahkan masalah kompleks, algoritma RL memiliki beberapa tantangan:
 - **Persyaratan Pengalaman yang Luas:** Agen RL membutuhkan interaksi yang signifikan dengan lingkungannya untuk mempelajari kebijakan yang efektif. Di lingkungan dengan latensi tinggi atau aksesibilitas terbatas, hal ini dapat memperlambat proses pembelajaran.
 - **Imbalan yang Tertunda:** Dalam banyak skenario RL, imbalan tidak langsung. Agen harus melakukan serangkaian tindakan sebelum menerima umpan balik, yang mempersulit pembelajaran kebijakan optimal – terutama ketika hubungan antara tindakan dan hasil tidak langsung.
 - **Kurangnya Interpretasi:** Agen RL membuat keputusan berdasarkan pengalaman yang terakumulasi dan kebijakan yang dipelajari, yang mungkin tidak transparan bagi pengamat eksternal sehingga dapat menghambat kepercayaan dan penerimaan, terutama di domain yang kritis terhadap keselamatan seperti perawatan kesehatan atau sistem otonom.

Untuk menerapkan RL secara efektif, penting untuk memahami ciri-ciri utamanya:

- **Umpan Balik Tertunda:** Imbalan sering diterima setelah beberapa langkah waktu sehingga agen perlu mengaitkan tindakan dengan konsekuensi jangka panjang.
- **Pengambilan Keputusan Sekuensial:** Waktu berperan penting karena agen harus membuat rangkaian keputusan yang memengaruhi keadaan dan imbalan pada masa depan.
- **Data yang Bergantung pada Lingkungan:** Tindakan agen secara langsung memengaruhi data yang diterimanya, menciptakan lingkaran umpan balik antara perilaku dan pembelajaran.
- **Pengawasan Minimal:** RL bergantung pada sinyal umpan balik yang jarang daripada data berlabel, sehingga memungkinkan agen untuk belajar secara otonom.

Dengan pemahaman tentang konsep inti RL, manfaat dan tantangannya, serta karakteristik utamanya, kita sekarang akan membahas dasar-dasar matematika yang mendasari algoritma RL. Dasar-dasar ini menyediakan struktur formal yang diperlukan untuk memodelkan interaksi agen-lingkungan, mengukur imbalan, dan menurunkan strategi optimal untuk pengambilan keputusan. Inti dari RL terletak pada kerangka kerja matematika yang memformalkan bagaimana agen berinteraksi dengan lingkungannya, membuat keputusan, dan belajar dari umpan balik. Bagian selanjutnya akan membahas kembali elemen-elemen kunci RL – keadaan, tindakan, imbalan, kebijakan, dan fungsi nilai – dan memperkenalkan konsep-konsep matematika kunci untuk lingkungan RL.

9.5 MODEL PENGAMBILAN KEPUTUSAN

Markov Decision Process (MDP) didasarkan pada properti Markov. Pertama, kita akan membahas definisi proses Markov. Urutan keadaan bersifat Markov jika dan hanya jika probabilitas untuk berpindah ke keadaan berikutnya S_{t+1} hanya bergantung pada keadaan saat ini S_t dan bukan pada keadaan-keadaan sebelumnya $S_1, S_2, \dots, S_{t-1}$. Secara tepat, untuk semua t , MDP dapat didefinisikan sebagai

$$P(S_{t+1} | S_t) = P(S_{t+1} | S_1, S_2, \dots, S_t) \quad (9.1)$$

Sesuai dengan sifat Markov, untuk mengetahui informasi pada waktu $t + 1$ satu-satunya hal yang penting adalah informasi pada waktu t . Di sini, lingkungan dianggap stasioner, yang berarti bahwa aturan dalam lingkungan tersebut tetap. Pemahaman intuitif dari sifat ini dapat dijelaskan dengan contoh helikopter otonom. Misalnya, langkah selanjutnya bagi helikopter untuk bergerak adalah ke kanan, atau ke kiri, untuk menukik, atau untuk berguling. Ini sepenuhnya bergantung pada posisi helikopter saat ini, bukan di mana posisinya beberapa menit sebelumnya.

Untuk keadaan *Markov* S dan keadaan penerus S' , fungsi probabilitas transisi keadaan didefinisikan sebagai

$$\rho_{SS'} = P[S_{t+1} = S' | S_t = S] \quad (9.2)$$

Distribusi probabilitas ini menangkap gagasan bahwa agen dalam suatu keadaan dapat berpindah ke keadaan lain dengan probabilitas tertentu dan dapat diekspresikan sebagai matriks, di mana jumlah setiap barisnya sama dengan 1.

$$\rho = \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \quad (9.3)$$

Selanjutnya, kami akan menyajikan ekspresi untuk imbalan masa depan yang didiskon.

9.6 PENGEMBALIAN DAN FAKTOR DISKON

Tujuan agen RL adalah untuk memilih tindakan dari waktu ke waktu sehingga memaksimalkan nilai harapan pengembalian. Tujuan agen adalah untuk memilih kebijakan optimal. Pengembalian G_t adalah total imbalan yang didiskon dari langkah waktu t yang dapat dinyatakan sebagai

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{n-1} R_{t+n}$$
$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (9.4)$$

Imbalan masa depan yang didiskon dapat ditafsirkan sebagai nilai saat ini dari imbalan di masa depan. Di sini, faktor diskon memberikan bobot yang lebih besar pada imbalan langsung daripada imbalan yang tertunda. γ dapat berupa 0 (berpandangan dekat) atau 1 (berpandangan jauh). Faktor diskon digunakan di sini untuk mencegah hasil yang tak terhingga dalam proses Markov siklik dan juga untuk memperhitungkan ketidakpastian dalam imbalan masa depan.

9.7 APA KITA BUTUH DISKON?

Secara matematis, mendiskon imbalan sangat memudahkan untuk menjamin bahwa algoritma akan konvergen dan menghindari pengembalian tak terbatas dalam proses Markov. Perhatikan di sini bahwa, meskipun fungsi pengembalian G_t adalah jumlah dari sejumlah tak terbatas suku, fungsi tersebut tetap terbatas jika imbalannya bukan nol dan konstan. Misalnya, ketika imbalannya +1 dan faktor diskon, $\gamma < 1$, maka pengembaliannya diberikan oleh:

$$G_t = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1 - \gamma} \quad (9.5)$$

Mengingat ketidakpastian, imbalan didiskon, dan agen mengambil imbalan langsung tanpa harus menunggu imbalan yang lebih besar di masa depan.

9.8 CONTOH: BELAJAR BERJALAN

Mari kita kerjakan contoh sederhana untuk mengilustrasikan semua dasar matematika RL yang telah kita pelajari sejauh ini. Pertimbangkan skenario di mana kita ingin mengajarkan bayi berjalan. Skenario ini tidak dapat sepenuhnya diawasi. Namun, skenario ini dapat dimodelkan sebagai masalah RL. Seorang bayi (agen RL dalam skenario ini) dapat belajar berjalan dengan mengambil langkah-langkah kecil seperti bergerak satu langkah ke depan, bergerak satu langkah ke belakang, jatuh dan berguling, lalu bangun dan berdiri. Lingkungan untuk bayi adalah lantai rumah dengan rintangan. Karena lingkungan tidak dapat diprediksi, bayi dapat berada dalam keadaan apa pun (berjalan satu langkah ke depan dengan kaki kanan, berjalan satu langkah ke depan dengan kaki kiri, jatuh ke lantai, bangun dan berdiri, dan lain-lain.) pada titik waktu mana pun. Tujuan bayi adalah berjalan beberapa langkah tanpa jatuh dan mendekati orang tuanya. Bayi tersebut melakukan berbagai tindakan seperti melangkah maju dengan kaki kanan, melangkah maju dengan kaki kiri, jatuh, dan bangun. Mari kita asumsikan bahwa setiap kali bayi melakukan suatu tindakan, ia mendapatkan imbalan. Dalam hal ini, tidak ada imbalan negatif, tetapi imbalan positif dapat berupa pemberian permen kepada bayi jika ia melangkah maju dengan kaki kanan atau kiri, atau mencapai keadaan tujuan, atau ketika ia bangun dan berdiri.

Masalah RL ini dapat dimodelkan sebagai masalah MDP. Dalam masalah ini, keadaan diskrit dan tindakan dijelaskan di bawah ini. (*Catatan:* Di sini kami hanya mempertimbangkan sebagian kecil dari keadaan dan tindakan diskrit yang mungkin terjadi dalam skenario ini untuk menjaga agar ilustrasi ini tetap sederhana.)

- **Keadaan** = {"*Jatuh*", "*Berdiri*", "*Langkah kaki kanan ke depan*", "*Langkah kaki kiri ke depan*"}
- **Tindakan** = {"*Langkah ke depan dengan kaki kanan*", "*Langkah ke depan dengan kaki kiri*", "*Jatuh*", "*Bangun*"}

Tabel 9.1 mendefinisikan matriks probabilitas transisi keadaan. Dalam Tabel 9.1, perhatikan bahwa jumlah baris sama dengan 1. Pembaca harus menafsirkan data yang diberikan dalam Tabel 9.1 sebagai berikut:

Jika keadaan bayi saat ini adalah *jatuh*, maka probabilitas bayi untuk *bangun dan berdiri* adalah 0, 7. Jika bayi saat ini dalam keadaan *berdiri*, maka ada probabilitas yang sama bahwa bayi akan berpindah ke keadaan *Langkah ke depan dengan kaki kanan* atau berpindah ke keadaan *Langkah ke depan dengan kaki kiri*. Probabilitas transisi dalam kedua kasus tersebut adalah 0, 3.

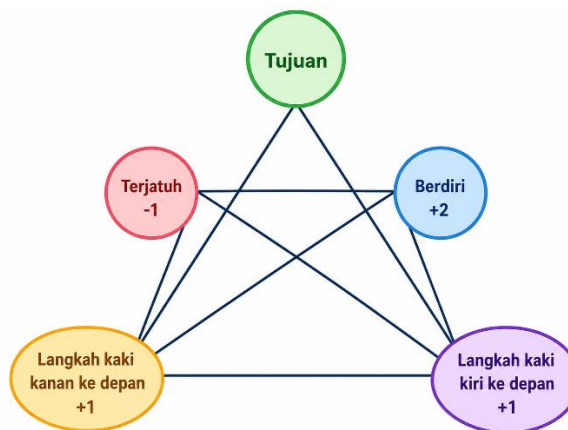
Langkah selanjutnya adalah menetapkan hadiah untuk setiap tindakan yang dilakukan oleh bayi. Keadaan tujuan di mana bayi harus mengambil beberapa langkah untuk mencapai orang tuanya dapat diberi hadiah +10 yang berarti bayi menerima 10 permen. Setiap kali bayi jatuh, kita dapat mengurangi 1 permen dari bayi. Oleh karena itu, hadiah dalam keadaan "*jatuh*" dapat berupa -1. Jika bayi bangun dan berdiri setelah jatuh, kita dapat memberi bayi 2 permen, *yaitu* hadiah +2. Terakhir, untuk setiap langkah yang diambil bayi untuk bergerak maju, baik

dengan kaki kanan atau kiri, ia dapat diberi hadiah permen, yaitu hadiah +1. Gambar 9.2 menunjukkan rantai Markov dengan keadaan dan hadiah yang terkait dengan setiap keadaan. Tapi antara keadaan rantai Markov semuanya dua arah.

Mari kita tentukan faktor diskon yang menunjukkan berapa banyak diskon yang berlaku untuk hadiah di masa mendatang ketika suatu tindakan dilakukan. Faktor diskon harus diberi nilai antara 0 dan 1. Faktor diskon dikalikan dengan hadiah untuk setiap langkah yang diambil bayi untuk mendapatkan hadiah yang didiskon. Jika $\gamma = 0.9$, maka hadiah pada waktu $t + 1$ dalam keadaan tujuan adalah $0.9 \times 10 = 9$ permen sedangkan hadiah pada waktu t dalam keadaan tujuan adalah 10 permen. Semakin besar γ , semakin kecil diskonnya. Nilai γ yang lebih tinggi memastikan bahwa agen selalu fokus pada hadiah jangka panjang.

TABEL 9.1 Matriks Probabilitas Transisi Keadaan

Keadaan	Jatuh	Berdiri	Langkah Maju dengan Kaki Kanan	Langkah Maju dengan Kaki Kiri
Jatuh	0.2	0.7	0.05	0.05
Berdiri	0.3	0.1	0.3	0.3
Langkah maju dengan kaki kanan	0.4	0.1	0.1	0.4
Langkah maju dengan kaki kiri	0.4	0.1	0.4	0.1



Gambar 9.2 Rantai Markov dengan informasi keadaan dan imbalan.

Berdasarkan informasi yang diberikan di atas, mari kita asumsikan bahwa bayi tersebut berpindah dari satu keadaan ke keadaan lain seperti yang dijelaskan di bawah ini:

Jatuh → *Berdiri* → *Melangkah maju dengan kaki kanan* →
Melangkah maju dengan kaki kiri → *Melangkah maju dengan kaki kanan*

Total pengembalian diskon G_t pada waktu t dengan asumsi bayi akan berada dalam keadaan jatuh pada waktu $t + 1$ dan $\gamma = 0.9$ diberikan sebagai

$$G_t = (-1) + 0.9 * (2) + (0.9)^2 * (1) + (0.9)^3 * (1) + (0.9)^4 * (1) = 3.9951$$

Contoh ini mengilustrasikan bagaimana agen RL mengakumulasi imbalan dari waktu ke waktu dan bagaimana faktor diskon memengaruhi penilaian hasil di masa depan. Dengan MDP dan konsep pengembalian yang didiskon, kita sekarang mengeksplorasi dua konstruksi sentral yang memandu perilaku agen: *fungsi kebijakan dan nilai*. Elemen-elemen ini sangat penting untuk mengevaluasi dan meningkatkan strategi pengambilan keputusan dalam sistem RL.

9.9 FUNGSI KEBIJAKAN DAN NILAI

Suatu kebijakan π adalah distribusi atas tindakan yang diberikan oleh keadaan dan didefinisikan sebagai:

$$\pi(a | s) = P(A_t = a | S_t = s) \quad (9.6)$$

Kebijakan bersifat independen terhadap waktu dan memandu pilihan tindakan agen pada keadaan tertentu. Untuk menemukan kebijakan optimal, kita mendefinisikan fungsi nilai keadaan dan fungsi nilai tindakan. **Fungsi Nilai Keadaan $v_\pi(s)$** : Fungsi nilai keadaan v_π adalah pengembalian yang diharapkan mulai dari keadaan s dan kemudian mengikuti kebijakan π :

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] \quad (9.7)$$

Karena $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{n-1} R_{t+n}$. Persamaan 9.7 sekarang menjadi:

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{n-1} R_{t+n} | S_t = s] \quad (9.8)$$

Persamaan 9.8 dapat disederhanakan lebih lanjut:

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \dots + \gamma^{n-2} R_{t+n}) | S_t = s] \quad (9.9)$$

Karena $G_{t+1} = R_{t+2} + \gamma R_{t+3} + \dots + \gamma^{n-2} R_{t+n}$, persamaan 9.9 menjadi:

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \quad (9.10)$$

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \quad (9.11)$$

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} | S_t = s] + \mathbb{E}_\pi[\gamma v_\pi(S_{t+1}) | S_t = s]$$

Dalam persamaan 9.11, $\mathbb{E}_\pi[R_{t+1} | S_t = s]$ adalah imbalan langsung dan $\mathbb{E}_\pi[\gamma v_\pi(S_{t+1}) | S_t = s]$ adalah nilai diskon dari keadaan penerus.

Perhatikan Gambar 9.2. Di sini, mari kita tentukan nilai keadaan *Jatuh*; dengan diketahui bahwa nilai keadaan *tujuan, berdiri, langkah maju dengan kaki kanan, dan langkah*

maju dengan kaki kiri masing-masing adalah 0, 5, 3, dan 3. Nilai keadaan *Jatuh* dapat ditentukan sebagai

$$v_{\pi}(\text{Fallen}) = (-1) + 0.7(5) + 0.05(3) + 0.05(3) = 2.8$$

Fungsi Nilai Aksi $q_{\pi}(s, a)$: Fungsi nilai aksi $q_{\pi}(s, a)$ adalah pengembalian yang diharapkan dimulai dari keadaan s , melakukan aksi a dan mengikuti kebijakan π yang diberikan sebagai:

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t \mid S_t = s, A_t = a] \quad (9.12)$$

Persamaan 9.12 dapat diuraikan untuk mendapatkan yang berikut ini

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a] \quad (9.13)$$

Mari kita definisikan

$$R_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a] \quad (9.14)$$

Sekarang kita dapat melihat bahwa hubungan antara $v_{\pi}(s)$ dan $q_{\pi}(s, a)$ (menggabungkan Persamaan 9.13 dan 9.14)

$$v_{\pi}(s) = \sum_{a \in A} \pi(a \mid s) q_{\pi}(s, a) \quad (9.15)$$

$$q_{\pi}(s, a) = R_s^a + \gamma \sum_{s' \in S} p_{ss'}^a v_{\pi}(s') \quad (9.16)$$

Dengan menggabungkan Persamaan 9.15 dan 9.16, kita mendapatkan persamaan Bellman untuk v_{π} .

$$v_{\pi}(s) = \sum_{a \in A} \pi(a \mid s) \left(R_s^a + \gamma \sum_{s' \in S} p_{ss'}^a v_{\pi}(s') \right) \quad (9.17)$$

Persamaan Bellman menghubungkan fungsi nilai keadaan dari satu keadaan dengan fungsi nilai keadaan lainnya. Demikian pula, kita juga memiliki persamaan Bellman untuk $q_{\pi}(s, a)$ yang diberikan sebagai

$$q_{\pi}(s, a) = R_s^a + \gamma \sum_{s' \in S} p_{ss'}^a \times \sum_{a' \in A} \pi(a' | s') q_{\pi}(s', a') \quad (9.18)$$

Untuk mengevaluasi dan meningkatkan kebijakan secara efektif dalam RL, kita bergantung pada hubungan rekursif yang menghubungkan nilai suatu keadaan dengan keadaan penerusnya. Hubungan ini diformalkan melalui *persamaan Bellman*, yang berfungsi sebagai tulang punggung matematika dari banyak algoritma RL.

9.10 PERSAMAAN BELLMAN

Dalam RL, tujuan agen adalah memaksimalkan jumlah imbalan dari setiap keadaan yang dikunjungi. Untuk itu, diperlukan fungsi nilai optimal, yaitu nilai maksimum dari imbalan kumulatif. Melalui persamaan Bellman, fungsi nilai dapat diuraikan menjadi dua komponen: imbalan langsung R_{t+1} dan nilai terdiskonto dari keadaan penerus $\gamma V(S_{t+1})$. Persamaan Bellman diberikan oleh

$$v(s) = \mathbb{E}[G_t | S_t = s] \quad (9.19)$$

Mengingat bahwa

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{n-1} R_{t+n} \quad (9.20)$$

Kita dapat mendefinisikan (dengan menggabungkan Persamaan 9.19 dan 9.20)

$$\begin{aligned} v(s) &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \\ &\quad + \dots + \gamma^{n-1} R_{t+n} | S_t = s] \\ \text{i.e., } v(s) &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \end{aligned} \quad (9.21)$$

Karena fungsi nilai harapan adalah fungsi linier, yaitu $\mathbb{E}(aX + bY) = a\mathbb{E}(X) + b\mathbb{E}(Y)$, maka nilai harapan dari pengembalian G_{t+1} adalah nilai dari keadaan S_{t+1}

Oleh karena itu,

$$v(s) = \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] \quad (9.22)$$

Persamaan ini mencerminkan dua komponen:

- **Imbalan langsung:** R_{t+1}
- **Nilai masa depan:** $\gamma v(S_{t+1})$, yang didiskontokan dengan faktor γ .

Persamaan Bellman menangkap gagasan bahwa nilai suatu keadaan tidak hanya bergantung pada imbalan langsung tetapi juga pada nilai yang diharapkan dari keadaan berikutnya, yang diberi bobot oleh probabilitas transisi ke keadaan tersebut.

9.11 INTERPRETASI PERSAMAAN BELLMAN

Untuk setiap keadaan dalam ruang keadaan, persamaan Bellman (Persamaan 9.22) memberi kita nilai keadaan tersebut, $v(s)$. Nilai keadaan s adalah imbalan yang kita terima saat meninggalkan keadaan tersebut, ditambah rata-rata yang didiskon atas keadaan penerus berikutnya yang mungkin, di mana nilai setiap keadaan penerus yang mungkin dikalikan dengan probabilitas mendarat di keadaan tersebut.

$$v(s) = R_s + \sum_{s' \in S} p_{ss'} v(s') \quad (9.23)$$

Di sini kita melihat bahwa persamaan Bellman (Persamaan 9.23) adalah persamaan linear.

$$\begin{aligned} v &= R + \gamma p v \\ (1 - \gamma p) v &= R \end{aligned}$$

Karena itu,

$$v = (1 - \gamma p)^{-1} R \quad (9.24)$$

Meskipun pendekatan ini elegan secara matematis, penyelesaian persamaan Bellman secara langsung dapat memakan biaya komputasi yang besar – terutama di lingkungan dengan sejumlah besar keadaan. Kompleksitas waktu biasanya $O(n^3)$, sehingga tidak praktis untuk masalah berdimensi tinggi.

Meskipun *pendekatan berbasis model* seperti *kerangka kerja MDP* bergantung pada pengetahuan lengkap tentang lingkungan – termasuk probabilitas transisi dan fungsi hadiah – banyak skenario dunia nyata tidak menawarkan informasi komprehensif tersebut. Dalam kasus ini, *pembelajaran tanpa model* menjadi alternatif yang praktis. Bab ini memperkenalkan dua metode dasar tanpa model yang digunakan untuk *tugas prediksi* dalam RL: *Monte Carlo (MC)* dan pembelajaran *Perbedaan Temporal (TD)*. Algoritma ini memungkinkan agen untuk belajar langsung dari pengalaman tanpa memerlukan model lingkungan. Metode RL tanpa model biasanya diterapkan pada dua jenis tugas:

- **Tugas Prediksi:** Tugas ini berfokus pada estimasi imbalan kumulatif yang diharapkan dari suatu keadaan dengan asumsi kebijakan $\pi(a | s)$ telah diketahui. Pada intinya, tugas prediksi bertujuan menghitung fungsi nilai V_π , baik dengan maupun tanpa akses ke model lingkungan. Contoh yang umum adalah *evaluasi kebijakan*, yaitu menilai kualitas suatu kebijakan.
- **Tugas Kontrol:** Tugas-tugas ini bertujuan untuk menemukan kebijakan optimal π^* yang memaksimalkan imbalan kumulatif yang diharapkan dari setiap keadaan. Dengan kebijakan π saat ini, tugas kontrol melibatkan *peningkatan kebijakan* – menyempurnakan kebijakan untuk mencapai hasil yang lebih baik.

Proses *iterasi kebijakan* menggabungkan prediksi dan kontrol: proses ini bergantian antara mengevaluasi kebijakan dan memperbaikinya untuk konvergen menuju kebijakan optimal. Dalam bab ini, kita akan fokus pada *tugas prediksi*, sedangkan *tugas kontrol* akan dibahas di Bab 10. Sebelum membahas metode tanpa model, penting untuk memahami dua konsep tambahan yang memengaruhi cara agen belajar:

- **Pembelajaran On-Policy vs. Off-Policy:** Perbedaan antara pembelajaran *on-policy* dan *off-policy* terletak pada hubungan antara kebijakan yang diikuti dan kebijakan yang dievaluasi:
 - **Pembelajaran On-Policy:** Agen belajar dengan mengevaluasi atau meningkatkan *kebijakan yang sama* yang digunakannya untuk membuat keputusan. Ini mirip dengan "belajar sambil bekerja," di mana agen memperbaiki perilakunya berdasarkan pengalamannya sendiri.
 - **Pembelajaran Off-Policy:** Agen belajar tentang *kebijakan target* sambil mengikuti *kebijakan perilaku yang berbeda*. Biasanya, kebijakan target bersifat deterministik, sedangkan kebijakan perilaku mungkin bersifat stokastik. Metode off-policy memungkinkan agen untuk belajar dari pengalaman masa lalu atau bahkan dari agen lain. Contoh yang terkenal adalah *Q-learning*, yang mempelajari nilai kebijakan optimal terlepas dari kebijakan perilaku yang digunakan untuk menghasilkan data.
- **Tugas Episodik vs. Kontinu:** Tugas RL dapat dikategorikan berdasarkan durasinya:
 - **Tugas Episodik:** Tugas ini memiliki titik akhir yang jelas. Misalnya, permainan catur bersifat episodik – berakhir dengan kemenangan, kekalahan, atau hasil imbang setelah serangkaian gerakan yang terbatas.
 - **Tugas Kontinu:** Tugas ini tidak memiliki titik akhir alami. Contohnya adalah perdagangan saham, di mana keputusan dibuat terus menerus dari waktu ke waktu tanpa titik akhir yang ditentukan.

Memahami perbedaan ini membantu dalam memilih algoritma pembelajaran yang tepat dan menyusun masalah RL secara efektif.

9.12 PENDEKATAN MONTE CARLO

Metode MC memperkirakan fungsi nilai dengan merata-ratakan pengembalian yang diperoleh dari episode interaksi lengkap. Metode ini cocok untuk *tugas episodik*, di mana setiap episode akhirnya berakhir. Misalkan sebuah episode direpresentasikan sebagai:

$$E = (S_1, A_1, R_1, S_2, A_2, R_2, \dots, S_T)$$

Nilai suatu keadaan s diperkirakan sebagai pengembalian rata-rata setelah kemunculan pertama s dalam setiap episode:

$$v(s) = \frac{1}{N} \sum_{i=1}^N G_i(s) \quad (9.25)$$

di mana:

- $G_i(s)$ adalah pengembalian dari keadaan s pada episode i ,
- N adalah jumlah episode di mana s muncul.

Algoritma untuk evaluasi kebijakan MC yang mengembalikan $v \approx v_\pi$ diadaptasi dari dan dibahas dalam Tabel 9.2. Dari Tabel 9.2 dapat disimpulkan bahwa $v(s)$ merupakan rata-rata G yang dihitung dengan pendekatan rata-rata inkremental karena seluruh nilai belum tersedia sekaligus. Nilai baru diperoleh setelah episode berakhir. Oleh sebab itu, rata-rata pengembalian diperbarui secara bertahap dengan prosedur berikut:

- Untuk setiap episode
- Untuk setiap keadaan S_t dengan pengembalian G_t

$$N(S_t) \leftarrow N(S_t) + 1$$

TABEL 9.2 Evaluasi Kebijakan MC Kunjungan Pertama

```

Initialize
   $\pi \leftarrow$  policy to be evaluated
   $v \leftarrow$  an arbitrary state-value function
   $Returns(s) \leftarrow$  an empty list, for all  $s \in S$ 

Repeat forever:
  Generate an episode using  $\pi$ 
  For each state  $s$  appearing in the episode
     $G \leftarrow$  return following the first occurrence of  $s$ 
  Append  $G$  to  $Returns(s)$ 
   $v(s) \leftarrow average(Returns(s))$ 
  
```

$$v(S_t) \leftarrow v(S_t) + \frac{1}{N(S_t)} (G_t - v(S_t)) \quad (9.26)$$

Dua pendekatan untuk mengevaluasi fungsi nilai suatu kebijakan untuk suatu keadaan adalah dengan menggunakan:

- **MC Setiap Kunjungan:** Memperbarui nilai suatu keadaan setiap kali muncul dalam sebuah episode:
 1. Evaluasi nilai keadaan (s) dari kebijakan tertentu
 2. Langkah waktu *pertama* (t) di mana keadaan (s) dikunjungi dalam sebuah episode
 3. Tambahkan penghitung: $N(s) \leftarrow N(s) + 1$
 4. Tambahkan total pengembalian: $S(s) \leftarrow S(s) + G_t$
 5. Nilai diperkirakan dengan pengembalian rata-rata: $v(s) = \frac{S(s)}{N(s)}$
 6. Saat $N(s) \rightarrow \infty$, $v(s) \rightarrow v_\pi(s)$

Dalam kedua pendekatan evaluasi, nilai hanya dapat dihitung setelah semua episode selesai. Oleh karena itu, rata-rata inkremental digunakan yang ditentukan menggunakan ekspresi

Rata-rata $\mu_1, \mu_2, \mu_3, \dots, \mu_n$ dari suatu urutan $x_1, x_2, x_3, \dots, x_n$ dapat dihitung secara inkremental sebagai

$$\mu_k = \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1}) \quad (9.27)$$

Untuk memperbarui $v(s)$ secara bertahap setelah episode $E =$

$\{(S_1, a_1, R_1), (S_2, a_2, R_2), (S_3, a_3, R_3), \dots, (S_t)\}$, digunakan ekspresi pada Persamaan 9.26.

Dalam masalah non-stasioner di mana seseorang dapat melupakan episode lama, pendekatan rata-rata bergerak dapat digunakan (lihat Persamaan 9.28). Dengan demikian

$$v(S_t) \leftarrow v(S_t) + \alpha(G_t - v(S_t)) \quad (9.28)$$

Pada bagian selanjutnya, kita akan membahas pendekatan TD untuk evaluasi kebijakan.

9.13 BELAJAR DARI PERUBAHAN NILAI

Metode TD menggabungkan gagasan Monte Carlo dan pemrograman dinamis. Seperti MC, TD bersifat bebas model dan belajar dari pengalaman. Berbeda dengan MC, TD tidak menunggu episode berakhir; estimasi nilai diperbarui pada setiap langkah waktu melalui bootstrap. Tujuan dari MC dan TD adalah untuk mempelajari v_π dari pengalaman di bawah π . Sebelumnya dalam MC, kita mempelajari bahwa jika suatu episode $E = \{(S_1, a_1, R_1), (S_2, a_2, R_2), (S_3, a_3, R_3), \dots, (S_T)\}$, suatu himpunan urutan keadaan hingga keadaan terminal maka

$$v(s) = \mathbb{E}[G_t \mid S_t = s]$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

Namun, dalam TD, persamaan Bellman digunakan untuk memperbarui estimasi.

$$v(s) = \mathbb{E}[G_t \mid S_t = s] \quad (9.29)$$

Karena $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{n-1} R_{t+n}$, maka Persamaan 9.29 sekarang menjadi

$$v(s) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{n-1} R_{t+n} \mid S_t = s]$$

$$v(s) = \mathbb{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \dots + \gamma^{n-2} R_{t+n}) \mid S_t = s]$$

Karena itu, $v(s) = \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$

$$v(s) = \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$$

Karena $G_{t+1} = R_{t+2} + \gamma R_{t+3} + \dots + \gamma^{n-2} R_{t+n}$

Oleh karena itu, dalam TD pembaruan estimasi imbalan masa depan yang diharapkan (V) diberikan sebagai

$$v(S_t) \leftarrow v(S_t) + \alpha(R_{t+1} + \gamma v(S_{t+1}) - v(S_t)) \quad (9.30)$$

Di sini, pada Persamaan 9.30, bagian $R_{t+1} + \gamma v(S_{t+1}) - v(S_t)$ adalah kesalahan TD. Dalam TD, pada setiap langkah t , target TD dibentuk segera menggunakan hadiah yang diamati R_{t+1} dan estimasi saat ini $v(S_{t+1})$. Metode TD paling sederhana yang dikenal sebagai TD (0) diberikan sebagai

$$v(S_t) \leftarrow v(S_t) + \alpha(R_{t+1} + \gamma v(S_{t+1}) - v(S_t)) \quad (9.31)$$

Tabel 9.3 memberikan pseudokode prediksi TD (0). Terdapat pertukaran bias dan varians antara pendekatan pembelajaran MC dan:

- **Monte Carlo:** Varians tinggi, bias nol. Ia menggunakan pengembalian aktual, yang bergantung pada banyak faktor acak, tetapi memberikan estimasi yang tidak bias.
- **Perbedaan Temporal:** Varians lebih rendah, beberapa bias. Ia bergantung pada estimasi imbalan di masa depan, yang dapat memperkenalkan bias tetapi sering kali mengarah pada pembelajaran yang lebih cepat dan lebih stabil.

TABEL 9.3 TD (0) untuk Memperkirakan v_π

```

Input: The policy  $\pi$  to be evaluated
Initialize  $v(S)$  arbitrarily (e.g.,  $v(S) = 0, \forall S \in S^+$ )
Repeat (for each episode):
  Initialize  $S_t$ 
  Repeat (for each step of episode):
     $A \leftarrow$  action given by  $\pi$  for  $S_t$ 
    Take action  $A$ , observe  $R, S_{t+1}$ 
     $v(S_t) \leftarrow v(S_t) + \alpha(R_{t+1} + \gamma v(S_{t+1}) - v(S_t))$ 
     $S_t \leftarrow S_{t+1}$ 
  Until  $S_t$  is terminal
  
```

Untuk memperkuat pemahaman kita tentang prediksi Monte Carlo dalam pengaturan RL tanpa model, mari kita bahas contoh sederhana menggunakan **dunia grid 1x4**. Skenario ini menggambarkan bagaimana agen dapat mempelajari fungsi nilai melalui interaksi episodik dan pembaruan inkremental.

9.14 DUNIA GRID DAN PREDIKSI MC

Sebagai ilustrasi algoritma prediksi MC, pertimbangkan dunia grid 1×4 yang terdiri atas empat sel. Agen berada pada Sel 2, sedangkan Sel 4 merupakan keadaan tujuan (lihat Gambar 9.3). Pada Gambar 9.3, keadaan $S \in \{\text{Sel 1, Sel 2, Sel 3, Sel 4}\}$ dan tindakan $A \in \{\text{bergerak ke kiri, bergerak ke kanan}\}$. Agen memperoleh imbalan 10 poin ketika mencapai

keadaan tujuan, sedangkan setiap tindakan dikenai biaya 1 poin. Tiga episode berikut digunakan untuk evaluasi:



Gambar 9.3 Dunia grid 1×4 dengan empat sel di mana agen berada di sel 2 dan sel 4 adalah keadaan tujuan.

Episode 1: Cell2, Cell 3, Cell 4

Episode 2: Cell2, Cell 3, Cell2, Cell 3, Cell 4

Episode 3: Cell2, Cell 1, Cell2, Cell 3, Cell 4

Sekarang, mari kita hitung pengembalian untuk setiap episode dengan $\gamma = 0.9$

Untuk sel 1

Karena episode 1 dan 2 tidak melibatkan sel 1, maka $G_t = 0$. Untuk episode 3, $G_t = (-1) \times 0.9^0 + (-1) \times 0.9^1 + 10 \times 0.9^2 = 6.2$. Angka (-1) pertama adalah untuk agen berpindah dari Sel 1 ke Sel 2. Angka (-1) kedua adalah untuk agen berpindah dari Sel 2 ke Sel 3. Angka 10 adalah untuk agen berpindah dari Sel 3 ke Sel 4 yang merupakan keadaan tujuan.

Sekarang, untuk sel 2

Untuk episode 1, $G_t = (-1) \times 0.9^0 + 10 \times 0.9^1 = 8$

Untuk episode 2, $G_t = (-1) \times 0.9^0 + (-1) \times 0.9^1 + (-1) \times 0.9^2 + 10 \times 0.9^3 = 4.58$

Untuk episode 3, $G_t = (-1) \times 0.9^0 + (-1) \times 0.9^1 + (-1) \times 0.9^2 + 10 \times 0.9^3 = 4.58$

Fungsi nilai empiris untuk Sel 2 adalah $\frac{8+4.58+4.58}{3} = 5.72$.

Sekarang, untuk sel 3

Untuk episode 1, $G_t = 10 \times 0.9^0 = 10$

Untuk episode 2, $G_t = (-1) \times 0.9^0 + (-1) \times 0.9^1 + 10 \times 0.9^3 = 6.2$

Untuk episode 3, $G_t = 10 \times 0.9^0 = 10$

Fungsi nilai empiris untuk Sel 3 adalah $\frac{10+6.2+10}{3} = 8.73$.

Fungsi nilai sel yang lebih akurat dapat diperoleh dengan mempertimbangkan episode tambahan.

9.15 RINGKASAN

Dalam bab ini, kita telah meletakkan dasar-dasar konsep *RL*, dengan fokus pada bagaimana agen mempelajari perilaku optimal melalui interaksi dengan lingkungannya. Kita mulai dengan memperkenalkan terminologi dan konvensi inti – seperti agen, keadaan, tindakan, imbalan, kebijakan, dan fungsi nilai – yang mendefinisikan struktur sistem *RL*. Kemudian kita mengeksplorasi kerangka kerja *MDP*, yang menyediakan model formal untuk lingkungan *RL*. Ketergantungan *MDP* pada *sifat Markov* memungkinkan agen untuk membuat

keputusan hanya berdasarkan keadaan saat ini, menyederhanakan proses pembelajaran. Dengan menggunakan kerangka kerja ini, kita menurunkan *persamaan Bellman*, yang secara rekursif menghubungkan nilai suatu keadaan dengan imbalan langsungnya dan nilai yang diharapkan dari keadaan penerus. Berdasarkan fondasi ini, kita memperkenalkan *pendekatan pembelajaran tanpa model* untuk tugas prediksi, di mana tujuannya adalah untuk memperkirakan fungsi nilai berdasarkan kebijakan. Dua metode utama dibahas:

- **Pembelajaran MC:** Memperkirakan fungsi nilai dengan merata-ratakan pengembalian dari episode lengkap. Metode ini sangat cocok untuk tugas episodik dan memberikan estimasi yang tidak bias, meskipun mungkin memiliki varians yang tinggi.
- **Pembelajaran TD:** Memperbarui estimasi nilai secara bertahap setelah setiap langkah waktu menggunakan bootstrapping. Metode TD lebih efisien dan dapat diterapkan pada tugas episodik dan kontinu, meskipun dapat menimbulkan bias.

Kami juga memeriksa *trade-off bias-variants* antara metode MC dan TD dan mengilustrasikan prediksi MC melalui *contoh dunia grid* sederhana, yang menunjukkan bagaimana agen mempelajari fungsi nilai dari pengalaman. Pada bab berikutnya, kami akan memperluas konsep-konsep ini ke *tugas kontrol*, di mana tujuannya bergeser dari mengevaluasi kebijakan yang diberikan ke mempelajari kebijakan optimal. Topik yang akan dibahas meliputi *Q-learning*, *metode Actor-Critic*, dan *Deep RL*, yang menggabungkan RL dengan jaringan saraf dalam untuk mengatasi lingkungan berdimensi tinggi dan kompleks.

9.16 PERTANYAAN

1. Analisis perbedaan utama antara Machine Learning (ML) dan Reinforcement Learning (RL). Berikan contoh aplikasi di mana RL lebih disukai daripada pendekatan ML tradisional, dan berikan alasan untuk pilihan Anda.
2. Definisikan dan jelaskan terminologi kunci berikut dalam RL: Agen, Lingkungan, Keadaan, Tindakan, Hadiah, Triple SAR, Kebijakan, Episode, Tujuan, dan Fungsi Nilai. Ilustrasikan bagaimana konsep-konsep ini berinteraksi dalam praktik menggunakan contoh yang relevan.
3. Diskusikan Proses Keputusan Markov (MDP) dalam konteks RL. Jelaskan signifikansi properti Markov, probabilitas transisi keadaan, dan persamaan Bellman dalam menyelesaikan masalah RL.
4. Manakah dari pernyataan berikut yang benar mengenai metode pembelajaran Monte Carlo (MC)?
 - a. Mereka memperbarui fungsi nilai setelah setiap langkah menggunakan bootstrapping
 - b. Mereka dapat diterapkan pada tugas kontinu tanpa penghentian
 - c. Mereka membutuhkan tugas episodik dengan episode yang berakhir
 - d. Mereka memperbarui fungsi nilai menggunakan kesalahan TD
5. Dalam pembelajaran Temporal-Difference (TD), kesalahan TD mewakili
 - a. Perbedaan antara hadiah langsung dan pengembalian yang didiskon
 - b. Perbedaan antara nilai prediksi dan pengembalian aktual

- c. Hadiah yang diperoleh di akhir episode
 - d. Distribusi probabilitas atas keadaan selanjutnya
6. [Latihan Pemrograman – Dasar] Implementasikan fungsi Python yang menghitung pengembalian terdiskonto G_t dengan diberikan daftar hadiah dan faktor diskonto γ . Demonstrasikan operasi fungsi menggunakan hadiah $[-1,1,2,10]$ dengan $\gamma = 0.9$.
 7. [Latihan Pemrograman – Menengah] Simulasikan Evaluasi Kebijakan Monte Carlo Kunjungan Pertama untuk lingkungan dunia grid 4-keadaan sederhana (seperti yang dijelaskan dalam bab ini). Gunakan minimal 10 episode untuk memperkirakan fungsi nilai setiap keadaan.

BAB 10

PEMBELAJARAN PENGUATAN MENDALAM

10.1 ALGORITMA Q-LEARNING

Pembelajaran Penguatan Mendalam (Deep Reinforcement Learning/DRL) menggabungkan kekuatan Pembelajaran Penguatan (Reinforcement Learning/RL) dan Pembelajaran Mendalam (Deep Learning/DL) agar agen dapat mempelajari strategi pengambilan keputusan yang kompleks melalui interaksi dengan lingkungan. Dengan memanfaatkan jaringan saraf dalam (deep neural networks/DNN) untuk merepresentasikan kebijakan, fungsi nilai, atau model lingkungan, DRL memungkinkan agen memproses masukan berdimensi tinggi dan belajar melalui coba-coba untuk memaksimalkan imbalan kumulatif. Seperti yang diperkenalkan pada Bab 9, RL adalah paradigma pembelajaran mesin di mana agen berinteraksi dengan lingkungan dengan mengambil tindakan, menerima umpan balik berupa imbalan atau penalti, dan mempelajari kebijakan – sebuah strategi – untuk memaksimalkan imbalan kumulatif dari waktu ke waktu. Komponen utama sistem RL meliputi agen, lingkungan, keadaan, tindakan, dan fungsi imbalan.

Metode RL tradisional, meskipun efektif dalam pengaturan yang lebih sederhana atau berdimensi rendah, mengalami kesulitan dalam hal skalabilitas dan kinerja di lingkungan yang kompleks. DRL mengatasi keterbatasan ini dengan menggabungkan DL untuk secara otomatis mengekstrak fitur dari data mentah, seperti gambar atau pembacaan sensor, memungkinkan agen untuk beroperasi dalam ruang keadaan dan aksi berdimensi tinggi. Misalnya, dalam game Atari, agen DRL dapat belajar bermain langsung dari input piksel tanpa rekayasa fitur manual. DRL telah menunjukkan keberhasilan di berbagai bidang, termasuk:

- **Permainan:** Menguasai permainan kompleks seperti Go (misalnya, AlphaGo) dan video game Atari.
- **Robotika:** Melatih robot untuk berjalan, menggenggam objek, dan menavigasi lingkungan dinamis.
- **Kendaraan Otonom:** Mengembangkan kemampuan mengemudi sendiri melalui pembelajaran berkelanjutan.
- **Perawatan Kesehatan:** Membantu dalam diagnosis, perencanaan perawatan, dan penemuan obat.
- **Keuangan:** Mengoptimalkan strategi perdagangan dan mengelola risiko.

Ia menawarkan beberapa keunggulan dibandingkan RL tradisional:

- **Pembelajaran Ujung-ke-Ujung:** DRL dapat belajar langsung dari input sensor mentah, menghilangkan kebutuhan akan rekayasa fitur manual.
- **Skalabilitas:** DNN memungkinkan DRL untuk menangani ruang keadaan dan aksi yang besar dan kompleks.
- **Adaptabilitas:** Agen DRL dapat menyesuaikan diri dengan lingkungan yang berubah dan mempelajari strategi baru dari waktu ke waktu.

- **Otomatisasi:** DRL memfasilitasi otomatisasi tugas-tugas kompleks di berbagai industri.

Terlepas dari keunggulannya, DRL menghadirkan beberapa tantangan:

- **Persyaratan Data:** DRL sering memerlukan data pelatihan dalam jumlah besar. Keterbatasan data dapat menyebabkan underfitting, overfitting, atau varians kinerja yang tinggi.
- **Dilema Eksplorasi–Eksplotasi:** Menyeimbangkan kebutuhan untuk mengeksplorasi strategi baru versus mengeksplotasi strategi yang sudah dikenal tetap menjadi tantangan utama. Pertimbangan ini akan dibahas secara rinci di bagian selanjutnya dalam bab ini.
- **Kekhawatiran Etis:** Seiring sistem DRL menjadi lebih mumpuni, muncul kekhawatiran seputar penggantian pekerjaan, bias algoritmik, dan potensi penyalahgunaan.

Agen DRL belajar melalui interaksi dengan lingkungan dengan memilih tindakan berdasarkan kebijakan. Lingkungan merespons dengan bertransisi ke keadaan baru dan memberikan imbalan. Umpan balik ini digunakan untuk memperbarui kebijakan sehingga kualitas pengambilan keputusan meningkat secara iteratif. Proses coba-coba tersebut memungkinkan agen menyempurnakan strateginya dari waktu ke waktu. Sebelum menyelami teknik DRL yang lebih canggih, penting untuk meninjau kembali salah satu algoritma dasar dalam RL tanpa model: *Q-learning*. Algoritma ini berfungsi sebagai batu loncatan untuk memahami bagaimana metode berbasis nilai dapat diperluas menggunakan DNN. *Q-learning* menyediakan dasar konseptual dan matematis untuk Deep Q-Networks (DQN), yang akan kita jelajahi nanti di bab ini.

Q-learning adalah algoritma RL tanpa model yang memungkinkan agen untuk mempelajari tindakan optimal melalui coba-coba, tanpa memerlukan model lingkungan. Hal ini dilakukan dengan mempelajari sebuah fungsi – yang disebut *fungsi Q* – yang memperkirakan imbalan kumulatif yang diharapkan untuk mengambil tindakan tertentu dalam keadaan tertentu dan kemudian mengikuti kebijakan optimal setelahnya. Fungsi *Q* diperbarui secara iteratif menggunakan *persamaan Bellman*, yang menyeimbangkan imbalan langsung dengan perkiraan imbalan di masa depan. Aturan pembaruan *Q-learning* satu langkah diberikan oleh:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (10.1)$$

di mana:

- $Q(s_t, a_t)$: Estimasi nilai *Q* saat ini untuk pasangan keadaan-tindakan (s_t, a_t)
- α : Laju pembelajaran.
- r_{t+1} : Hadiah yang diterima setelah mengambil tindakan di
- γ : Faktor diskonto.
- $\max_a Q(s_{t+1}, a)$: Estimasi nilai *Q* maksimum untuk keadaan berikutnya

TABEL 10.1 Algoritma Q-Learning Off-Policy

```
Initialize  $Q(s_t, a_t)$  arbitrarily
Repeat (for each episode):
  Initialize  $S_t$ 
  Repeat (for each step of episode):
    Choose  $a_t$  from  $S_t$  using the policy derived from  $Q$ 
    Take action  $a_t$ , observe  $r$ ,  $S_{t+1}$ 
     $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$ 
     $S_t \leftarrow S_{t+1}$ 
  Until  $S_t$  is terminal
```

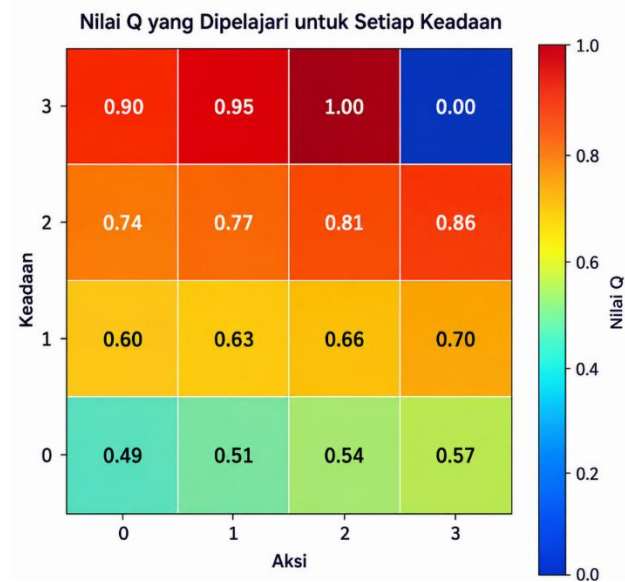
Aturan pembaruan ini memungkinkan agen untuk memperbaiki nilai Q -nya dari waktu ke waktu, secara bertahap mendekati fungsi nilai aksi yang optimal. Tabel 10.1 menyajikan pseudocode untuk algoritma Q-learning off-policy. Mari kita coba memahami bagaimana algoritma Q-learning ini bekerja:

1. **Inisialisasi:** Tabel Q diinisialisasi dengan nol atau nilai acak kecil.
2. **Pemilihan Aksi:** Pada setiap langkah waktu, agen memilih aksi menggunakan strategi eksplorasi seperti ϵ -greedy, yang menyeimbangkan eksplorasi dan eksploitasi.
3. **Interaksi:** Agen melakukan aksi yang dipilih, menerima hadiah, dan beralih ke keadaan baru.
4. **Pembaruan Nilai Q :** Nilai Q untuk pasangan keadaan-aksi sebelumnya diperbarui menggunakan persamaan Bellman.
5. **Iterasi:** Langkah 2–4 diulang di beberapa episode hingga nilai Q konvergen.

10.2 CONTOH LINGKUNGAN GRID

Untuk mendemonstrasikan Q-learning dalam aksi, pertimbangkan lingkungan grid-world sederhana di mana agen harus belajar untuk mencapai keadaan tujuan. Tabel 10.2 menyediakan implementasi Python dari algoritma Q-learning dalam pengaturan ini. Skrip Python dalam Tabel 10.2 mendemonstrasikan implementasi dasar algoritma Q-learning dalam lingkungan grid-world yang disederhanakan dengan 16 keadaan dan 4 kemungkinan tindakan per keadaan. Agen memulai dalam keadaan acak dan mencoba mencapai keadaan tujuan (keadaan 15). Pada setiap langkah, ia memilih tindakan menggunakan strategi ϵ -greedy: dengan probabilitas 0,2, ia melakukan eksplorasi dengan memilih tindakan acak; jika tidak, ia melakukan eksploitasi dengan memilih tindakan dengan nilai Q tertinggi.

Lingkungan merespons dengan hadiah 1 jika agen mencapai keadaan tujuan, dan 0 jika tidak. Tabel Q diperbarui menggunakan aturan pembaruan Q-learning, yang menggabungkan laju pembelajaran dan faktor diskon. Selama lebih dari 1.000 episode pelatihan, agen tersebut menyempurnakan nilai Q -nya, secara bertahap mempelajari jalur optimal menuju tujuan. Nilai Q akhir divisualisasikan sebagai peta panas (Gambar 10.1), di mana nilai yang lebih tinggi menunjukkan keadaan yang lebih menguntungkan, membimbing agen menuju tujuan.



Gambar 10.1 Nilai Q yang dipelajari untuk setiap keadaan dalam lingkungan seperti grid.

Mari kita rangkum secara singkat langkah-langkah yang diimplementasikan dalam skrip Python (lihat Tabel 10.2). Pertama, kita menginisialisasi tabel Q dengan semua 0. Dalam fase pembelajaran, agen bertindak (mengambil tindakan) dan menerima imbalan. Hal ini menghasilkan pembaruan nilai dalam tabel Q . Untuk pasangan keadaan-tindakan, yaitu $Q(s_t, a_t)$ nilai Q diperbarui menggunakan ekspresi (lihat persamaan 10.2):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \quad (10.2)$$

Pembaruan konstan pada nilai Q pada akhirnya akan menghasilkan konvergensi tabel, yaitu, memperoleh nilai yang akurat untuk suatu tindakan dalam keadaan tertentu. Pendekatan ϵ -greedy digunakan di sini untuk eksplorasi dan eksploitasi. Kita akan membahas pendekatan ϵ -greedy sebagai bagian dari strategi eksplorasi dan eksploitasi nanti di bab ini. Singkatnya, eksplorasi adalah tentang mencari opsi baru melalui eksperimen dan pengembangan untuk menghasilkan nilai baru, sedangkan eksploitasi adalah tentang menyempurnakan opsi yang ada dengan mengulangi opsi tersebut dan meningkatkan nilainya. Setelah beberapa iterasi, tindakan terbaik di antara semua tindakan, pada setiap keadaan, akan ditemukan. Hal ini karena agen mendapat kesempatan untuk mengeksplorasi tindakan acak baru serta mengeksploitasi tindakan yang ada dan menyempurnakannya.

TABEL 10.2 Skrip Python untuk Q-Learning dalam Lingkungan Grid

```
import numpy as np

n_states = 16
n_actions = 4
goal_state = 15
```

```
Q_table = np.zeros((n_states, n_actions))

learning_rate = 0.8
discount_factor = 0.95
exploration_prob = 0.2
epochs = 1000

for epoch in range(epochs):
    current_state = np.random.randint(0, n_states)
    while current_state != goal_state:
        if np.random.rand() < exploration_prob:
            action = np.random.randint(0, n_actions)
        else:
            action = np.argmax(Q_table[current_state])

        next_state = (current_state + 1) % n_states
        reward = 1 if next_state == goal_state else 0

        Q_table[current_state, action] += learning_rate * \
            (reward + discount_factor *
             np.max(Q_table[next_state]) - Q_table[current_state, action])

        current_state = next_state

q_values_grid = np.max(Q_table, axis=1).reshape((4, 4))

# Plot the grid of Q-values
plt.figure(figsize=(6, 6))
plt.imshow(q_values_grid, cmap='coolwarm', interpolation='nearest')
plt.colorbar(label='Q-value')
plt.title('Learned Q-values for each state')
plt.xticks(np.arange(4), ['0', '1', '2', '3'])
plt.yticks(np.arange(4), ['0', '1', '2', '3'])
plt.gca().invert_yaxis() # To match grid layout
plt.grid(True)

# Annotating the Q-values on the grid
for i in range(4):
    for j in range(4):
        plt.text(j, i, f'{q_values_grid[i, j]:.2f}', ha='center',
                 va='center', color='black')
plt.show()

# Print learned Q-table
print("Learned Q-table:")
print(Q_table)
```

```

[[0.48767498 0.48377293 0.48377358 0.48377358]
 [0.51334208 0.4928084 0.50881484 0.51330923]
 [0.54035981 0.54036009 0.54036009 0.54036003]
 [0.56880009 0.56880009 0.56880009 0.56879952]
 [0.59873694 0.59873694 0.59873694 0.59873694]
 [0.63024941 0.63024941 0.63024941 0.63024941]
 [0.66342043 0.66342043 0.66342043 0.66342043]
 [0.6983373 0.6983373 0.6983373 0.6983373 ]
 [0.73509189 0.73509189 0.73509189 0.73509189]
 [0.77378094 0.77378094 0.77378094 0.77378094]
 [0.81450625 0.81450625 0.81450625 0.81450625]
 [0.857375 0.857375 0.857375 0.857375 ]
 [0.9025 0.9025 0.9025 0.9025 ]
 [0.95 0.95 0.95 0.95 ]
 [1. 1. 1. 1. ]

```

Gambar 10.2 Nilai Q dalam tabel Q yang telah konvergen untuk lingkungan seperti grid.

Tabel Q yang telah konvergen setelah menjalankan skrip ditunjukkan pada Gambar 10.2. Q -Learning memiliki beberapa keunggulan:

- Mempelajari kebijakan optimal melalui pengalaman tanpa memerlukan model lingkungan.
- Mampu belajar dari imbalan yang tertunda.
- Mudah diimplementasikan dan efektif di lingkungan diskrit.

Namun, bukan tanpa keterbatasan:

- Konvergensi lambat di ruang keadaan yang besar atau kontinu.
- Menderita kutukan dimensi; tabel Q tumbuh secara eksponensial dengan jumlah keadaan dan tindakan.
- Tidak cocok untuk ruang tindakan kontinu tanpa diskretisasi.
- Ketidakefisienan sampel; membutuhkan banyak interaksi dengan lingkungan untuk belajar.
- Keterbatasan ini memotivasi pengembangan pendekatan yang lebih terukur, seperti DQN, yang akan kita bahas nanti di bab ini.

Meskipun Q -learning menyediakan kerangka kerja yang ampuh untuk mempelajari kebijakan berbasis nilai, metode ini menjadi kurang efektif di lingkungan dengan ruang aksi yang besar atau kontinu. Dalam kasus seperti itu, sering kali lebih efisien untuk mempelajari kebijakan secara langsung daripada secara tidak langsung melalui fungsi nilai. Hal ini membawa kita pada *metode Policy Gradient (PG)*, yang menjadi dasar bagi banyak algoritma RL modern, termasuk yang digunakan dalam robotika dan tugas kontrol kontinu.

10.3 GRADIEN KEBIJAKAN

Tujuan agen RL adalah mempelajari kebijakan optimal yang memaksimalkan imbalan kumulatif yang diharapkan. Metode Policy Gradient (PG) mengoptimalkan kebijakan secara langsung, bukan menurunkannya dari fungsi nilai. Kebijakan dimodelkan sebagai fungsi berparameter $\pi_\theta(a | s)$, dengan θ menyatakan parameter model, biasanya bobot jaringan saraf. Tujuannya adalah untuk menemukan parameter optimal yang memaksimalkan pengembalian yang diharapkan. Ini diformalkan melalui fungsi imbalan (tujuan):

$$\begin{aligned} J(\theta) &= \sum_{s \in S} d^\pi(s) V^\pi(s) \\ J(\theta) &= \sum_{s \in S} d^\pi(s) \sum_{a \in A} \pi_\theta(a | s) Q^\pi(s, a) \end{aligned} \quad (10.3)$$

di mana:

- $d_\pi(s)$ adalah distribusi stasioner keadaan di bawah kebijakan π_θ ,
- $V_\pi(s)$ adalah fungsi nilai untuk keadaan s ,
- $Q_\pi(s, a)$ adalah fungsi nilai tindakan di bawah kebijakan π .

Untuk mengoptimalkan $J(\theta)$, kita menerapkan gradient ascent, menyesuaikan ke arah gradien $\nabla_\theta J(\theta)$ untuk meningkatkan imbalan yang diharapkan.

Secara intuitif, gradient ascent dimulai dengan tebakan awal untuk nilai bobot kebijakan yang memaksimalkan pengembalian yang diharapkan. Kemudian algoritma mengevaluasi gradien pada titik yang menunjukkan arah peningkatan paling curam dari fungsi pengembalian yang diharapkan. Oleh karena itu, langkah kecil dapat diambil ke arah tersebut. Harapannya adalah untuk mencapai nilai bobot kebijakan baru yang menghasilkan nilai fungsi pengembalian yang diharapkan sedikit lebih besar. Algoritma kemudian mengulangi proses evaluasi gradien dan mengambil langkah-langkah hingga akhirnya dianggap telah mencapai pengembalian yang diharapkan maksimum. Untuk menghitung $\nabla_\theta J(\theta)$, penting untuk mengetahui pemilihan tindakan (ditentukan langsung oleh π_θ) dan distribusi stasioner keadaan yang mengikuti perilaku pemilihan target (ditentukan secara tidak langsung oleh π_θ). Sekarang, mari kita definisikan *teorema PG*. Teorema PG menyatakan bahwa:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} [\nabla_\theta \log \pi_\theta(s, a) Q^{\pi_\theta}(s, a)] \quad (10.4)$$

Ekspresi ini mengungkapkan wawasan kunci: untuk meningkatkan kebijakan, kita harus meningkatkan probabilitas tindakan yang menghasilkan pengembalian yang lebih tinggi dan mengurangi probabilitas tindakan yang menghasilkan pengembalian yang lebih rendah. Gradien dihitung menggunakan sampel dari pengalaman agen, sehingga pendekatan ini cocok untuk ruang tindakan berdimensi tinggi dan kontinu.

Metode PG secara iteratif memperbarui parameter jaringan kebijakan untuk memperkuat tindakan yang menguntungkan. Pada setiap langkah:

- Agen mengambil sampel lintasan (urutan keadaan–tindakan–imbalan) dengan berinteraksi dengan lingkungan.
- Agen menghitung pengembalian G_t untuk setiap langkah waktu.
- Parameter kebijakan diperbarui ke arah yang meningkatkan log-probabilitas tindakan yang menghasilkan pengembalian yang lebih tinggi.

Proses ini berlanjut hingga kebijakan konvergen ke kebijakan yang memaksimalkan imbalan yang diharapkan. Tidak seperti metode berbasis nilai, pendekatan PG pada dasarnya bersifat stokastik, yang memungkinkan eksplorasi yang lebih baik dan pembaruan yang lebih lancar di lingkungan yang kompleks.

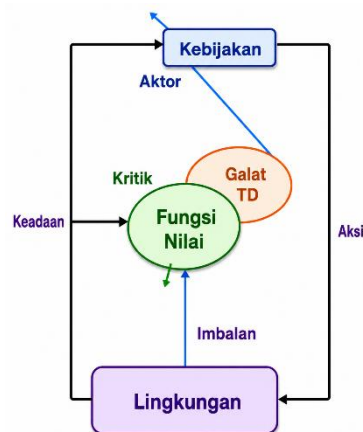
Meskipun metode PG menawarkan pendekatan langsung untuk mengoptimalkan kebijakan, metode ini sering kali mengalami varians yang tinggi dalam estimasi gradien, yang dapat menyebabkan pembelajaran yang tidak stabil. Untuk mengatasi hal ini, metode Actor–Critic menggabungkan kekuatan pendekatan berbasis nilai dan berbasis kebijakan. Algoritma hibrida ini menggunakan dua model terpisah: Actor, yang memilih tindakan, dan Critic, yang mengevaluasinya. Struktur ganda ini memungkinkan pembelajaran yang lebih stabil dan efisien dalam penggunaan sampel, terutama di lingkungan dengan ruang keadaan dan tindakan yang kontinu atau berdimensi tinggi.

10.4 METODE ACTOR–CRITIC

Kerangka kerja Actor–Critic terdiri dari dua komponen:

- **Actor:** Mempelajari kebijakan $\pi_{\theta}(a | s)$ dan memilih tindakan berdasarkan keadaan saat ini.
- **Kritikus (Critic):** Memperkirakan fungsi nilai, baik nilai keadaan $V_{\pi}(s)$ maupun nilai tindakan $Q_{\pi}(s, a)$, untuk mengevaluasi kualitas tindakan yang dipilih Aktor.

Kritikus memberikan umpan balik kepada Aktor untuk mengarahkan pembaruan kebijakan menuju peningkatan pengembalian yang diharapkan. Lingkaran umpan balik ini mengurangi varians estimasi gradien dan mempercepat konvergensi. Gambar 10.3 menggambarkan arsitektur Aktor-Kritikus, yang dapat menggunakan parameter bersama maupun jaringan yang terpisah.



Gambar 10.3 Arsitektur Aktor-Kritikus.

Ingat kembali ekspresi PG:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau} \left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) G_t \right] \quad (10.5)$$

Formulasi ini, meskipun ampuh, menimbulkan variabilitas tinggi karena sifat stokastik dari lintasan. Hal ini selanjutnya akan mengakibatkan pembelajaran yang tidak stabil dan/atau kemiringan distribusi probabilitas ke arah yang tidak optimal. Ada masalah lain dengan PG yang terjadi ketika lintasan memiliki imbalan kumulatif 0. Jika imbalan kumulatifnya 0, akan sulit bagi agen untuk mempelajari tindakan "baik" dan "buruk". Untuk mengurangi varians ini, kita mengganti pengembalian G_t dengan fungsi nilai aksi $Q_{\pi}(s_t, a_t)$, menghasilkan:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau} \left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) Q_w(s_t, a_t) \right] \quad (10.6)$$

di mana Q_w merupakan estimasi Kritikus terhadap fungsi nilai tindakan yang diparameterisasi oleh w . Untuk lebih menyempurnakan gradien, kami memperkenalkan *fungsi keuntungan*:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$$

Fungsi ini mengukur seberapa jauh suatu tindakan tertentu lebih baik dibandingkan dengan tindakan rata-rata pada kondisi tertentu. Dengan menggunakan persamaan Bellman, kita dapat menyatakan keunggulan tersebut sebagai:

$$A(s_t, a_t) = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \quad (10.7)$$

Dengan mensubstitusikan ini ke dalam PG, diperoleh formulasi Advantage Actor–Critic (A2C):

$$\nabla_{\theta} J(\theta) = \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) A(s_t, a_t) \quad (10.8)$$

Pendekatan ini meningkatkan stabilitas pembelajaran dengan memfokuskan pembaruan pada tindakan yang berkinerja lebih baik daripada baseline. Tabel 10.3 memuat langkah-langkah algoritma Q Actor-Critic.

TABEL 10.3 Algoritma Actor–Critic On-Policy Q

- | |
|---|
| <ol style="list-style-type: none"> 1. Initialize s, θ, ω at random and learning rates α_{θ}, α_{ω}; sample $a \sim \pi_{\theta}(a s)$. 2. For $t = 1, 2, \dots, T$: do <ol style="list-style-type: none"> 2.1 Sample reward $r_t \sim R(s, a)$ and next state $s' \sim P(s' s, a)$ 2.2 Then sample the next action $a' \sim \pi_{\theta}(a' s')$ 2.3 Update the policy parameters: $\theta \leftarrow \theta + \alpha_{\theta} Q_w(s, a) \nabla_{\theta} \log \pi_{\theta}(a s)$ |
|---|

```
2.4 Compute the correction (TD error) for action-value at time  $t$ :  
 $\delta_t = r_t + \gamma Q_w(s', a') - Q_w(s, a)$ , and use it to update the parameters of  $Q$  function  
 $\omega \leftarrow \omega + \alpha_\omega \delta_t \nabla_\omega Q_w(s, a)$ .  
2.5 Update  $a \leftarrow a'$  and  $s \leftarrow s'$ .  
End for
```

Algoritma Actor-Critic menggabungkan Aktor, yang memilih tindakan, dan Kritikus, yang mengevaluasi tindakan tersebut. Kombinasi ini membantu agen belajar secara lebih efektif melalui keseimbangan antara pengambilan keputusan dan umpan balik. Aktor mempelajari cara memilih tindakan, sedangkan Kritikus menilai kualitas keputusan itu. Peran ganda tersebut mendukung eksplorasi tindakan baru sekaligus pemanfaatan pengetahuan yang telah diperoleh. Tabel 10.4, yang diadaptasi dari, menyajikan implementasi Python algoritma Actor-Critic menggunakan TensorFlow dan OpenAI Gym. Agen dilatih pada lingkungan CartPole dengan tujuan menyeimbangkan sebuah tiang di atas kereta yang bergerak.

Pertama, lingkungan CartPole dibuat menggunakan fungsi *gym.make* dari pustaka Gym (diperkenalkan pada Bab 1). Selanjutnya, jaringan Aktor dan Kritikus dibangun menggunakan API Keras pada TensorFlow. Jaringan Aktor memetakan keadaan ke distribusi probabilitas tindakan, sedangkan jaringan Kritikus memperkirakan nilai keadaan. Kedua jaringan menggunakan pengoptimal Adam. Perulangan pelatihan dijalankan selama 1.000 episode; agen berinteraksi dengan lingkungan, menghitung advantage, dan memperbarui parameter Aktor serta Kritikus. Pembaca dianjurkan menjalankan skrip pada Tabel 10.4 dan memvisualisasikan hasilnya.

TABEL 10.4 Skrip Python untuk Mendemonstrasikan Implementasi Metode Aktor-Kritikus pada Lingkungan CartPole

```
# Libraries
import gym
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers
import tensorflow_probability as tfp

# Hyperparameters
env_name = 'CartPole-v1'
gamma = 0.99
num_episodes = 500
actor_lr = 0.001
critic_lr = 0.002

# Environment
env = gym.make(env_name)
num_actions = env.action_space.n
num_states = env.observation_space.shape[0]

# Actor network
actor = tf.keras.Sequential([
    layers.InputLayer(input_shape=(num_states,)),
```

```
layers.Dense(24, activation='relu'),
layers.Dense(num_actions, activation='softmax')
])

# Critic network
critic = tf.keras.Sequential([
    layers.InputLayer(input_shape=(num_states,)),
    layers.Dense(24, activation='relu'),
    layers.Dense(1)
])

actor_optimizer = tf.keras.optimizers.Adam(actor_lr)
critic_optimizer = tf.keras.optimizers.Adam(critic_lr)

for episode in range(num_episodes):
    state, _ = env.reset()
    episode_reward = 0
    done = False

    states, actions, rewards, values = [], [], [], []

    while not done:
        state_tensor = tf.convert_to_tensor([state], dtype=tf.float32)
        action_probs = actor(state_tensor)
        value = critic(state_tensor).numpy()

        dist = tfp.distributions.Categorical(probs=action_probs)
        action = dist.sample().numpy()[0]

        next_state, reward, terminated, truncated, info = env.step(action)
        done = terminated or truncated

        states.append(state)
        actions.append(action)
        rewards.append(reward)
        values.append(value)
        state = next_state
        episode_reward += reward

    # Compute returns and advantages
    returns = []
    G = 0
    for r in reversed(rewards):
        G = r + gamma * G
        returns.insert(0, G)
    returns = np.array(returns, dtype=np.float32)
    values = np.array(values, dtype=np.float32)
    advantages = returns - values

    # Convert to tensors
    states_tensor = tf.convert_to_tensor(np.array(states),
                                         dtype=tf.float32)
    actions_tensor = tf.convert_to_tensor(np.array(actions),
                                         dtype=tf.int32)
```

```
advantages_tensor = tf.convert_to_tensor(advantages,
dtype=tf.float32)
returns_tensor = tf.convert_to_tensor(returns, dtype=tf.float32)

with tf.GradientTape() as tape1, tf.GradientTape() as tape2:
    action_probs = actor(states_tensor)
    values_pred = critic(states_tensor)[: , 0]
    dist = tfp.distributions.Categorical(probs=action_probs)
    log_probs = dist.log_prob(actions_tensor)
    actor_loss = -tf.reduce_mean(log_probs * advantages_tensor)
    critic_loss = tf.reduce_mean(tf.square(returns_tensor -
values_pred))

    actor_grads = tape1.gradient(actor_loss, actor.trainable_variables)
    critic_grads = tape2.gradient(critic_loss,
critic.trainable_variables)
    actor_optimizer.apply_gradients(zip(actor_grads,
actor.trainable_variables))
    critic_optimizer.apply_gradients(zip(critic_grads,
critic.trainable_variables))

    if episode % 10 == 0:
        print(f"Episode {episode}, Reward: {episode_reward}")

env.close()
```

Keunggulan dan keterbatasan algoritma Actor-Critic adalah sebagai berikut:

Keuntungan:

- Menggabungkan pembelajaran kebijakan dan nilai untuk meningkatkan efisiensi sampel.
- Konvergensi lebih cepat karena pembaruan simultan.
- Menangani ruang aksi diskrit dan kontinu.
- Varian seperti A2C dan A3C mendukung pembelajaran asinkron dan pembaruan off-policy.

Keterbatasan:

- Varians tinggi dalam estimasi PG dapat mengganggu pelatihan.
- Membutuhkan penyetelan hiperparameter yang cermat.
- Melatih dua jaringan meningkatkan kompleksitas komputasi.
- Kesalahan aproksimasi dalam Critic dapat menyebar dan memengaruhi kualitas kebijakan.

Terdapat juga situasi trade-off bias-varians di sini. Dalam skenario di mana fungsi nilai Critic menghasilkan estimasi PG dengan varians yang lebih rendah, bias diperkenalkan karena fungsi nilai Critic tidak akurat. Pertukaran ini berdampak pada kecepatan konvergensi dan optimalitas kebijakan yang dipelajari.

Meskipun metode Actor–Critic dapat lebih efisien dalam memanfaatkan tindakan baik yang telah diketahui sebelumnya, metode ini mungkin memerlukan teknik tambahan untuk memastikan eksplorasi lingkungan yang cukup untuk menemukan kebijakan yang lebih baik.

Terlepas dari tantangan ini, metode Actor–Critic tetap menjadi landasan DRL modern, menawarkan kerangka kerja yang fleksibel dan terukur untuk pembelajaran di lingkungan yang kompleks.

10.5 EKSPLORASI DAN EKSPLOITASI

Dalam RL, agen harus belajar untuk membuat keputusan yang memaksimalkan imbalan jangka panjang. Namun, proses pembelajaran ini melibatkan pertukaran mendasar: agen harus mengeksplorasi tindakan baru untuk menemukan strategi yang berpotensi lebih baik, sambil juga memanfaatkan tindakan yang diketahui yang menghasilkan imbalan tinggi. Keseimbangan antara eksplorasi dan eksploitasi ini sangat penting untuk desain sistem RL yang efisien dan adaptif. Pada bagian ini, kita akan membahas prinsip, strategi, dan tantangan yang terkait dengan pengelolaan pertukaran ini.

- **Memahami Pertukaran:** Baik strategi eksplorasi maupun eksploitasi sangat penting dan sentral untuk desain yang efisien. Eksplorasi melibatkan mencoba tindakan yang tidak dikenal untuk mengumpulkan informasi tentang lingkungan. Eksploitasi, di sisi lain, melibatkan memanfaatkan pengetahuan yang ada untuk memilih tindakan yang diketahui menghasilkan hasil yang menguntungkan. Menemukan keseimbangan yang tepat sangat penting: terlalu banyak eksplorasi dapat menyebabkan inefisiensi dan kinerja suboptimal, sementara eksploitasi yang berlebihan dapat mencegah agen menemukan strategi yang lebih baik.

Dalam lingkungan stokastik, di mana hasilnya tidak pasti dan dinamis, agen harus mengambil sampel tindakan yang cukup untuk memperkirakan imbalan yang diharapkan secara akurat. Hal ini membutuhkan desain strategi eksplorasi yang cermat yang memungkinkan cakupan luas ruang keadaan tanpa mengorbankan efisiensi pembelajaran.

- **Strategi Eksploitasi:** Eksploitasi berfokus pada memaksimalkan imbalan jangka pendek berdasarkan pemahaman agen saat ini tentang lingkungan. Aspek-aspek kuncinya meliputi:
 - **Maksimisasi Imbalan:** Memilih tindakan yang menghasilkan imbalan yang diharapkan tertinggi.
 - **Efisiensi Keputusan:** Mengurangi beban komputasi dengan berfokus pada tindakan yang diketahui memberikan imbalan tinggi.
 - **Penghindaran Risiko:** Menghindari tindakan yang tidak pasti atau kurang dipahami. Teknik eksploitasi umum meliputi:
 - **Algoritma Serakah:** Selalu memilih tindakan dengan imbalan yang diperkirakan tertinggi.
 - **Eksploitasi Berbasis Kebijakan:** Menggunakan kebijakan yang telah dipelajari sebelumnya untuk memandu pengambilan keputusan.
 - **Metode Berbasis Model:** Memanfaatkan model prediktif untuk memilih tindakan dengan pengembalian yang diharapkan tinggi.

Eksplorasi juga melibatkan tingkat risiko yang lebih rendah karena bergantung pada tindakan yang telah dicoba dan diuji sehingga menghindari ketidakpastian.

- **Strategi Eksplorasi:** Eksplorasi bertujuan untuk meningkatkan pemahaman agen tentang lingkungan dengan memilih tindakan dengan hasil yang tidak pasti. Aspek-aspek kuncinya meliputi:
 - **Perolehan Informasi:** Mengumpulkan data baru untuk menyempurnakan model agen tentang lingkungan.
 - **Pengurangan Ketidakpastian:** Memprioritaskan tindakan yang mengurangi ambiguitas dalam estimasi imbalan.
 - **Cakupan Ruang Keadaan:** Memastikan pengalaman yang beragam untuk menghindari pembelajaran yang bias.

Teknik eksplorasi populer meliputi:

- **Strategi Epsilon-Greedy:** Dengan probabilitas ϵ , pilih tindakan acak (eksplorasi); dengan probabilitas $1 - \epsilon$, pilih tindakan terbaik yang telah diketahui (eksploitasi).
- **Pengambilan Sampel Thompson:** Menggunakan metode Bayesian untuk menyeimbangkan eksplorasi dan eksploitasi berdasarkan ketidakpastian dalam estimasi imbalan.
- **Strategi Epsilon-Greedy:** Strategi epsilon-greedy adalah salah satu metode yang paling banyak digunakan untuk menyeimbangkan eksplorasi dan eksploitasi. Dalam pendekatan ini:
 - Agen memilih tindakan acak dengan probabilitas ϵ (eksplorasi).
 - Dengan probabilitas $1 - \epsilon$, agen memilih tindakan dengan perkiraan imbalan tertinggi (eksploitasi).

Setelah setiap tindakan, agen memperbarui perkiraan imbalannya untuk tindakan tersebut menggunakan:

$$Q(a) \leftarrow Q(a) + \frac{1}{N(a)}(R - Q(a))$$

di mana:

- $Q(a)$: Perkiraan imbalan rata-rata untuk tindakan a
- $N(a)$: Jumlah kali tindakan a telah dipilih
- R : Imbalan yang diterima setelah melakukan tindakan a

Pembaruan inkremental ini membantu agen untuk menyempurnakan perkiraan imbalannya dari waktu ke waktu.

- **Tantangan dalam Menyeimbangkan Eksplorasi dan Eksploitasi:** Mencapai keseimbangan yang tepat bukanlah hal yang sepele dan bergantung pada konteks. Tantangan utama meliputi:
 - **Eksplorasi Berlebihan:** Membuang sumber daya pada tindakan dengan imbalan rendah.

- **Eksplorasi Kurang:** Melewatkan peluang untuk memaksimalkan imbalan yang diketahui.
- **Skalabilitas:** Mengelola trade-off secara efisien di lingkungan yang besar atau kompleks.
- **Keterbatasan Sumber Daya:** Keterbatasan waktu atau daya komputasi dapat membatasi eksplorasi.

Untuk mengatasi tantangan ini, para pengembang telah mengembangkan teknik-teknik canggih seperti:

- **Penyetelan Parameter Dinamis:** Menyesuaikan tingkat eksplorasi berdasarkan perubahan kinerja dan lingkungan.
- **Kerangka Kerja Multi-Armed Bandit:** Menganalisis trade-off di bawah struktur imbalan yang berbeda.
- **Pendekatan Hierarkis:** Mengatur tindakan dan kebijakan dalam lapisan untuk mengoptimalkan pencarian dan pembelajaran.

Meskipun Q-learning efektif dalam lingkungan diskrit dan berdimensi rendah, metode ini menjadi tidak praktis ketika ruang keadaan besar atau kontinu. Untuk mengatasi keterbatasan ini, DQN memperluas Q-learning dengan menggunakan DNN untuk mendekati fungsi Q . Inovasi ini memungkinkan agen untuk mempelajari kebijakan optimal secara langsung dari input berdimensi tinggi seperti gambar, data sensor, atau representasi keadaan yang kompleks. Pada bagian selanjutnya, kita akan mengeksplorasi arsitektur, proses pelatihan, dan peningkatan yang menjadikan DQN sebagai teknik dasar dalam DRL.

10.6 DEEP Q-NETWORK (DQN)

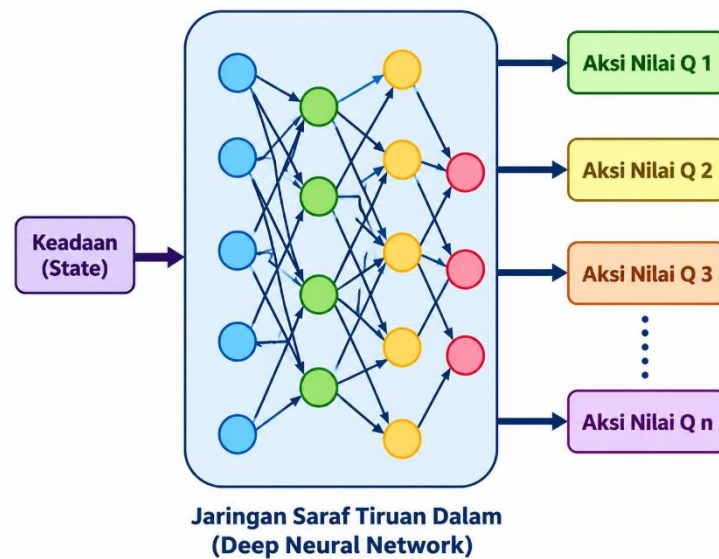
DQN menggabungkan prinsip-prinsip Q-learning dengan DL untuk menangani lingkungan di mana nilai Q tabular tidak memungkinkan. Alih-alih mempertahankan tabel Q , DQN menggunakan jaringan saraf untuk mendekati fungsi Q :

$$Q(s, a; \theta)$$

di mana:

- s : Keadaan saat ini
- a : Aksi
- θ : Parameter (bobot) jaringan saraf.

Jaringan tersebut menerima keadaan saat ini sebagai input dan menghasilkan nilai Q untuk semua aksi yang mungkin (lihat Gambar 10.4). Agen memilih aksi menggunakan strategi eksplorasi (misalnya, ϵ -greedy), berinteraksi dengan lingkungan, dan memperbarui jaringan berdasarkan imbalan yang diamati.



Gambar 10.4 Arsitektur DQN. (Diadaptasi dari.

Komponen utama DQN meliputi:

- **Aproksimasi Nilai Q :** DNN menggantikan tabel Q , memungkinkan generalisasi di seluruh keadaan yang serupa.
- **Pemutaran Ulang Pengalaman:** Sebuah buffer menyimpan pengalaman masa lalu (s, a, r, s'), dan mini batch diambil secara acak selama pelatihan. Ini memecah korelasi antara pengalaman berurutan dan meningkatkan stabilitas pembelajaran.
- **Jaringan Target:** Salinan terpisah dari jaringan Q digunakan untuk menghitung nilai Q target. Jaringan ini diperbarui lebih jarang untuk menstabilkan pelatihan.
- **Fungsi Kerugian:** Jaringan dilatih untuk meminimalkan perbedaan antara nilai Q yang diprediksi dan nilai Q target:

$$L(\theta) = \mathbb{E} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right)^2 \right]$$

di mana θ^- menyatakan parameter jaringan target.

- **Proses Pelatihan:** Siklus pelatihan DQN melibatkan langkah-langkah berikut:
 1. Inisialisasi buffer replay, jaringan utama, dan jaringan target θ^- .
 2. Pilih tindakan menggunakan kebijakan ϵ -greedy untuk menyeimbangkan eksplorasi dan eksploitasi.
 3. Berinteraksi dengan lingkungan, kumpulkan pengalaman, dan simpan di buffer replay.
 4. Ambil sampel mini-batch dari buffer dan hitung nilai Q target menggunakan jaringan target.

5. Perbarui jaringan utama dengan meminimalkan fungsi kerugian menggunakan penurunan gradien.
6. Perbarui jaringan target secara berkala dengan menyalin bobot dari jaringan utama.
7. Secara bertahap kurangi ϵ untuk beralih dari eksplorasi ke eksploitasi.

Penggunaan DNN memungkinkan DQN untuk memproses dan belajar dari input kompleks dan berdimensi tinggi. Selain itu, pengalaman pemutaran ulang dan jaringan target keduanya secara signifikan meningkatkan stabilitas proses pembelajaran dibandingkan dengan Q-learning tradisional.

DQN telah berhasil diterapkan pada berbagai domain, terutama dalam menguasai permainan Atari dan menunjukkan potensi di bidang seperti robotika dan sistem otonom. Selama beberapa waktu, DQN telah mengalami banyak peningkatan. Arsitektur yang lebih baik telah diusulkan untuk memberikan kinerja dan stabilitas yang lebih besar.

- **Double DQN:** DQN standar cenderung melebih-lebihkan nilai Q , yang dapat menyebabkan kebijakan yang suboptimal. Double DQN mengatasi hal ini dengan memisahkan pemilihan dan evaluasi tindakan:

$$Q_{\text{target}} = r + \gamma Q(s', \arg\max_a Q(s', a; \theta); \theta^-)$$

Di sini, jaringan utama memilih tindakan, dan jaringan target θ^- mengevaluasinya. Ini mengurangi estimasi berlebihan dan mempercepat pelatihan.

- **Dueling DQN:** Dueling DQN memisahkan estimasi nilai keadaan $V(s)$ dan keuntungan dari setiap tindakan $A(a)$:

$$Q(s, a) = V(s) + A(a)$$

Arsitektur ini memungkinkan agen untuk mempelajari keadaan mana yang berharga secara independen dari tindakan yang diambil, sehingga meningkatkan efisiensi dan ketahanan pembelajaran.

10.7 RINGKASAN

Dalam bab ini, kita telah mengeksplorasi dasar-dasar dan kemajuan DRL – perluasan yang ampuh dari RL tradisional yang mengintegrasikan DL untuk menangani lingkungan kompleks dan berdimensi tinggi. DRL memungkinkan agen untuk mempelajari perilaku optimal secara langsung dari input sensorik mentah, seperti gambar atau data sensor, sehingga cocok untuk aplikasi dunia nyata dalam game, robotika, sistem otonom, perawatan kesehatan, dan keuangan. Kita mulai dengan meninjau kembali *Q-learning*, metode berbasis nilai yang mempelajari tindakan optimal melalui uji coba dan kesalahan. Meskipun efektif di lingkungan diskrit, Q-learning mengalami kesulitan dalam hal skalabilitas. Keterbatasan ini menyebabkan pengembangan DQN, yang menggunakan DNN untuk mendekati nilai Q dan menggabungkan

teknik seperti pemutaran ulang pengalaman dan jaringan target untuk meningkatkan stabilitas dan kinerja.

Selanjutnya, kita meneliti *metode PG*, yang secara langsung mengoptimalkan kebijakan agen menggunakan peningkatan gradien. Metode-metode ini sangat berguna dalam ruang aksi kontinu dan menjadi dasar bagi banyak algoritma DRL modern. Untuk mengatasi varians yang tinggi dalam estimasi PG, kami memperkenalkan *metode Aktor-Kritikus*, yang menggabungkan kekuatan pendekatan berbasis nilai dan berbasis kebijakan. Aktor memilih tindakan, sementara Kritikus mengevaluasinya, memungkinkan pembelajaran yang lebih stabil dan efisien dalam penggunaan sampel. Kami juga membahas varian A2C, yang menggunakan fungsi keuntungan untuk menyempurnakan pembaruan kebijakan. Aspek penting dari DRL adalah mengelola *trade-off eksplorasi-eksploitasi*. Kami mengeksplorasi strategi seperti ϵ -greedy dan pengambilan sampel Thompson, yang membantu agen menyeimbangkan kebutuhan untuk menemukan strategi baru dengan tujuan memaksimalkan imbalan yang diketahui.

Terakhir, kami membahas arsitektur dan proses pelatihan DQN, beserta peningkatannya – *Double DQN* dan *Dueling DQN* – yang mengatasi estimasi berlebihan dan meningkatkan efisiensi pembelajaran. Bersama-sama, konsep-konsep ini membentuk kerangka kerja komprehensif untuk membangun agen cerdas yang mampu belajar dan beradaptasi di lingkungan yang kompleks. Pada bab selanjutnya, kita akan mengeksplorasi bagaimana teknik-teknik ini dapat diterapkan pada sistem dunia nyata dan membahas tren yang muncul dalam pengembangan RL.

10.8 PERTANYAAN

1. Bandingkan Reinforcement Learning (RL) dan Deep Reinforcement Learning (DRL). Analisis peran Deep Neural Networks (DNN) dalam DRL dan jelaskan mengapa DRL sesuai untuk permasalahan kompleks berdimensi tinggi.
2. Berikan penjelasan rinci tentang algoritma Q-learning. Diskusikan bagaimana persamaan Bellman mendukung pembaruan nilai Q dan periksa secara kritis peran trade-off eksplorasi-eksploitasi dalam Q-learning.
3. Analisis metode Policy Gradient dan kerangka kerja Actor-Critic. Jelaskan bagaimana komponen Actor dan Critic berkolaborasi untuk meningkatkan efisiensi dan stabilitas pembelajaran.
4. Dalam strategi epsilon-greedy, epsilon (ϵ) mengontrol
 - a. Tingkat pembelajaran untuk memperbarui nilai Q
 - b. Probabilitas eksplorasi tindakan acak
 - c. Faktor diskon untuk imbalan di masa mendatang
 - d. Tingkat pembaruan jaringan target di DQN
5. Komponen mana dalam Deep Q-Networks (DQN) yang membantu mengurangi korelasi antara pengalaman berurutan dan meningkatkan stabilitas pelatihan?
 - a. Policy Gradient
 - b. Experience Replay Buffer

- c. Jaringan Target
 - d. Fungsi Keunggulan
6. [Latihan Pemrograman – Menengah] Ubah skrip Python Q-learning (Tabel 10.2) untuk mencatat nilai Q setelah setiap 100 episode. Implementasikan visualisasi tentang bagaimana nilai Q ini konvergen dari waktu ke waktu.
 7. [Latihan Pemrograman – Lanjutan] Implementasikan algoritma Actor–Critic yang disederhanakan menggunakan pustaka pembelajaran mendalam seperti TensorFlow atau PyTorch untuk lingkungan CartPole. Latih agen minimal selama 200 episode dan buat plot yang menampilkan hadiah episode dari waktu ke waktu.

BAB 11

PEMODELAN BAHASA ALAMI

11.1 PEMROSESAN BAHASA ALAMI (NLP)

Pemrosesan Bahasa Alami (NLP) adalah subbidang Kecerdasan Buatan (AI) yang memungkinkan komputer untuk memahami, menafsirkan, dan menghasilkan bahasa manusia. Ini berfungsi sebagai jembatan antara komunikasi manusia dan kecerdasan mesin, memungkinkan sistem untuk memproses dan menanggapi teks dan ucapan dengan cara yang bermakna. Dalam pengertian luas, *bahasa alami* adalah cara manusia berkomunikasi melalui tuturan maupun tulisan. Data tekstual yang umumnya tidak terstruktur tersedia dalam jumlah besar pada situs web, surel, percakapan daring, blog, buku, dan media sosial. Sebagaimana dibahas pada Bab 1, data tidak terstruktur, termasuk teks, audio, dan video, memiliki nilai penting dalam sistem Big Data. Sebagai contoh, platform media sosial seperti Twitter (kini X) memuat informasi emosional dan kontekstual yang dapat dianalisis untuk kajian sentimen, deteksi tren, dan dukungan keputusan. Karena itu, diperlukan metode untuk memahami dan menalar informasi lisan atau tertulis sebagaimana data numerik.

Meskipun telah ada kemajuan yang signifikan, bekerja dengan bahasa alami tetap merupakan tugas yang menantang bagi mesin. NLP berfokus pada manipulasi bahasa secara otomatis, bertujuan untuk mengekstrak informasi yang bermakna dari teks dan ucapan. Ini mencakup dua subdomain utama:

- **Pemahaman Bahasa Alami:** Menafsirkan dan mengekstrak makna dari bahasa manusia.
- **Generasi Bahasa Alami:** Menghasilkan keluaran bahasa yang koheren dan sesuai konteks.

Bersama-sama, kemampuan ini memungkinkan aplikasi seperti chatbot, asisten virtual, penerjemahan mesin, dan mesin pencari cerdas.

11.2 TAHAP-TAHAP NLP

NLP melibatkan serangkaian tahap analitis yang memungkinkan mesin untuk memahami dan menafsirkan bahasa manusia. Fase-fase ini – Analisis Leksikal, Analisis Sintaksis, Analisis Semantik, Integrasi Wacana, dan Analisis Pragmatik – bekerja bersama untuk mengekstrak makna dari teks secara terstruktur dan sadar konteks. Setiap fase dibangun di atas fase sebelumnya, secara progresif menyempurnakan pemahaman mesin tentang bahasa. Berikut adalah gambaran singkat dari setiap fase:

1. **Analisis Leksikal:** Analisis leksikal adalah fase awal NLP, di mana teks masukan dipecah menjadi unit yang lebih kecil yang disebut leksim, seperti kata, tanda baca, atau simbol. Fase ini meliputi:
 - **Tokenisasi:** Memisahkan teks menjadi kata-kata atau token individual.
 - **Stemming:** Mengurangi kata ke bentuk akarnya dengan menghilangkan imbuhan.

- **Lematisasi:** Memetakan kata ke bentuk kamusnya (lemma), dengan mempertimbangkan konteks dan bagian ucapan.
Analisis leksikal mempersiapkan teks mentah untuk pemrosesan sintaksis dan semantik lebih lanjut.
- 2. **Analisis Sintaksis:** Juga dikenal sebagai *parsing*, analisis sintaksis memeriksa struktur tata bahasa kalimat. Ini mengidentifikasi bagaimana kata-kata saling berhubungan dan mengorganisasikannya menjadi frasa, klausa, dan struktur kalimat hierarkis. Fase ini memastikan bahwa teks masukan mematuhi aturan tata bahasa dan membantu menentukan batas dan hubungan kalimat.
- 3. **Analisis Semantik:** Analisis semantik berfokus pada pemahaman makna literal kata, frasa, dan kalimat. Ini melampaui tata bahasa untuk menafsirkan makna yang disampaikan oleh teks. Fase ini memastikan bahwa elemen struktural yang diidentifikasi selama analisis sintaksis diproses dengan cara yang mencerminkan makna yang dimaksudkan, baik pada tingkat kata maupun kalimat.
- 4. **Integrasi Wacana:** Integrasi wacana menganalisis bagaimana kalimat-kalimat individual berhubungan satu sama lain dalam konteks yang lebih luas. Ini mempertimbangkan kalimat-kalimat di sekitarnya untuk menyelesaikan referensi (misalnya, kata ganti) dan untuk menjaga koherensi di seluruh bagian. Fase ini penting untuk memahami masukan multi-kalimat, seperti paragraf atau percakapan.
- 5. **Analisis Pragmatik:** Analisis pragmatik menafsirkan makna yang dimaksudkan dari teks dengan mempertimbangkan konteks, maksud pembicara, dan pengetahuan dunia nyata. Fase ini memperhitungkan nuansa seperti sarkasme, kesopanan, nada, dan makna tersirat. Fase ini sangat penting untuk aplikasi seperti chatbot dan analisis sentimen, di mana pemahaman isyarat halus sangat penting.

Bersama-sama, kelima fase ini membentuk tulang punggung sistem NLP. Fase-fase ini memungkinkan mesin untuk memproses bahasa dengan cara yang meniru pemahaman manusia, mendukung aplikasi seperti agen percakapan, terjemahan mesin, dan analisis teks cerdas..

11.3 ILMU LINGUISTIK

Linguistik adalah studi ilmiah tentang bahasa – struktur, penggunaan, dan evolusinya. Linguistik memberikan landasan teoritis untuk banyak teknik NLP dengan menawarkan wawasan tentang bagaimana bahasa dibentuk, dipahami, dan diinterpretasikan. Pada bagian ini, kita akan mengeksplorasi *metodologi ilmiah*, *ruang lingkup penyelidikan*, dan *konsep-konsep kunci* yang mendefinisikan bidang linguistik.

1. **Metodologi Ilmiah:** Para ahli bahasa menerapkan metode ilmiah yang ketat untuk mempelajari bahasa. Hal ini meliputi:
 - **Observasi:** Memeriksa bagaimana orang menggunakan bahasa dalam konteks lisan dan tulisan.
 - **Pengumpulan Data:** Mengumpulkan sampel bahasa dari percakapan alami, teks tertulis, dan eksperimen terkontrol.

- **Pengembangan Teori:** Merumuskan hipotesis tentang struktur dan penggunaan bahasa.
- **Pengujian dan Validasi:** Menganalisis data untuk mengkonfirmasi atau menyempurnakan teori linguistik.
- **Objektivitas dan Tinjauan Sejawat:** Menggunakan metode standar untuk memastikan keandalan dan meminimalkan bias.

Pendekatan empiris ini memungkinkan para ahli bahasa untuk mengidentifikasi pola dan menggeneralisasi prinsip-prinsip tentang bagaimana bahasa berfungsi di berbagai konteks dan populasi.

2. **Cakupan Pengembangan:** Linguistik mencakup berbagai subbidang yang mengkaji berbagai aspek bahasa. Aspek-aspek yang membentuk konsep dasar dalam linguistik meliputi:

- **Fonetik:** Studi tentang sifat fisik bunyi ucapan.
- **Fonologi:** Studi tentang pola dan aturan bunyi dalam bahasa.
- **Morfologi:** Studi tentang bagaimana kata-kata dibentuk dari unit-unit yang lebih kecil (morfem).
- **Sintaksis:** Aturan yang mengatur struktur kalimat dan urutan kata.
- **Semantik:** Studi tentang makna dalam kata, frasa, dan kalimat.
- **Pragmatik:** Studi tentang bagaimana makna dipengaruhi oleh konteks dan maksud pembicara.
- **Linguistik Historis:** Studi tentang perubahan dan evolusi bahasa.
- **Sosiolinguistik:** Studi tentang bagaimana bahasa bervariasi dan berfungsi dalam lingkungan sosial.

Konsep-konsep ini membentuk dasar bagi banyak tugas NLP, seperti penandaan bagian ujaran, penguraian sintaksis, dan analisis semantik.

11.4 CARA AI BELAJAR BAHASA

Setelah mengeksplorasi fase-fase dasar dan prinsip-prinsip linguistik di balik NLP, sekarang kita mengalihkan perhatian kita ke pendekatan pembelajaran yang mendukung sistem NLP modern. Pendekatan-pendekatan ini menentukan bagaimana mesin memperoleh kemampuan untuk memahami dan menghasilkan bahasa manusia. Secara umum, tugas-tugas NLP dapat ditangani menggunakan *sistem berbasis aturan*, *algoritma pembelajaran mesin (ML)*, atau *hibrida keduanya*. Setiap pendekatan memiliki kekuatan dan keterbatasannya, dan pilihan sering kali bergantung pada kompleksitas tugas, ketersediaan data, dan tingkat interpretasi yang diinginkan.

- **Pendekatan Berbasis Aturan:** Sistem berbasis aturan bergantung pada aturan dan pola linguistik yang dibuat secara manual untuk memproses dan menganalisis teks. Aturan-aturan ini dapat mencakup struktur tata bahasa, pola sintaksis, atau hubungan semantik yang didefinisikan oleh para ahli di bidangnya.

Langkah-langkah umum dalam alur kerja NLP berbasis aturan meliputi:

- **Pembuatan Aturan:** Mendefinisikan aturan untuk tugas-tugas seperti penandaan bagian ucapan atau pengenalan entitas bernama.
- **Penerapan Aturan:** Menerapkan aturan pada teks masukan untuk mengekstrak atau mengklasifikasikan informasi.
- **Penyempurnaan Aturan:** Meningkatkan aturan secara iteratif berdasarkan umpan balik dan kinerja.

Contohnya termasuk ekspresi reguler (regex) untuk pencocokan pola dan parser berbasis tata bahasa untuk analisis sintaksis. Sistem berbasis aturan bersifat transparan dan dapat diinterpretasikan, serta efektif untuk tugas-tugas yang terdefinisi dengan baik dengan struktur linguistik yang jelas. Namun, sistem ini sulit untuk diskalakan ke bahasa yang kompleks atau ambigu dan membutuhkan upaya manual yang signifikan untuk membuat dan memeliharanya.

- **Pendekatan Pembelajaran Mesin:** Metode ML mempelajari pola dari data daripada mengandalkan aturan yang telah ditentukan sebelumnya. Pendekatan ini telah menjadi dominan dalam NLP karena fleksibilitas dan kemampuannya untuk melakukan generalisasi di berbagai input bahasa.

Paradigma ML umum dalam NLP meliputi:

- **Pembelajaran Terawasi:** Model dilatih pada kumpulan data berlabel untuk melakukan tugas-tugas seperti klasifikasi, analisis sentimen, atau pengenalan entitas bernama.
- **Pembelajaran Tak Terawasi:** Algoritma menemukan pola tersembunyi dalam data yang tidak berlabel, berguna untuk pengelompokan, pemodelan topik, dan pengurangan dimensi.
- **Pembelajaran Penguatan (RL):** Agen mempelajari strategi optimal dengan berinteraksi dengan lingkungan dan menerima umpan balik. RL sangat berguna dalam sistem dialog dan tugas pengambilan keputusan.
- **Pembelajaran Mendalam:** Jaringan saraf, terutama arsitektur mendalam, telah merevolusi NLP dengan memungkinkan model untuk mempelajari representasi kompleks dan menangkap ketergantungan jarak jauh dalam teks.

Pendekatan pembelajaran mesin memiliki kemampuan untuk menangani fenomena bahasa yang kompleks, belajar dari kumpulan data besar, dan beradaptasi dengan data dan tugas baru. Namun, dibutuhkan sejumlah besar data untuk pembelajaran, khususnya dalam kasus metode pembelajaran terawasi dan mendalam, yang dapat memakan biaya komputasi yang tinggi, dan mungkin kurang dapat diinterpretasikan dibandingkan pendekatan berbasis aturan.

- **Pendekatan Hibrida:** Sistem hibrida menggabungkan teknik berbasis aturan dan pembelajaran mesin untuk memanfaatkan kekuatan keduanya. Misalnya, aturan dapat digunakan untuk prapemrosesan atau ekstraksi fitur, sementara model ML menangani klasifikasi atau prediksi. Pendekatan ini dapat mencapai akurasi dan ketahanan yang tinggi dengan menggabungkan kekuatan dan kelebihan dari kedua pendekatan.

Namun, dari segi desain dan implementasi, ini bisa menjadi pendekatan yang sangat kompleks.

11.5 TEKNIK DAN TUGAS NLP

NLP mencakup berbagai teknik yang memungkinkan mesin untuk memahami, menafsirkan, dan menghasilkan bahasa manusia. Teknik-teknik ini diterapkan di berbagai tugas, termasuk klasifikasi teks, ekstraksi informasi, analisis sentimen, penerjemahan mesin, dan peringkasan. Berikut adalah gambaran umum teknik-teknik NLP utama dan tugas-tugas terkaitnya.

- **Tokenisasi:** Tokenisasi adalah proses memecah teks menjadi unit-unit yang lebih kecil yang disebut *token*, yang dapat berupa kata, subkata, atau karakter tergantung pada aplikasinya. Ini adalah langkah dasar dalam sebagian besar alur kerja NLP dan sangat penting untuk tugas-tugas seperti penerjemahan mesin, pengambilan informasi, dan analisis teks.
- **Stemming dan Lemmatisasi:** Baik Stemming maupun lemmatisasi adalah teknik normalisasi teks yang mengurangi kata-kata ke bentuk akarnya. Meskipun keduanya memiliki tujuan yang sama, keduanya berbeda dalam pendekatannya. Stemming adalah proses berbasis aturan yang menghilangkan awalan dan akhiran, yang berpotensi menghasilkan kata-kata non-kamus. Di sisi lain, lemmatisasi mempertimbangkan konteks dan bagian ujaran, menggunakan kamus untuk menemukan bentuk dasar atau bentuk kamus dari sebuah kata (lemma), sehingga menghasilkan hasil yang lebih akurat.
- **Penghapusan Kata Hentian (Stop Word Removal):** Kata hentian adalah kata-kata umum (misalnya, "the, " "is, " "and") yang memiliki bobot semantik yang kecil. Menghapusnya selama prapemrosesan membantu mengurangi kebisingan, meningkatkan efisiensi komputasi, dan meningkatkan kinerja model dalam tugas-tugas seperti klasifikasi teks dan pencarian.
- **Penandaan Bagian Ujaran (Part-of-Speech/POS Tagging):** Penandaan POS menetapkan kategori gramatikal (misalnya, kata benda, kata kerja, dan kata sifat) untuk setiap kata dalam sebuah kalimat. Ini membantu mesin memahami struktur kalimat dan peran sintaksis setiap kata, yang sangat penting untuk penguraian, ekstraksi informasi, dan menjawab pertanyaan.
- **Pengenalan Entitas Bernama (NER):** NER mengidentifikasi dan mengklasifikasikan entitas bernama dalam teks ke dalam kategori yang telah ditentukan sebelumnya seperti nama orang, organisasi, lokasi, tanggal, dan nilai moneter. Ini adalah komponen kunci dari ekstraksi informasi dan banyak digunakan dalam aplikasi seperti mesin pencari, chatbot, dan sistem rekomendasi.
- **Penyematan Kata (Word Embeddings):** Penyematan kata adalah representasi vektor padat dari kata-kata yang menangkap hubungan semantik. Kata-kata dengan makna yang serupa diposisikan berdekatan dalam ruang vektor. Penyematan merupakan dasar untuk tugas NLP berbasis pembelajaran mendalam seperti analisis sentimen,

terjemahan mesin, dan AI percakapan. Bab 12 akan mengeksplorasi teknik penyematan populer seperti *Word2Vec* dan *GloVe*.

- **Analisis Sentimen:** Analisis sentimen menentukan nada emosional yang diekspresikan dalam sebuah teks – apakah itu positif, negatif, atau netral. Ini banyak digunakan dalam analisis umpan balik pelanggan, pemantauan media sosial, dan manajemen reputasi merek.
- **Peringkasan Teks:** Peringkasan teks mengurangi dokumen panjang menjadi ringkasan yang ringkas sambil mempertahankan informasi kunci. Ini dapat bersifat *ekstraktif* (memilih kalimat kunci) atau *abstraktif* (menghasilkan kalimat baru). Teknik ini berharga di bidang seperti hukum, perawatan kesehatan, dan keuangan, di mana sintesis informasi yang cepat sangat penting.
- **Terjemahan Mesin:** Terjemahan mesin secara otomatis mengkonversi teks dari satu bahasa ke bahasa lain. Ia menggunakan teknik NLP untuk menganalisis struktur dan semantik bahasa sumber, kemudian menghasilkan teks yang setara dalam bahasa target. Sistem modern sering mengandalkan model pembelajaran mendalam seperti transformer.
- **Pemodelan Topik:** Pemodelan topik adalah teknik pembelajaran tanpa pengawasan yang mengidentifikasi struktur tematik tersembunyi dalam kumpulan dokumen. Ini membantu mengatur dan meringkas korpus teks besar dengan menemukan topik abstrak berdasarkan pola kemunculan bersama kata.

Teknik-teknik ini membentuk blok bangunan sistem NLP modern. Pada bagian selanjutnya, kita akan membandingkan pendekatan *NLP klasik* dengan *NLP berbasis pembelajaran mendalam*, menyoroti kekuatan, keterbatasan, dan kasus penggunaannya.

11.6 NLP TRADISIONAL VS AI MODERN

NLP telah berevolusi secara signifikan selama beberapa dekade terakhir. Secara tradisional, NLP bergantung pada metode berbasis aturan dan statistik, yang sering disebut sebagai NLP klasik. Baru-baru ini, pendekatan pembelajaran mendalam telah mengubah bidang ini dengan memungkinkan model untuk belajar langsung dari data, menangkap pola kompleks dan nuansa kontekstual. Pada bagian ini, kami membandingkan NLP klasik dan NLP pembelajaran mendalam dalam hal metodologi, kemampuan, dan trade-off.

- **NLP Klasik:** Metode NLP klasik sangat bergantung pada aturan yang ditentukan manusia dan rekayasa fitur. Model-model ini menggunakan aturan linguistik dan teknik statistik untuk menganalisis dan memproses teks. Alat dan teknik umum meliputi:
 - **Ekspresi reguler** untuk pencocokan pola
 - **Penandaan bagian ucapan** menggunakan Model Markov Tersembunyi
 - **NER** menggunakan sistem berbasis aturan
 - **Analisis sentimen** menggunakan leksikon dan fitur buatan tangan
 - **Ekstraksi fitur** menggunakan Bag of Words dan frekuensi istilah–frekuensi dokumen terbalik

Model NLP klasik sering kali dapat diinterpretasikan dan efisien secara komputasi, sehingga cocok untuk aplikasi skala kecil atau tugas yang terdefinisi dengan baik. Namun, mereka kesulitan dengan ambiguitas, sarkasme, dan makna yang bergantung pada konteks, yang umum dalam bahasa alami. Ada upaya manual yang signifikan untuk memelihara dan memperbarui aturan, dan model memiliki skalabilitas dan adaptabilitas yang terbatas.

- **NLP Pembelajaran Mendalam:** Model NLP pembelajaran mendalam menggunakan jaringan saraf – sering kali dengan banyak lapisan – untuk mempelajari fitur langsung dari data. Model ini dapat menangkap pola kompleks dan hubungan semantik tanpa definisi aturan eksplisit. Arsitektur populer meliputi:
 - Jaringan Saraf Berulang (Recurrent Neural Networks) untuk data sekuensial
 - Transformer untuk pemodelan bahasa yang sadar konteks
 - Jaringan Saraf Konvolusional (Convolutional Neural Networks) untuk klasifikasi teks

Pendekatan NLP berbasis pembelajaran mendalam telah merevolusi tugas seperti penerjemahan mesin, analisis sentimen, dan penjawaban pertanyaan. Model secara otomatis mempelajari fitur dari kumpulan data besar, menangani ambiguitas dan konteks, menangkap ketergantungan jarak jauh, serta dapat diskalakan untuk tugas kompleks dan korpus multibahasa. Namun, model tersebut memerlukan data berlabel dalam jumlah besar, komputasi yang intensif, sumber daya yang tinggi, dan umumnya lebih sulit diinterpretasikan daripada sistem berbasis aturan.

Singkatnya, NLP klasik dan NLP berbasis pembelajaran mendalam sama-sama digunakan untuk memproses data teks. NLP klasik mengandalkan aturan dan fitur yang dirancang secara manual, sehingga lebih efisien dan mudah diinterpretasikan untuk tugas sederhana atau kumpulan data kecil. Sebaliknya, NLP berbasis pembelajaran mendalam menggunakan jaringan saraf untuk mempelajari fitur langsung dari data dan mampu menangani tugas serta nuansa bahasa yang lebih kompleks, tetapi memerlukan kumpulan data besar dan sumber daya komputasi yang tinggi. Pada bagian selanjutnya, kita akan memperkenalkan konsep model bahasa.

11.7 MODEL BAHASA

Model bahasa merupakan komponen dasar NLP. Model ini dirancang untuk memahami, menghasilkan, dan memprediksi bahasa manusia dengan memberikan probabilitas pada urutan kata. Model ini memungkinkan mesin untuk memperkirakan kemungkinan suatu kata atau frasa muncul dalam konteks tertentu, yang sangat penting untuk tugas-tugas seperti pembuatan teks, koreksi ejaan, terjemahan mesin, dan pengenalan ucapan. Pada intinya, model bahasa melakukan dua tugas utama:

- **Prediksi Kata Berikutnya:** Diberikan urutan kata, model memprediksi kata berikutnya yang paling mungkin.
- **Estimasi Probabilitas Urutan:** Model memberikan probabilitas pada seluruh urutan kata, membantu menentukan seberapa alami atau mungkin urutan tersebut.

Secara formal, dengan diberikan urutan kata $w_1, w_2, \dots, w_l$, sebuah model bahasa memperkirakan probabilitas gabungan:

$$P(W) = P(w_1, w_2, \dots, w_l)$$

Probabilitas ini mencerminkan seberapa besar kemungkinan rangkaian tersebut terjadi dalam bahasa alami. Model bahasa digunakan dalam berbagai aplikasi NLP, termasuk:

- **Koreksi Ejaan:** Memprediksi kata yang benar berdasarkan konteks.
- **Pengenalan Ucapan:** Menentukan transkripsi masukan lisan yang paling mungkin.
- **Terjemahan Mesin:** Menghasilkan terjemahan yang lancar dengan memodelkan kefasihan bahasa target.
- **Text Generation:** Menghasilkan teks yang koheren dan sesuai konteks.

Misalnya, dalam koreksi ejaan, model memprediksi kata w_l berdasarkan kata sebelumnya $w_1, w_2, \dots, w_{l-1}$, menggunakan:

$$P(w_l | w_{l-1}, w_{l-2}, \dots, w_1)$$

Untuk mengevaluasi kualitas model bahasa, *perpleksitas* umumnya digunakan. Perpleksitas mengukur seberapa baik model memprediksi sampel teks. Perpleksitas yang lebih rendah menunjukkan kinerja prediksi yang lebih baik. Pada bagian selanjutnya, kita akan mengeksplorasi *model N-gram*, yang merupakan model bahasa statistik sederhana namun ampuh yang memperkirakan probabilitas urutan kata berdasarkan riwayat kata dengan panjang tetap.

11.8 MODEL N-GRAM

Model *N-gram* dapat digunakan untuk merepresentasikan kata. Model ini adalah model paling sederhana yang menetapkan distribusi probabilitas pada urutan N kata. *N-gram* dapat dianggap sebagai urutan N kata, dengan pengertian itu, 2-gram (atau bigram) adalah urutan dua kata sedangkan 3-gram (atau trigram) adalah urutan tiga kata, dan seterusnya. Untuk memperjelas konsep *N-gram*, digunakan probabilitas bersyarat. Jika A dan B merupakan dua kejadian dengan $P(B) \neq 0$, maka $P(A | B) = \frac{P(A \cap B)}{P(B)}$. Dengan menerapkan aturan rantai pada urutan kata $w_1, w_2, \dots, w_l$, probabilitas gabungan diberikan sebagai:

$$P(w_1, w_2, \dots, w_l) = P(w_1)P(w_2 | w_1) \cdots \\ \times P(w_l | w_{l-1}, w_{l-2}, \dots, w_1)$$

Hal ini dapat direpresentasikan lebih lanjut sebagai

$$P(w_1, w_2, \dots, w_l) = \prod_{i=1}^l P(w_i | w_{i-1}, w_{i-2}, \dots, w_1) \quad (11.1)$$

Sekarang mari kita definisikan konsep *unigram*, *bigram*, dan *model N-gram* berdasarkan ekspresi (1) yang diberikan di atas.

Model unigram mempertimbangkan kata-kata individual dan membagi probabilitas kata-kata yang berbeda dalam suatu konteks. Misalnya, kalimat yang mengandung l kata akan diprediksi dengan probabilitas:

$$P(w_1, w_2, \dots, w_l) \approx \prod_{i=1}^l P(w_i) \quad (11.2)$$

$$P(w_i) \approx \frac{C(w_i)}{l}$$

di mana $C(w_i)$ adalah jumlah kemunculan w_i dan l adalah jumlah seluruh kata dalam korpus. Model bigram mempertimbangkan kata berurutan atau dua kata sekaligus. Sebuah kalimat dengan l kata akan menghasilkan $l - 1$ bigram dan akan diprediksi dengan probabilitas:

$$P(w_1, w_2, \dots, w_l) \approx \prod_{i=2}^l P(w_i | w_{i-1}) \quad (11.3)$$

Dengan menggunakan Estimasi Kemungkinan Maksimum, probabilitas bigram dapat diestimasi sebagai:

$$P(w_i | w_{i-1}) \approx \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$

Pada dasarnya, model bigram memprediksi kata berikutnya berdasarkan kata sebelumnya. Model ini menggunakan kosakata, bigram, dan distribusi probabilitas untuk menghitung kemungkinan suatu kata.

Sekarang, mari kita definisikan model *N-gram*. Model *N-gram* mempertimbangkan urutan N kata berurutan. Probabilitas bersyarat dari model *N-gram* dapat dihitung sebagai:

$$P(w_1, w_2, \dots, w_l) \approx \prod_{i=n}^l P(w_i | w_{i-1}, w_{i-2}, \dots, w_{i-n}) \quad (11.4)$$

Probabilitas *N-gram* dapat diperkirakan sebagai:

$$P(w_i | w_{i-1}w_{i-2}) \approx \frac{C(w_{i-2}w_{i-1}w_i)}{C(w_{i-2}w_{i-1})}$$

Kinerja model bahasa sepenuhnya bergantung pada kualitas kumpulan data (korpus). Secara khusus, model N -gram hanya bekerja dengan baik ketika korpus pelatihan dan pengujian serupa. Dengan demikian, model bahasa N -gram akan mengalami overfitting. Pada korpus berukuran lebih kecil, N -gram menjadi sangat jarang, dan jumlahnya mungkin bernilai 0. Model bahasa akan menetapkan probabilitas nol untuk N -gram tersebut selama proses pelatihan. Ini adalah situasi yang bermasalah jika N -gram tersebut muncul dalam data uji karena probabilitas suatu urutan sama dengan hasil perkalian probabilitas individual dari N -gram. N -gram dengan probabilitas bernilai 0 akan menetapkan probabilitas urutan tersebut menjadi 0. Untuk mengatasi masalah ini, jumlah setiap kemungkinan N -gram dapat diatur menjadi 1 untuk menjamin bahwa setiap N -gram akan memiliki probabilitas kecil. Ini dikenal sebagai perataan Laplace. Perataan Laplace menggeser distribusi jumlah kata dan memberikan peningkatan yang sederhana pada hasil dalam masalah klasifikasi teks.

Dalam model N -gram, mungkin ada skenario di mana N -gram ada dalam kumpulan data pengujian tetapi tidak ada dalam kumpulan data pelatihan. Skenario ini dikenal sebagai out-of-vocabulary (OOV). Ada beberapa cara untuk menangani situasi seperti itu. Salah satu cara untuk mengatasi masalah ini adalah dengan mengganti semua kata (dalam kumpulan data pengujian) yang tidak ada dalam kosakata dengan token khusus yang direpresentasikan sebagai *<unk>*. Dalam hal ini, semua kata OOV diperlakukan sebagai satu entitas tunggal, mengabaikan informasi linguistik. Pilihan lain adalah mengganti atau memetakan semua kata atau N -gram OOV ke N -gram terdekat yang ada dalam kosakata. Sekarang, mari kita jelaskan bagaimana mengukur kinerja model bahasa. Di sini, kita mendefinisikan istilah perplexitas sebagai ukuran untuk mengevaluasi kinerja model bahasa.

Secara umum, *perpleksitas* mengukur seberapa baik distribusi probabilitas memprediksi sampel. Model bahasa terbaik adalah model yang paling baik memprediksi kalimat uji yang belum pernah dilihat sebelumnya. Perpleksitas mengukur kualitas intrinsik model bahasa sebagai fungsi dari probabilitas $PP(w)$ bahwa model memprediksi urutan kata uji w seperti $w = w_1, w_2, \dots, w_l$

$$PP(w) = P(w_1, w_2, \dots, w_l)^{-1/l} \quad (11.5)$$

Berdasarkan ekspresi (11.5), perplexitas dapat didefinisikan sebagai probabilitas invers dari urutan uji yang dinormalisasi oleh jumlah kata. Meminimalkan perplexitas sama dengan memaksimalkan probabilitas. Pada bagian selanjutnya, kita akan mengeksplorasi *semantik vektor dan embedding*, yang memberikan representasi kata yang lebih kuat dan fleksibel dengan menangkap hubungan kontekstualnya dalam ruang berdimensi tinggi.

11.9 SEMANTIK VEKTOR DAN EMBEDDING

Semantik vektor adalah pendekatan yang ampuh dalam NLP yang berfokus pada representasi makna kata sebagai vektor dalam ruang berdimensi tinggi. Representasi ini memungkinkan mesin untuk menangkap hubungan semantik antar kata berdasarkan

penggunaannya dalam konteks. Pada bagian ini, kita memperkenalkan dua konsep dasar: *hipotesis distribusi* dan *embedding kata*, yang bersama-sama membentuk dasar untuk representasi terdistribusi modern dalam NLP.

- **Memahami Semantik dalam NLP:** Untuk membangun model NLP yang efektif, penting untuk memahami semantik teks – bagaimana makna disampaikan melalui kata-kata dan struktur. Alur kerja analisis semantik yang umum meliputi:
 1. Membagi teks menjadi unit leksikal (misalnya, kata atau token)
 2. Mengidentifikasi makna setiap unit
 3. Menganalisis struktur sintaksis
 4. Menentukan makna kontekstual

Langkah-langkah ini secara kolektif memungkinkan mesin untuk mengekstrak properti linguistik dan membangun representasi semantik.

Ada dua pendekatan populer untuk representasi teks, yaitu pendekatan vektorisasi dasar dan representasi terdistribusi.

- **Pendekatan Vektorisasi Dasar:** Model ruang vektor adalah model aljabar yang merepresentasikan unit teks – seperti kata, frasa, atau dokumen – sebagai vektor dalam ruang multidimensi. Vektor-vektor ini mengkodekan pengidentifikasi seperti frekuensi istilah atau nilai indeks dari kosakata korpus.

Teknik-teknik utama meliputi:

- **Matriks Dokumen-Istilah:** Dokumen direpresentasikan sebagai baris dan istilah sebagai kolom; nilai setiap sel menunjukkan frekuensi atau keberadaan istilah.
- **Term Frequency-Inverse Document Frequency (TF-IDF):** Istilah penting diberi bobot berdasarkan frekuensinya dalam sebuah dokumen relatif terhadap frekuensinya di seluruh korpus.
- **Kesamaan Kosinus:** Mengukur kesamaan antara dua vektor dokumen dengan menghitung kosinus sudut di antara keduanya. Nilainya berkisar dari -1 (tidak mirip) hingga 1 (mirip).

Diberikan dua vektor dokumen $A = [a_1, a_2, a_3, \dots, a_n]$ dan $B = [b_1, b_2, b_3, \dots, b_n]$ dengan masing-masing n atribut, Kesamaan Kosinus, $\cos(\theta)$, direpresentasikan sebagai:

$$\cos\theta = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n a_i b_i}{\sqrt{\sum_{i=1}^n a_i^2} \sqrt{\sum_{i=1}^n b_i^2}}$$

di mana a_i dan b_i adalah komponen dari vektor A dan B masing-masing dan $A \cdot B$ adalah hasil perkalian titik (produk dalam) dari kedua vektor tersebut. Meskipun efektif, pendekatan vektorisasi dasar memiliki keterbatasan:

- **Dimensi Tinggi:** Vektor jarang dengan banyak entri nol menyebabkan inefisiensi memori dan komputasi.

- **Masalah Kata di Luar Kosakata (OOV):** Kata-kata yang tidak dikenal yang tidak ada dalam kosakata tidak dapat direpresentasikan secara bermakna.
- **Representasi Terdistribusi:** Untuk mengatasi keterbatasan model ruang vektor, NLP telah beralih ke representasi terdistribusi, yang menggunakan vektor padat berdimensi rendah untuk merepresentasikan kata dan frasa. Vektor-vektor ini dipelajari dari data dan menangkap hubungan semantik berdasarkan pola kemunculan bersama kata. Sebelum membahas representasi terdistribusi, kita harus membahas beberapa konsep umum:

- **Hipotesis Distribusional dan Kesamaan:** Hipotesis distribusional menyatakan bahwa kata-kata yang muncul dalam konteks serupa cenderung mempunyai makna serupa. Misalnya, kata bahasa Inggris *bank* memiliki arti berbeda dalam kalimat “at the *bank* of the river” (di tepi sungai) dan “the *bank* is located at the corner of this road” (bank berada di sudut jalan). Sebaliknya, kata *eagle* dan *airplane* dapat muncul dalam konteks serupa, misalnya terbang, sehingga representasi vektornya cenderung berdekatan dalam ruang semantik.
- **Penyematan dan Semantik Vektor:** Penyematan mengacu pada pemetaan kata-kata ke dalam ruang vektor kontinu. Vektor-vektor ini mengkodekan hubungan semantik, memungkinkan mesin untuk bernalar tentang makna. Kata-kata dengan makna yang serupa diposisikan lebih dekat bersama, memungkinkan aplikasi seperti pencarian semantik, sistem rekomendasi, dan AI percakapan.

Teknik penyematan populer meliputi:

- **Word2Vec:** Mempelajari penyematan tetap untuk setiap kata berdasarkan konteksnya.
- **GloVe:** Menggabungkan statistik kemunculan bersama kata secara global dengan jendela konteks lokal.

Model-model ini dilatih pada korpus besar dan menghasilkan embedding yang dapat digeneralisasi dengan baik di berbagai tugas NLP.

Representasi distribusi dipelajari menggunakan jaringan saraf dan biasanya berasal dari matriks kemunculan bersama, yang menangkap seberapa sering kata-kata muncul bersama dalam sebuah korpus. Vektor-vektor ini dapat mewakili tidak hanya kata-kata tetapi juga frasa dan seluruh dokumen. Pada Bab 12, Word2Vec dan GloVe dibahas secara lebih rinci, termasuk varian arsitektur Continuous Bag-of-Words dan Skip-Gram serta proses pelatihannya untuk menghasilkan penyematan kata berkualitas tinggi.

11.10 RINGKASAN

Dengan pertumbuhan eksponensial data teks dari media sosial, situs web, dan komunikasi digital, NLP telah menjadi alat penting untuk mengekstrak wawasan dan mengotomatiskan tugas-tugas seperti analisis teks, terjemahan, dan interaksi percakapan. Bab ini meletakkan konsep-konsep dasar yang diperlukan untuk memahami sistem NLP klasik dan modern. Kami memulai dengan memperkenalkan *fase-fase NLP*, yang meliputi analisis leksikal, sintaksis, semantik, wacana, dan pragmatik – masing-masing berkontribusi pada kemampuan mesin untuk memahami dan menghasilkan bahasa manusia. Kemudian, kami mengeksplorasi

ilmu linguistik, menyoroti perannya dalam membentuk NLP melalui studi fonetik, morfologi, sintaksis, semantik, dan pragmatik.

Selanjutnya, kami meneliti *pendekatan pembelajaran* untuk tugas-tugas NLP, membandingkan sistem berbasis aturan dengan metode pembelajaran mesin dan pembelajaran mendalam. Kami membahas bagaimana pendekatan hibrida dapat menggabungkan kekuatan kedua paradigma untuk meningkatkan kinerja dan kemampuan adaptasi. Kemudian, kami meninjau *teknik dan tugas inti NLP*, seperti tokenisasi, penandaan POS, pengenalan entitas bernama, analisis sentimen, dan terjemahan mesin. Teknik-teknik ini membentuk blok bangunan aplikasi NLP di dunia nyata. Bab ini juga membandingkan *NLP klasik* – yang bergantung pada aturan buatan tangan dan model statistik – dengan *NLP pembelajaran mendalam*, yang menggunakan jaringan saraf untuk mempelajari fitur langsung dari data. Sementara metode klasik menawarkan interpretasi dan efisiensi, model pembelajaran mendalam unggul dalam menangani bahasa yang kompleks dan kaya konteks.

Terakhir, kami memperkenalkan *model bahasa*, termasuk *model N-gram*, dan membahas bagaimana model tersebut memperkirakan probabilitas urutan kata. Kami menyimpulkan dengan tinjauan umum tentang *semantik vektor dan embedding*, yang merepresentasikan kata-kata dalam ruang berdimensi tinggi dan menangkap hubungan semantik berdasarkan konteks. Konsep-konsep tersebut menjadi landasan bagi bab berikutnya, yang membahas *Word2Vec*, *GloVe*, dan *arsitektur Transformer* hingga perkembangan *Model Bahasa Besar*.

11.11 PERTANYAAN

1. Jelaskan lima fase Pemrosesan Bahasa Alami (NLP): Leksikal, Sintaksis, Semantik, Integrasi Wacana, dan Analisis Pragmatik. Jelaskan bagaimana setiap fase berkontribusi pada pemrosesan bahasa manusia.
2. Bandingkan dan kontraskan pendekatan berbasis aturan dan pembelajaran mesin terhadap NLP. Diskusikan kelebihan dan kekurangan masing-masing pendekatan dan identifikasi skenario di mana salah satunya mungkin lebih efektif daripada yang lain.
3. Jelaskan perbedaan antara NLP Klasik dan NLP Pembelajaran Mendalam. Berikan contoh teknik atau model dari setiap kategori untuk mendukung penjelasan Anda.
4. Manakah dari berikut ini yang paling tepat menggambarkan Pengenalan Entitas Bernama (NER)?
 - a. Menetapkan kategori tata bahasa pada kata-kata
 - b. Mengidentifikasi dan mengklasifikasikan kata benda khusus seperti nama, organisasi, dan lokasi
 - c. Menentukan sentimen suatu teks
 - d. Menerjemahkan teks dari satu bahasa ke bahasa lain
5. Dalam konteks model bahasa, perplexitas didefinisikan sebagai
 - a. Jumlah kata dalam kosakata
 - b. Probabilitas N-gram yang tidak terlihat diatur ke nol
 - c. Ukuran seberapa baik model memprediksi sampel

- d. Rasio kata-kata penghenti (stopwords) terhadap kata-kata yang bermakna
- 6. [Latihan Pemrograman – Dasar] Kembangkan fungsi Python yang memecah paragraf yang diberikan menjadi kata-kata, menghapus kata-kata penghenti menggunakan Natural Language Toolkit (NLTK), dan mengembalikan token yang telah dibersihkan.
- 7. [Latihan Pemrograman – Lanjutan] Implementasikan model bahasa bigram di Python menggunakan Estimasi Kemungkinan Maksimum. Latih model pada korpus sampel dan hitung probabilitas kalimat uji yang diberikan. Gabungkan perataan Laplace untuk menangani bigram yang tidak terlihat.

BAB 12

ARSITEKTUR TRANSFORMER DAN EVOLUSI LLM

12.1 PENDAHULUAN

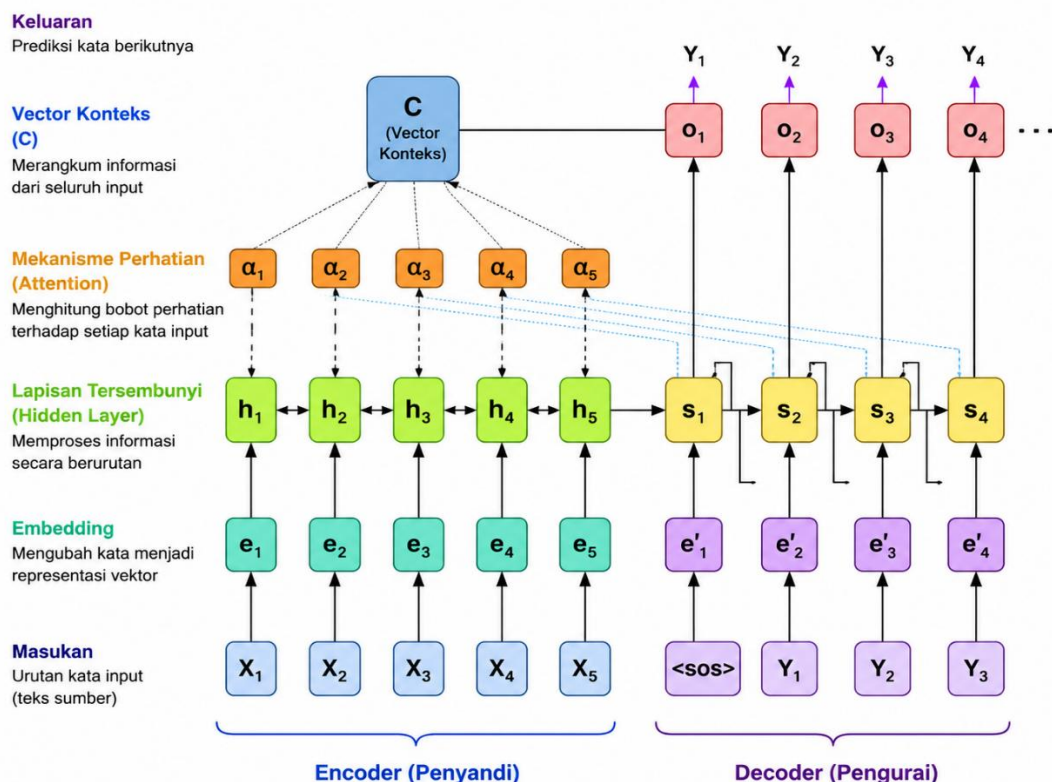
Pada bab sebelumnya, kita telah membahas Terjemahan Mesin Statistik (SMT), sebuah pendekatan dasar untuk terjemahan bahasa otomatis. Bab ini mengalihkan fokus ke Terjemahan Mesin Neural (NMT), yang merupakan lompatan besar ke depan di bidang ini. Sistem SMT beroperasi dengan memanfaatkan model statistik yang dibangun dari korpus bilingual besar. Model-model ini memperkirakan probabilitas terjemahan dengan menguraikan proses menjadi komponen-komponen terpisah – masing-masing dioptimalkan secara independen. SMT mengidentifikasi urutan bahasa target yang paling mungkin untuk kalimat sumber tertentu menggunakan tiga komponen kunci: model terjemahan, yang memetakan frasa antar bahasa; model bahasa, yang memastikan kelancaran dalam bahasa target; dan dekoder, yang memilih terjemahan terbaik. Proses ini melibatkan segmentasi teks sumber, mencocokkan segmen dengan terjemahan yang secara statistik mungkin, dan menyusunnya menjadi keluaran yang koheren. Landasan matematika SMT berpusat pada penghitungan probabilitas urutan target berdasarkan susunan kata dalam teks sumber. Pelatihan melibatkan penyelarasan korpus paralel untuk mengekstrak unit terjemahan dan membangun model statistik, sementara aplikasi menggunakan model ini untuk menghasilkan terjemahan melalui algoritma pencarian. Alih-alih mengandalkan aturan linguistik eksplisit, SMT mempelajari pola dari kumpulan data besar.

Sebaliknya, NMT menggunakan jaringan saraf untuk mempelajari seluruh proses terjemahan dari ujung ke ujung, menghilangkan kebutuhan akan segmentasi modular. NMT menggunakan arsitektur encoder-decoder yang ditingkatkan oleh mekanisme perhatian. Encoder mengubah teks sumber menjadi vektor numerik yang menangkap makna dan konteks, sementara decoder menghasilkan terjemahan kata demi kata. Mekanisme perhatian memungkinkan model untuk fokus secara dinamis pada bagian input yang relevan selama setiap langkah decoding. Gambar 12.1 mengilustrasikan arsitektur NMT. Proses dimulai dengan token sumber yang memasuki lapisan embedding, yang mengubahnya menjadi vektor padat. Vektor-vektor ini kemudian diproses melalui lapisan tersembunyi dari encoder, menghasilkan representasi yang diperkaya secara kontekstual. Mekanisme perhatian, yang ditunjukkan oleh garis putus-putus, membangun koneksi fleksibel antara encoder dan decoder, memungkinkan model untuk secara selektif fokus pada komponen input tertentu selama proses penerjemahan.

Bobot perhatian mengontrol bagaimana setiap keadaan encoder berkontribusi pada vektor konteks, yang pada gilirannya memandu pembangkitan output decoder. Pengaturan yang rumit ini memungkinkan NMT untuk secara efektif mengelola struktur linguistik yang kompleks dan ketergantungan jarak jauh, tantangan yang sering kali sulit diatasi oleh metode statistik tradisional. Sistem NMT bergantung pada dua tipe arsitektur utama: model

Transformer dan model *Jaringan Saraf Berulang*. Dalam bab ini, kita akan memfokuskan diskusi kita hanya pada model *Transformer*. Model *Transformer* membawa kemajuan signifikan pada penerjemahan dengan memproses semua elemen input sekaligus, menggunakan mekanisme perhatian diri untuk menganalisis hubungan antara setiap kata dalam urutan. Pemrosesan paralel ini meningkatkan kecepatan dan efisiensi penerjemahan jika dibandingkan dengan model penerjemahan sebelumnya. Yang penting, *Transformer* juga mampu melakukan penerjemahan "zero-shot", yang berarti mereka dapat menerjemahkan antara pasangan bahasa yang tidak termasuk dalam set pelatihan dengan memanfaatkan pola dari kombinasi bahasa lain.

Bab ini melanjutkan landasan NLP dari bab sebelumnya dengan memperkenalkan inovasi utama yang mengubah bidang tersebut, yaitu penyematan kata, arsitektur *Transformer*, dan kemunculan Model Bahasa Besar (LLM). Pembahasan dimulai dengan *Word2Vec* dan *Global Vectors for Word Representation (GloVe)*, kemudian beralih ke model *Transformer*. Selanjutnya, evolusi LLM ditelusuri hingga *ChatGPT-4*, disertai pembahasan mengenai peluang dan tantangannya.



Gambar 12.1 Arsitektur NMT yang menyoroti Encoder dan Decoder.

12.2 WORD2VEC

Word2Vec adalah teknik berbasis jaringan saraf untuk mempelajari representasi vektor kata dari korpus teks berukuran besar. Teknik ini mengubah cara mesin memahami hubungan antarkata dengan memetakan kata ke ruang vektor kontinu, sehingga kata yang serupa secara

semantik atau sintaksis ditempatkan berdekatan. Tidak seperti metode tradisional yang memperlakukan kata sebagai simbol diskrit, Word2Vec menghasilkan vektor numerik padat yang mempertahankan hubungan kontekstual. Penyematan ini memungkinkan operasi aritmatika – seperti penjumlahan dan pengurangan vektor – yang mencerminkan analogi linguistik (misalnya, *raja – pria + wanita ≈ ratu*). Word2Vec memperkenalkan dua arsitektur jaringan saraf, yaitu *Continuous Bag-of-Words* (CBOW) dan *Skip-Gram*. Keduanya merupakan jaringan saraf dangkal yang dirancang untuk mempelajari penyematan kata secara efisien, tetapi berbeda dalam cara memproses kata konteks dan kata target selama pelatihan. Pembahasan berikut dimulai dengan CBOW.

Continuous bag-of-words (cbow)

CBOW memprediksi kata target berdasarkan kata-kata konteks di sekitarnya dalam jendela berukuran tetap. Model ini mengambil beberapa kata konteks sebagai input dan mencoba memprediksi kata di tengah jendela. Salah satu perubahan kunci dalam CBOW, dibandingkan dengan model bahasa jaringan saraf tradisional, adalah penghapusan lapisan tersembunyi nonlinier dan pengenalan lapisan proyeksi bersama yang digunakan di semua kata. Pengaturan ini disebut sebagai model bag-of-words (BoW) karena posisi setiap kata konteks tidak memengaruhi cara model memproses input.

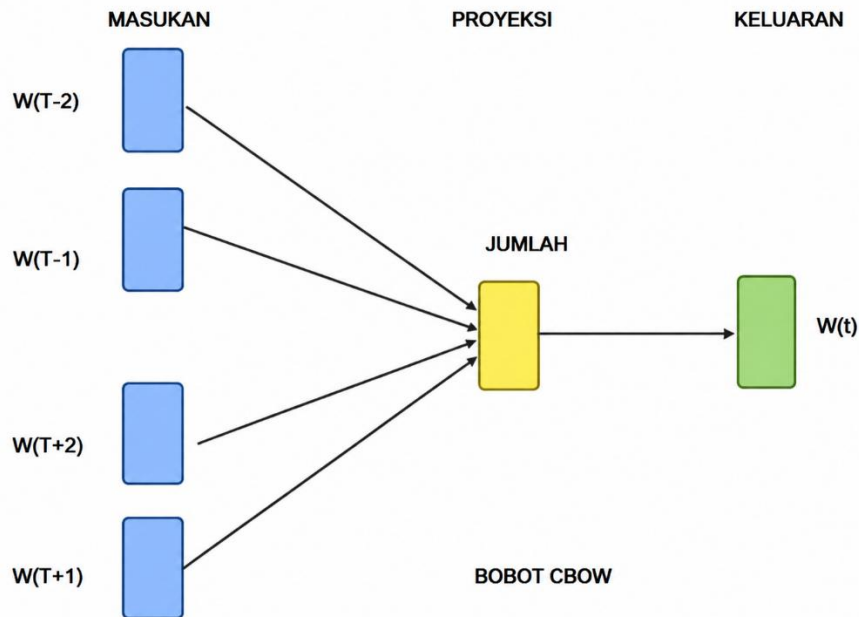
Dalam CBOW, tujuan pelatihan adalah untuk memaksimalkan probabilitas memprediksi kata target berdasarkan konteks di sekitarnya. Model ini mengambil beberapa kata konteks, biasanya empat kata sebelum dan empat kata setelah kata target, dan memproyeksikannya ke dalam ruang embedding bersama. Vektor kata ini kemudian dirata-ratakan untuk membentuk representasi konteks tunggal, yang berfungsi sebagai input untuk memprediksi kata pusat. Dengan mempelajari cara memprediksi kata target, atau kata pusat, dari konteksnya, CBOW menangkap pola bagaimana kata-kata digunakan bersama dalam bahasa alami. Kita dapat melihat ini dalam arsitektur (Gambar 12.2), di mana beberapa kata konteks diproyeksikan ke dalam ruang bersama dan digabungkan untuk memprediksi kata pusat. Kompleksitas pelatihan model CBOW diberikan dalam sebagai:

$$Q = N * D + D * \log_2(V) \quad (12.1)$$

Di sini, N mewakili jumlah kata konteks, D adalah dimensi embedding, dan V adalah ukuran kosakata. Ekspresi $N \times D$ memperhitungkan proyeksi kata konteks input ke dalam ruang embedding bersama. Istilah kedua, yaitu D dikalikan dengan logaritma basis-2 dari ukuran kosakata (V), sesuai dengan biaya komputasi untuk memprediksi kata target menggunakan SoftMax hierarkis. Pada dasarnya, persamaan ini merangkum operasi-operasi kunci yang terlibat selama pelatihan. Ini mencerminkan efisiensi arsitektur CBOW.

Contoh CBOW dalam lingkungan akademis dapat berupa latihan mengisi titik-titik kosong. Dalam kalimat, “mahasiswa biasa melakukan eksperimen sains,” model mengambil kata-kata di sekitarnya, “mahasiswa,” “menggunakan,” “untuk,” “melakukan,” “sains,” dan “eksperimen,” sebagai input untuk memprediksi kata yang hilang “peralatan.” CBOW bekerja dengan menggunakan kata-kata konteks di sekitar posisi target untuk memprediksi kata yang

termasuk dalam posisi tersebut. Seperti halnya mahasiswa yang mempelajari kosakata melalui paparan berbagai struktur kalimat, model ini mempelajari pola penggunaan kata dengan berulang kali mengamati bagaimana kata-kata muncul dalam konteks yang berbeda.



Gambar 12.2 Arsitektur CBOW.

CBOW berbeda secara signifikan dari model BoW tradisional baik dalam struktur maupun kemampuan. Sementara BoW merepresentasikan teks dengan menghitung frekuensi kata, ia tidak mempertahankan urutan kata atau informasi kontekstual, sehingga sulit untuk menangkap makna di balik penggunaan bahasa. Sebaliknya, CBOW memanfaatkan pendekatan berbasis jaringan saraf untuk memprediksi kata target dari konteks sekitarnya, memungkinkan model untuk mempelajari hubungan semantik dan sintaksis antar kata. Dengan memasukkan konteks ke dalam prediksinya, CBOW menawarkan pemahaman makna kata yang lebih bernuansa dibandingkan dengan representasi berbasis frekuensi yang lebih kaku yang digunakan dalam BoW.

Skip-gram

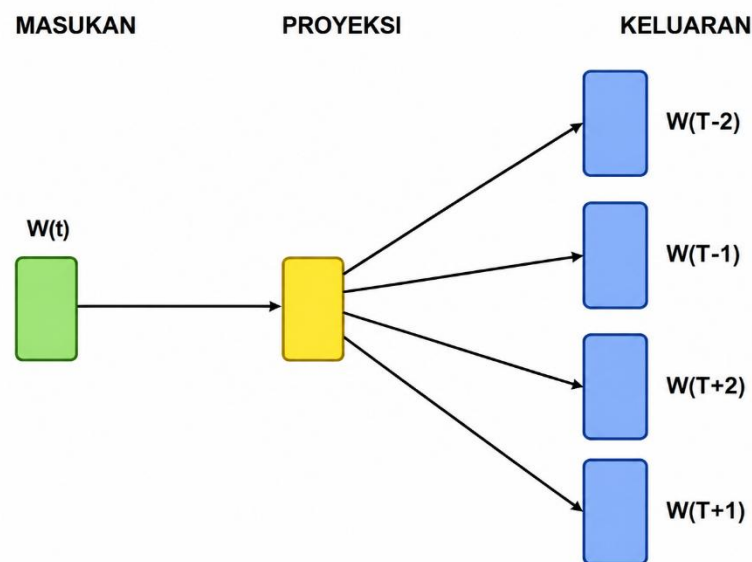
Arsitektur Skip-Gram berfungsi berlawanan dengan CBOW. Alih-alih menggunakan kata-kata konteks untuk memprediksi kata target, Skip-Gram menggunakan kata target (atau pusat) untuk memprediksi konteks di sekitarnya. Dalam pengaturan ini, setiap kata saat ini berfungsi sebagai input, dan model dilatih untuk memprediksi kata-kata yang berada dalam jendela yang ditentukan baik sebelum maupun sesudahnya. Hal ini memungkinkan model untuk mempelajari representasi kata yang dioptimalkan untuk menangkap hubungan kontekstual di berbagai posisi dalam kalimat. Skip-Gram menganalisis teks dengan membuat pasangan kata target dan konteks. Untuk setiap kata dalam korpus, model mengidentifikasi kata-kata di sekitarnya dalam jendela yang ditentukan untuk membentuk pasangan ini. Proses

ini diilustrasikan pada Gambar 12.3, di mana kata target digunakan untuk memprediksi kata-kata konteks di kedua sisi: Kompleksitas pelatihan model Skip-Gram diberikan sebagai:

$$Q = C * (D + D * \log_2(V)) \quad (12.2)$$

Di sini, C menunjukkan jarak maksimum kata konteks, D adalah dimensi embedding, dan V adalah ukuran kosakata. Peningkatan jendela konteks telah terbukti meningkatkan kualitas vektor kata yang dipelajari, karena mengekspos model ke pola linguistik yang lebih luas. Namun, ini juga meningkatkan biaya komputasi selama pelatihan. Untuk mengelola pertukaran ini, model memberikan bobot yang lebih rendah pada kata-kata yang jauh, secara efektif mengurangi pengaruhnya pada tujuan pelatihan sambil mempertahankan efisiensi komputasi.

Contoh Skip-Gram dapat berupa aplikasi pembelajaran bahasa. Model Skip-Gram dapat mendukung pemerolehan kosakata dengan memprediksi kata-kata yang umum muncul di dekat kata target tertentu. Misalnya, jika kata "kurikulum" digunakan sebagai input, model tersebut dapat mengidentifikasi istilah terkait seperti "sekolah," "pendidikan," dan "mata pelajaran." Hal ini memungkinkan aplikasi untuk menghasilkan contoh kalimat yang relevan atau menyarankan kosakata terkait, membantu mahasiswa memahami bagaimana kata tersebut digunakan dalam konteks.



Gambar 12.3 Arsitektur Skip-Gram.

Arsitektur CBOW dan Skip-Gram menawarkan keunggulan yang berbeda tergantung pada kasus penggunaannya. CBOW umumnya lebih cepat dilatih karena memprediksi satu kata target dari beberapa kata konteks, sehingga lebih efisien secara komputasi. Sebaliknya, Skip-Gram membutuhkan lebih banyak sumber daya tetapi cenderung berkinerja lebih baik pada tugas-tugas semantik dan lebih efektif dalam mempelajari representasi untuk kata-kata yang

jarang muncul. CBOW, di sisi lain, berkinerja baik pada tugas-tugas sintaksis dan lebih cocok untuk kata-kata yang sering muncul. Saat bekerja dengan kumpulan data yang lebih kecil, CBOW sering menghasilkan hasil yang lebih baik dengan menghaluskan distribusi penggunaan kata, sedangkan Skip-Gram mempertahankan informasi yang lebih detail dan biasanya berkinerja lebih baik pada kumpulan data yang lebih besar.

Word2Vec mewakili kemajuan besar dalam pembelajaran representasi kata dan telah memainkan peran kunci dalam membentuk tugas-tugas NLP modern. Dengan menangkap hubungan semantik dan sintaksis antar kata, ia memungkinkan perhitungan kesamaan yang lebih akurat dan telah menghasilkan peningkatan kinerja dalam berbagai tugas NLP, termasuk analisis sentimen, terjemahan mesin, dan menjawab pertanyaan. Tidak seperti metode sebelumnya, Word2Vec menawarkan akurasi yang lebih tinggi sambil mempertahankan biaya komputasi yang lebih rendah, sehingga efisien untuk dilatih pada kumpulan data skala besar. Kemampuannya untuk mempertahankan hubungan kata yang bermakna dalam bentuk vektor padat juga menjadikannya komponen dasar dalam banyak sistem NLP kontemporer. Varian Skip-Gram dan CBOW menggunakan masukan yang sama, yaitu kalimat yang dipecah menjadi token kata, dan menghasilkan keluaran yang sama berupa dua matriks penyematan yang barisnya merepresentasikan vektor kata hasil pembelajaran. Namun, karena arah masalah prediksinya berlawanan, aliran informasi dan karakteristik statistik tujuan pelatihannya berbeda.

Memahami kapan harus menggunakan setiap varian Word2Vec sangatlah penting. Pemilihan varian Word2Vec perlu mempertimbangkan ukuran korpus, distribusi kosakata, dan tugas hilir. CBOW dapat dilatih sekitar dua kali lebih cepat dan berkinerja baik ketika data yang memadai memungkinkan perataan konteks. Skip-Gram lebih lambat, tetapi menghasilkan representasi yang lebih kaya untuk kata langka dan sering dipilih pada bahasa yang kompleks secara morfologis atau domain khusus dengan terminologi berekor panjang. Kedua arsitektur sangat efisien karena tidak memiliki lapisan tersembunyi nonlinier, sehingga dapat dilatih pada miliaran token dalam waktu relatif singkat. Efisiensi tersebut menjadikan Word2Vec standar *de facto* dalam semantik distribusional sebelum munculnya penyematan kontekstual. Melalui pemodelan probabilistik sederhana, CBOW dan Skip-Gram mendorong perkembangan semantik ruang vektor, penalaran analogi, dan inisialisasi jaringan dalam, serta tetap relevan meskipun model kontekstual telah berkembang.

12.3 REPRESENTASI KATA VEKTOR GLOBAL

Global Vectors for Word Representation (GloVe) adalah teknik penyematan kata yang banyak digunakan dan berbeda secara fundamental dari model prediktif seperti mempelajari penyematan dengan memprediksi kata berdasarkan konteks lokal, GloVe adalah model berbasis hitungan yang memanfaatkan statistik kemunculan bersama kata global untuk mempelajari hubungan semantik. Tidak seperti pengkodean one-hot, yang menetapkan setiap kata vektor biner unik dan mengabaikan kesamaan semantik, GloVe menempatkan kata-kata yang serupa lebih dekat satu sama lain dalam ruang multi-dimensi yang kontinu. Mari kita pertimbangkan contoh sederhana untuk mengilustrasikan penempatan kata oleh GloVe.

Bayangkan grafik dua dimensi di mana sumbu horizontal diberi label "Dimensi 1 (semantik)" dan sumbu vertikal diberi label "Dimensi 2 (semantik)." Setiap kata diplot sebagai titik pada grafik ini. Di sini, kita akan menentukan lokasi kata-kata yang berbeda pada grafik ini. Di bagian kiri atas grafik, mari kita bayangkan kata-kata berikut membentuk sebuah kelompok: *royalty* (sekitar $x = 3.5$, $y = 4.7$), *queen* (sekitar $x = 4.0$, $y = 4.0$), dan *king* (sekitar $x = 3.0$, $y = 3.7$). Sedikit di bawah dan di sebelah kirinya, muncul kelompok lain dengan kata-kata berikut: *man* (sekitar $x = 1.5$, $y = 3.0$) dan *woman* (sekitar $x = 2.5$, $y = 2.8$). Di bagian kanan bawah grafik, kelompok ketiga yang berbeda mencakup kata-kata berikut: *puppy* (sekitar $x = 6.5$, $y = 3.0$), *dog* (sekitar $x = 5.5$, $y = 2.5$), dan *cat* (sekitar $x = 6.5$, $y = 1.5$). Contoh ini akan dengan jelas menunjukkan bahwa kata-kata yang berhubungan secara semantik terletak berdekatan satu sama lain pada grafik sehingga membentuk kelompok-kelompok yang berbeda berdasarkan kesamaan konseptualnya. Kata-kata yang berhubungan secara semantik berkelompok di sepanjang dimensi konseptual, seperti *jenis kelamin* atau *spesies*. Dalam contoh ini, kami secara visual menunjukkan bagaimana vektor numerik mengkodekan makna melalui hubungan spasial. Misalnya, *raja* dan *ratu* akan memiliki vektor yang serupa, perbedaannya terutama terletak pada dimensi *jenis kelamin*. Dengan cara yang sama, *anjing* dan *anak anjing* akan berdekatan karena sering muncul dalam konteks yang serupa.

Jadi, bagaimana cara kerja GloVe? GloVe dimulai dengan membangun matriks kemunculan bersama kata, yang mencatat seberapa sering setiap kata muncul di dekat setiap kata lain dalam korpus. Setiap sel dalam matriks mencerminkan frekuensi pasangan kata. Sel yang lebih terang menunjukkan frekuensi kemunculan bersama yang lebih tinggi. Menggunakan teknik faktorisasi matriks, GloVe menguraikan matriks ini untuk mengungkap pola semantik laten. Vektor kata yang dihasilkan mengkodekan konteks lokal (kata-kata terdekat) dan konteks global (pola kemunculan bersama secara keseluruhan di seluruh korpus). Pendekatan konteks ganda ini memungkinkan GloVe untuk menangkap hubungan yang bernuansa antar kata. Misalnya, perbedaan antara *raja* dan *ratu* mirip dengan perbedaan antara *pria* dan *wanita*, yang mencerminkan dimensi *gender* yang sama. Mirip dengan contoh sebelumnya, mari kita pertimbangkan contoh lain di sini untuk mengilustrasikan potensi GloVe dalam menangkap hubungan vektor paralel antar kata.

Pertimbangkan grafik dua dimensi dengan dimensi semantik horizontal dan vertikal seperti yang telah dibahas sebelumnya. Mari kita bayangkan sebuah titik yang mewakili kata *Raja* yang terletak di area kiri atas grafik (misalnya, sekitar $x = 0.7$, $y = 4.5$). Selain itu, mari kita bayangkan titik lain yang mewakili *Pria* yang terletak lebih jauh ke bawah dan sedikit ke kanan dari "Raja" (misalnya, sekitar $x = 1.0$, $y = 2.0$). Sebuah panah konseptual atau segmen garis dapat ditarik dari "Pria" ke "Raja," yang mewakili vektor untuk "Raja - Pria." Dengan cara yang serupa, pasangan kata terkait lainnya dapat diplot di sebelah kanan titik-titik ini sebagai berikut: *Ratu* terletak di area kanan atas (misalnya, sekitar $x = 3.5$, $y = 4.0$) dan *Wanita* terletak lebih jauh ke bawah dan sedikit ke kanan dari "Ratu" (misalnya, sekitar $x = 3.0$, $y = 2.0$). Sebuah panah konseptual atau segmen garis dapat ditarik dari "Wanita" ke "Ratu," mewakili vektor untuk "Ratu - Wanita." Wawasan utama yang ditunjukkan di sini adalah bahwa

vektor yang menghubungkan "Pria" ke "Raja" hampir sejajar dengan, dan sering kali memiliki panjang yang sama dengan, vektor yang menghubungkan "Wanita" ke "Ratu." Paralelisme visual ini menggarisbawahi konsep bahwa transformasi semantik dari "pria" ke "raja" analog dengan transformasi dari "wanita" ke "ratu," yang menyiratkan pengkodean yang konsisten dari hubungan abstrak (seperti 'gender' dan 'kerajaan') dalam ruang vektor kata.

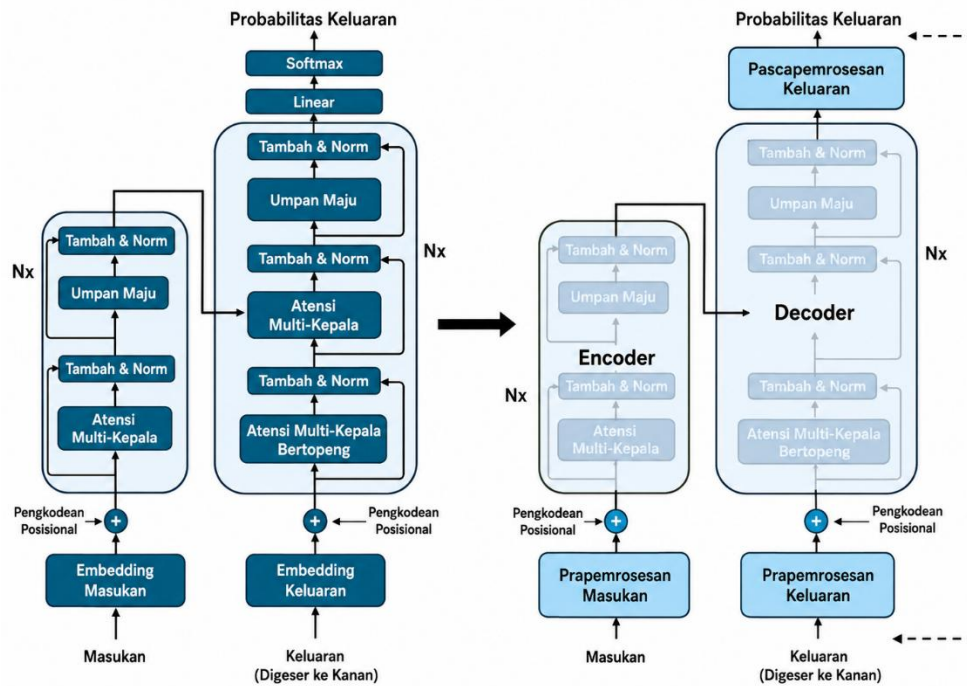
Hubungan paralel pada contoh tersebut menunjukkan bahwa selisih vektor dapat mengodekan relasi semantik. GloVe mengekstrak pola ini dengan memanfaatkan statistik kemunculan bersama kata secara global. Dalam analitik data, GloVe mengubah teks mentah yang tidak terstruktur menjadi vektor numerik terstruktur yang dapat dianalisis menggunakan metode matematika dan statistik. Penyematan kata tersebut mendukung pengelompokan istilah serupa, pengungkapan hubungan topik, dan peningkatan kinerja pembelajaran mesin. Dengan demikian, GloVe menjembatani bahasa manusia dan interpretasi mesin serta menjadi landasan bagi model kontekstual berikutnya, seperti BERT dan Generative Pre-trained Transformer (GPT). Setelah membahas Word2Vec dan GloVe, pembahasan beralih ke arsitektur yang mendefinisikan ulang NLP modern, yaitu Transformer. Diperkenalkan pada 2017, *Transformer* merevolusi pemodelan sekuens dengan menggantikan rekurensi menggunakan self-attention, sehingga memungkinkan pemrosesan paralel dan meningkatkan kinerja pada penerjemahan, pembangkitan teks, serta pemodelan bahasa berskala besar.

12.4 ARSITEKTUR TRANSFORMER

Model Transformer, yang diperkenalkan oleh Vaswani dan kawan-kawan pada tahun 2017, merevolusi NLP dengan mengganti rekurensi dengan mekanisme perhatian. Tidak seperti model sebelumnya yang memproses urutan langkah demi langkah, Transformer memungkinkan pemrosesan paralel data masukan, secara dramatis meningkatkan kinerja dalam penerjemahan, pembuatan teks, dan pemodelan bahasa skala besar. Arsitektur ini terdiri dari struktur encoder-decoder. Encoder memproses urutan masukan (misalnya, kalimat dalam bahasa sumber), sedangkan decoder menghasilkan urutan keluaran (misalnya, kalimat terjemahan). Gambar 12.4 mengilustrasikan struktur ini. Setiap lapisan encoder mencakup:

- *Mekanisme self-attention*, yang memungkinkan setiap kata untuk memperhatikan setiap kata lain dalam urutan input.
- *Jaringan saraf feed-forward*, yang diterapkan pada setiap posisi secara independen.
- *Penambahan & Normalisasi Lapisan*, yang menstabilkan pelatihan dan meningkatkan konvergensi.

Dekoder mencerminkan struktur ini tetapi menambahkan lapisan perhatian encoder-decoder, yang memungkinkannya untuk fokus pada bagian input yang relevan selama pembangkitan. Baik encoder maupun decoder dimulai dengan penyematan kata dan pengkodean posisi, yang menangkap makna semantik dan urutan kata. Dekoder kemudian menghasilkan distribusi probabilitas atas token kosakata, memungkinkan model untuk menghasilkan terjemahan satu token pada satu waktu.



Gambar 12.4 Arsitektur Transformer.

Mekanisme perhatian

Inovasi inti dari Transformer adalah mekanisme perhatian, yang memungkinkan model untuk menimbang pentingnya berbagai elemen input secara dinamis. Perhatian produk titik berskala dihitung sebagai:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (12.3)$$

Di mana:

- Q , K , dan V adalah *kueri*, *kunci*, dan *nilai* yang diperoleh dari embedding input.
- d_k adalah dimensi kunci.

Mekanisme perhatian memungkinkan jaringan saraf untuk fokus pada bagian input yang paling relevan dengan memberikan bobot kepentingan yang berbeda pada setiap elemen. Ini meniru bagaimana manusia secara selektif memprioritaskan informasi tertentu sambil menyaring informasi lainnya, tergantung pada tugasnya. Proses perhatian dimulai dengan mengubah elemen masukan menjadi representasi vektor atau penyematanan yang mengodekan fitur setiap elemen secara numerik. Fungsi kompatibilitas kemudian menghitung skor relevansi setiap elemen terhadap tugas. Skor tersebut dinormalisasi menjadi distribusi probabilitas yang disebut bobot perhatian. Selanjutnya, jumlah tertimbang dari penyematanan masukan dihitung menggunakan bobot tersebut untuk menghasilkan vektor konteks yang memuat informasi paling relevan. Selama pelatihan, model belajar memberikan bobot lebih tinggi kepada masukan yang paling berkontribusi terhadap prediksi yang akurat.

Dalam penerjemahan mesin, perhatian sangat berpengaruh dengan memungkinkan model untuk secara dinamis menggeser fokus di berbagai bagian kalimat sumber saat menghasilkan setiap kata dalam terjemahan. Model sequence-to-sequence tradisional mengompres input ke dalam vektor tunggal dengan panjang tetap, yang sering menyebabkan hilangnya informasi, terutama pada kalimat yang lebih panjang atau lebih kompleks. Perhatian mengatasi keterbatasan ini dengan memungkinkan model untuk mengakses bagian-bagian yang relevan dari urutan input pada setiap langkah decoding. Hal ini dimungkinkan melalui fungsi kompatibilitas dan mekanisme bobot perhatian yang sama seperti yang dijelaskan sebelumnya. Dengan belajar memprioritaskan bagian-bagian kalimat sumber yang paling informatif, model menghasilkan terjemahan yang lebih akurat dan sesuai konteks.

Model tidak perlu diberi tahu secara eksplisit elemen mana yang penting. Sebaliknya, model mempelajarinya selama pelatihan melalui backpropagation. Dengan mengoptimalkan kinerja pada tugas hilir, seperti analisis sentimen atau terjemahan mesin, model secara bertahap menyesuaikan parameter fungsi kompatibilitas untuk menekankan bagian input yang paling relevan. Misalnya, dalam tugas klasifikasi sentimen, model tidak memerlukan label untuk kata-kata yang mengandung sentimen. Model akan secara alami belajar untuk memberikan bobot perhatian yang lebih tinggi pada kata-kata seperti *fenomenal* atau *mengerikan* sementara memberikan bobot yang lebih rendah pada kata-kata netral seperti *hari ini* atau *kemarin*.

Jenis-jenis perhatian

Transformer mengimplementasikan beberapa jenis perhatian:

- **Perhatian Diri (Self-Attention):** Setiap kata memperhatikan semua kata lain dalam urutan yang sama, menangkap ketergantungan tanpa memandang jarak.
- **Perhatian Multi-Kepala (Multi-Head Attention):** Beberapa mekanisme perhatian berjalan secara paralel, masing-masing menangkap berbagai jenis hubungan. Outputnya digabungkan dan ditransformasikan secara linier untuk membentuk representasi yang lebih kaya.
- **Perhatian Hierarkis (Hierarchical Attention):** Digunakan untuk input dengan struktur berlapis seperti dokumen; mekanisme ini pertama-tama memperhatikan kata kunci dalam kalimat dan kemudian kalimat penting di seluruh dokumen.

Untuk memahami apa yang dianggap penting oleh model, alat visualisasi perhatian seperti peta panas (heatmap) dapat digunakan. Visualisasi ini menampilkan bobot perhatian sebagai gradien warna, membantu mengidentifikasi elemen yang paling dapat difokuskan oleh model. Hal ini dapat mengungkapkan wawasan berharga tentang perilaku model dan potensi bias. Misalnya, seperti yang ditunjukkan pada diagram sebelumnya, mekanisme perhatian menghitung bobot di seluruh urutan input untuk fokus pada elemen yang paling relevan. Alat visual serupa dapat membantu menafsirkan dan menyempurnakan kinerja model.

Jadi bagaimana kita meningkatkan terjemahan mesin? Untuk meningkatkan kualitas terjemahan mesin, salah satu langkah pertama adalah melampaui batasan vektor panjang tetap dalam model encoder-decoder tradisional. Alih-alih mengompres seluruh kalimat masukan ke dalam satu vektor, perhatian memungkinkan decoder untuk mengakses semua

keadaan tersembunyi encoder selama proses decoding. Mekanisme perhatian juga memungkinkan penyelarasan kata dinamis antara bahasa sumber dan bahasa target. Untuk setiap kata yang dihasilkan dalam keluaran, model menghitung bobot perhatian pada kalimat sumber, memungkinkannya untuk fokus pada kata-kata yang paling relevan pada saat itu. Ini sangat berguna untuk pasangan bahasa dengan struktur tata bahasa atau urutan kata yang berbeda. Misalnya, ketika menerjemahkan dari bahasa Inggris ke bahasa Korea, di mana kata sifat mendahului kata benda dalam bahasa Inggris tetapi mengikuti kata benda dalam bahasa Korea, mekanisme perhatian akan memungkinkan model untuk melihat seluruh kalimat bahasa Inggris dan fokus pada kata sifat yang relevan saat menghasilkan frasa kata benda Korea.

Perhatian juga membantu mengatasi ambiguitas kata yang memiliki beberapa arti dengan mempertimbangkan konteks sekitarnya. Sebagai contoh, kata bahasa Inggris *light* dapat merujuk pada penerangan, bobot yang ringan, atau warna yang terang. Model menggunakan kata-kata di sekitarnya untuk menentukan interpretasi yang sesuai. Setelah mengeksplorasi arsitektur Transformer dan dampaknya yang transformatif pada NLP, kini kita beralih ke evolusi LLM – kelas model yang dibangun di atas fondasi Transformer untuk mencapai kemampuan luar biasa dalam pemahaman, pembangkitan, dan interaksi bahasa. Bagian berikut menelusuri perkembangan historis LLM, menyoroti tonggak penting dan inovasi yang telah membentuk lanskap sistem bahasa bertenaga AI saat ini.

12.5 EVOLUSI LLM

LLM telah merevolusi berbagai domain dalam ML/AI, khususnya di bidang NLP dan interaksi manusia-komputer. Evolusi NLP dimulai pada awal tahun 1950-an dan 1960-an dengan sistem berbasis aturan, di mana programmer secara manual membuat aturan linguistik untuk tugas bahasa tertentu. Tahun 1980-an menandai transisi ke pendekatan statistik, yang menggabungkan teknik seperti pohon keputusan, model Markov tersembunyi, dan mesin vektor pendukung. Pada dekade 2000-an, jaringan saraf, khususnya Recurrent Neural Networks, semakin banyak digunakan karena kemampuannya memodelkan ketergantungan jarak jauh dalam data sekuensial. Sekitar 2014, berkembang pembelajaran sequence-to-sequence melalui arsitektur seperti LSTM dan gated recurrent unit. Mekanisme perhatian dan terutama arsitektur Transformer kemudian menjadi terobosan besar karena meningkatkan kinerja dan efisiensi komputasi. Inovasi tersebut membentuk paradigma prelatihan dan fine-tuning: model terlebih dahulu dilatih pada korpus teks berskala besar, kemudian disesuaikan untuk tugas hilir. Model penting seperti ELMo, GPT, dan BERT, diikuti GPT-3 dan GPT-4, selanjutnya menetapkan tolok ukur baru pada berbagai aplikasi NLP.

GPT adalah model autoregresif pertama berdasarkan arsitektur Transformer. Hal ini menandai tonggak penting dalam NLP dengan menunjukkan kemampuan pembangkitan bahasa yang mengesankan. Versi selanjutnya, GPT-2 dan GPT-3, lebih meningkatkan kinerja GPT dengan meningkatkan ukuran model dan pelatihan pada kumpulan data yang lebih besar. GPT-3 dilatih sebelumnya pada korpus umum dan menyoroti kemampuan pembangkitan bahasa yang belum pernah terjadi sebelumnya. Terakhir, GPT-4, versi keempat dari seri GPT,

menjadi model multimodal skala besar yang memperoleh kemampuan memproses input gambar dan teks untuk menghasilkan output teks.

Chatgpt dan kebangkitan ai percakapan

ChatGPT mencontohkan kekuatan dan kecanggihan LLM. Ini adalah turunan yang disempurnakan dari seri GPT, dioptimalkan untuk percakapan seperti manusia. Model ChatGPT dilatih untuk memprediksi kata berikutnya dalam sebuah kalimat, mempelajari sintaksis, semantik, dan pengetahuan dunia dalam prosesnya. Istilah *model bahasa besar* menjadi terkenal dengan GPT-2 dan GPT-3, yang berisi miliaran parameter dan dilatih pada kumpulan data yang sangat besar. Model-model ini dianggap "besar" bukan hanya karena jumlah parameternya tetapi juga karena skala dan keragaman data pelatihannya. ChatGPT dibangun di atas fondasi ini dengan melakukan penyempurnaan tambahan pada kumpulan data percakapan. Hal ini memungkinkannya untuk menghasilkan dialog yang lebih alami, koheren, dan sadar konteks. Perkembangan dari GPT ke ChatGPT merupakan tonggak penting dalam AI percakapan. Tergantung pada versinya, model GPT berkisar dari 117 juta parameter (GPT-2 kecil) hingga 175 miliar parameter (GPT-3), memungkinkan mereka untuk menangkap pola linguistik dan kontekstual yang rumit. ChatGPT-4, khususnya, mendukung input multimodal dan telah mendapatkan popularitas yang luas. Model bahasa besar, termasuk yang ada dalam seri GPT, telah mencapai kinerja terbaik di berbagai tugas linguistik dan penalaran. Mereka mampu menghasilkan teks yang koheren, menjawab pertanyaan, menyusun esai, meringkas dokumen, menerjemahkan bahasa, dan bahkan menghasilkan kode yang dapat dieksekusi. Melalui few-shot learning, model seperti GPT-3 dapat melakukan tugas baru hanya dengan contoh minimal, tanpa perlu pelatihan ulang eksplisit. Dilatih pada teks internet yang luas dan beragam, LLM juga memiliki pengetahuan faktual yang luas, memungkinkan mereka untuk memberikan wawasan tentang berbagai topik. Selain itu, model-model ini meningkatkan kreativitas manusia dengan membantu brainstorming, menulis, dan kreasi artistik. Seiring dengan terus meningkatnya daya komputasi dan ketersediaan data, kinerja dan fleksibilitasnya diharapkan akan semakin meningkat.

Terlepas dari keunggulannya, LLM juga menghadirkan tantangan yang signifikan. Mereka dapat secara tidak sengaja mereplikasi dan memperkuat bias yang ada dalam data pelatihan mereka dan dapat dieksploitasi untuk menghasilkan informasi yang salah, teks deepfake, atau spam otomatis. Kebutuhan komputasi mereka yang sangat besar menimbulkan kekhawatiran mengenai konsumsi energi dan dampak lingkungan. Lebih lanjut, karena model menjadi semakin kompleks, proses pengambilan keputusan internal mereka tetap tidak transparan, sehingga menyulitkan upaya untuk memastikan transparansi dan akuntabilitas. LLM juga dapat menghasilkan teks yang tampak kredibel namun mengandung ketidakakuratan faktual atau inkonsistensi logis, yang mengungkapkan keterbatasan berkelanjutan dalam kekokohan dan keandalan.

12.6 KESIMPULAN

Bab ini menguraikan perkembangan pemodelan bahasa dari penyematan kata awal hingga model berbasis Transformer skala besar modern. Dimulai dengan *Word2Vec*, di mana

arsitektur CBOW dan Skip-Gram mempelajari representasi terdistribusi yang menangkap hubungan semantik antar kata. *GloVe* memperluas ide-ide ini dengan menggabungkan konteks lokal dengan statistik ko-okurensi global untuk menghasilkan vektor kata yang dapat diinterpretasikan. Bab ini kemudian memperkenalkan *arsitektur Transformer*, yang menggantikan rekurensi dengan perhatian diri, memungkinkan pemrosesan paralel yang efisien dan penanganan ketergantungan jarak jauh yang lebih baik. Inovasi ini membuka jalan bagi LLM seperti ChatGPT-4, yang memanfaatkan kumpulan data besar, tumpukan Transformer yang dalam, dan pembelajaran penguatan dari umpan balik manusia untuk mencapai penalaran tingkat lanjut, pemahaman kontekstual, dan generasi dialog alami.

Singkatnya, bab ini mengkaji evolusi, arsitektur, dan kemampuan LLM dengan penekanan pada seri GPT sebagai studi kasus. Model-model tersebut telah mendefinisikan ulang lanskap AI dan NLP serta mendorong kemajuan besar dalam pemahaman dan pembangkitan bahasa. Namun, pengembangannya perlu memperhatikan aspek etika, lingkungan, dan interpretabilitas agar manfaat teknologi dapat diwujudkan secara bertanggung jawab dan adil.

12.7 PERTANYAAN

1. Bandingkan dan kontraskan Penerjemahan Mesin Statistik (SMT) dan Penerjemahan Mesin Neural (NMT) dalam hal arsitektur dan proses pembelajarannya. Jelaskan bagaimana pengenalan mekanisme perhatian telah berdampak pada kualitas terjemahan.
2. Jelaskan dua arsitektur Word2Vec – Continuous Bag of Words (CBOW) dan Skip-Gram – termasuk tujuan pelatihannya, karakteristik komputasi, dan kasus penggunaan tipikalnya.
3. Jelaskan tujuan optimasi GloVe (Global Vectors for Word Representation) dan bagaimana pemanfaatan statistik ko-okurensi globalnya menghasilkan struktur analogis (misalnya, raja – pria + wanita $\approx$ ratu).
4. Uraikan struktur encoder-decoder dari model Transformer dan jelaskan peran self-attention, multi-head attention, dan embedding posisi/kata.
5. Ringkas evolusi Model Bahasa Besar (LLM) dari ELMo/BERT/GPT ke GPT-4/ChatGPT. Diskusikan dua kekuatan dan dua tantangan dari model-model ini seperti yang disorot dalam bab ini.
6. [Latihan Pemrograman – Dasar] Implementasikan perhatian dot-product berskala dengan kueri Q , kunci K , nilai V (untuk satu head). Tunjukkan bahwa penskalaan dengan menstabilkan softmax.
7. [Latihan Pemrograman – Lanjutan] Kembangkan loop pelatihan Skip-gram minimal dengan pengambilan sampel negatif. Buat pasangan (pusat, konteks) dalam jendela yang ditentukan dan latih embedding yang membedakan pasangan sebenarnya dari contoh negatif yang diambil sampelnya.
8. Dalam arsitektur Transformer, sublayer perhatian tambahan mana yang ada di decoder tetapi tidak ada di encoder?

- A. Perhatian diri (Self-attention)
 - B. Perhatian (silang) encoder-decoder
 - C. Perhatian diri multi-kepala dengan masking
 - D. Jaringan feed-forward
9. **T:** Bagian “skala” dari perhatian produk titik skala menskalakan QK^T dengan:
- A. $\sqrt{d_v}$
 - B. d_k
 - C. $\sqrt{d_k}$
 - D. $\log d_k$
10. Dibandingkan dengan Word2Vec, GloVe paling tepat dikarakterisasi sebagai
- A. Model konteks lokal yang murni prediktif
 - B. Model berbasis hitungan yang memanfaatkan statistik kemunculan bersama kata global
 - C. Model konvolusional tingkat karakter
 - D. Model urutan berulang

BAB 13

Merancang Sistem AI Terdistribusi

13.1 POLA DESAIN DAN BAHASA POLA

Kemajuan teknologi Big Data telah memungkinkan ilmuwan dan insinyur data mengekstrak nilai dari data terstruktur maupun tidak terstruktur dalam volume besar dengan investasi infrastruktur yang relatif minimal. Sistem yang lazim disebut *pipeline Big Data* (diperkenalkan pada Bab 3) menyerap data mentah dari berbagai sumber, memprosesnya melalui serangkaian tahapan, dan mengubahnya menjadi wawasan yang dapat ditindaklanjuti. Pipeline Big Data biasanya terdiri dari layanan yang saling terhubung yang memungkinkan pemrosesan data yang terukur, efisien, dan tangguh. Organisasi semakin bergantung pada sistem ini untuk mengintegrasikan data dari berbagai sumber dan memperoleh kecerdasan bisnis untuk keunggulan kompetitif. Tergantung pada kasus penggunaannya, sistem Big Data dapat berkisar dari ekstraksi dan pemuatan data sederhana hingga alur kerja kompleks yang melibatkan pelatihan dan inferensi model pembelajaran mesin (ML).

Pipeline yang umum diawali dengan pengumpulan dan prapemrosesan data, kemudian penyimpanan pada repositori seperti basis data relasional, gudang data, danau data, atau data mart. Algoritma ekstraksi fitur mengidentifikasi atribut relevan untuk membangun kumpulan data pelatihan. Kumpulan data tersebut dimasukkan ke algoritma ML yang mempelajari pola dan motif, sedangkan model terlatih selanjutnya diterapkan pada instans tanpa label untuk melakukan klasifikasi atau mengaitkannya dengan pola yang telah dipelajari.

Terlepas dari ketersediaan alat dan teknologi, perancangan sistem Big Data menghadirkan beberapa tantangan:

- **Ketidaksesuaian antara Tujuan Bisnis dan Implementasi Teknis:** Kapabilitas AI dan ML sering gagal mengintegrasikan beragam teknologi yang dibutuhkan organisasi.
- **Masalah Integrasi:** Konektivitas ujung-ke-ujung yang buruk antara komponen sistem dapat menyebabkan penundaan data, kesalahan format, duplikasi, dan penurunan kualitas.
- **Kekakuan Arsitektur:** Alur kerja Extract, Transform, Load (ETL) tradisional cenderung tidak fleksibel dan mahal. Pola berbasis cloud, seperti ingestion, modeling, enrichment, transformation, dan delivery, menawarkan fleksibilitas dan pemisahan komponen yang lebih baik.
- **Kurangnya Kejelasan Implementasi:** Kerangka arsitektur yang ada (misalnya, Lambda, Kappa, Real-time) tidak secara memadai mengatasi masalah seperti kualitas data, skalabilitas, keamanan, dan keandalan.

Untuk mengatasi tantangan ini, bab ini mengeksplorasi bagaimana *pola desain* dapat berfungsi sebagai blok bangunan dasar untuk merancang sistem Big Data. Dengan menggunakan pipeline Big Data OCI sebagai referensi, kami menyajikan pendekatan pemetaan multi-langkah:

1. Memetakan persyaratan fungsional (F) dan nonfungsional (NF) OCI ke pola desain yang relevan.
2. Mengaitkan pola-pola ini dengan lapisan spesifik arsitektur OCI.
3. Mengidentifikasi atribut kualitas yang ditargetkan oleh setiap pola.

Untuk mengatasi tantangan arsitektur sistem AI terdistribusi, bab ini dibangun di atas konsep dasar yang diperkenalkan dalam Bab 3 dengan meninjau kembali *pola desain* dan *bahasa pola*. Konsep-konsep ini menawarkan pendekatan terstruktur dan dapat digunakan kembali untuk memecahkan masalah berulang dalam desain sistem. Dengan menerapkannya pada pipeline Big Data – khususnya dalam Oracle Cloud Infrastructure (OCI) – kita dapat secara sistematis meningkatkan atribut kualitas sistem seperti skalabilitas, keandalan, dan pemeliharaan.

Pola desain dan bahasa pola menyediakan kerangka kerja formal untuk merancang sistem Big Data. Pola desain adalah solusi yang dapat digunakan kembali untuk masalah yang sering terjadi dalam konteks tertentu. Alih-alih mewakili implementasi lengkap, pola desain berfungsi sebagai templat yang dapat diadaptasi untuk memenuhi persyaratan sistem yang berkembang. Pola-pola ini mempercepat pengembangan dengan menawarkan paradigma yang terbukti dan teruji yang mengurangi kemungkinan kesalahan desain dan utang teknis. Ketika sistem dibangun dari awal tanpa panduan, masalah-masalah halus dapat muncul di akhir siklus pengembangan – masalah yang sulit dideteksi dan mahal untuk diselesaikan. Dengan mengadopsi pola desain, arsitek dapat mengantisipasi dan mengurangi masalah tersebut sejak dini, meningkatkan kekokohan dan kemampuan penggunaan kembali sistem. Bahasa pola merupakan kumpulan terstruktur dari pola desain yang saling terkait dan dapat disusun untuk memecahkan masalah arsitektural yang kompleks. Bahasa tersebut mengungkapkan solusi yang berasal dari pengalaman praktis serta membantu pihak lain memahami dan menerapkannya secara efektif.

Misalnya, Yokoyama mengusulkan pola desain ML yang disebut *Membedakan Logika Bisnis dari Model ML*, yang meningkatkan stabilitas operasional dengan memisahkan logika bisnis dari komponen ML. Pola ini menggantikan arsitektur tiga lapis tradisional dengan struktur yang tersegmentasi secara horizontal:

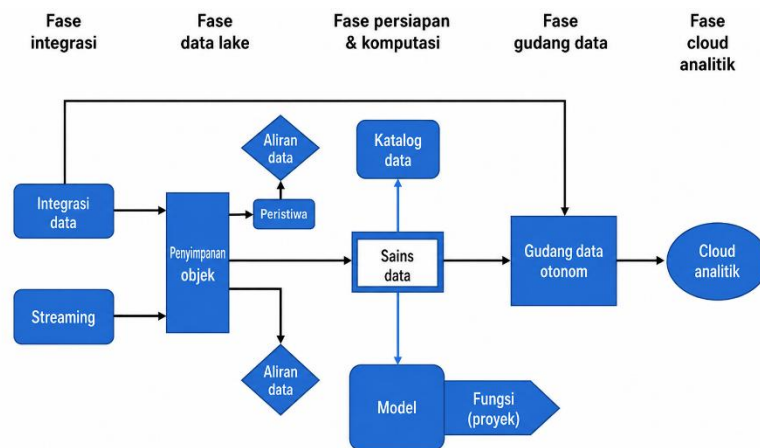
- **Lapisan Presentasi:** Mengelola antarmuka pengguna dan pengumpulan data.
- **Lapisan Logika:** Mengimplementasikan logika bisnis, mesin inferensi, dan modul prapemrosesan.
- **Lapisan Data:** Menangani penyimpanan dalam basis data, data mart, dan data lake.

Secara vertikal, pola ini memisahkan logika bisnis dari hal-hal yang berkaitan dengan ML, mengurangi utang teknis dan meningkatkan pemeliharaan. Pola ini juga membedakan antara pengumpulan data dan penyimpanan data, yang selanjutnya meningkatkan modularitas. Memahami bahwa *pemisahan logika bisnis dari model ML* itu sendiri merupakan pola desain sangatlah penting. Sekarang kita menerapkan konsep-konsep ini pada studi kasus dunia nyata: *pipeline Big Data OCI*. Contoh ini mengilustrasikan bagaimana pola desain dapat dipetakan ke komponen sistem dan atribut kualitas untuk memandu pengembangan sistem AI yang terukur, tangguh, dan mudah dipelihara.

13.2 PIPELINE BIG DATA OCI

Pipeline Big Data OCI adalah arsitektur tanpa server yang dirancang untuk mendukung penyerapan, pemrosesan, dan analitik data skala besar. Seperti yang ditunjukkan pada Gambar 13.1, pipeline terdiri dari lima fase utama: *penyerapan*, *penyimpanan* (Fase Data Lake), *ETL* (Fase Persiapan dan Komputasi), *pergudangan* (Fase Data Warehouse), dan *presentasi* (Fase Analitik Cloud). Dalam arsitektur tanpa server, pengembang fokus pada penulisan dan penyebaran kode sementara penyedia cloud mengelola penyediaan infrastruktur, penskalaan, dan pemeliharaan. Model ini memungkinkan penanganan data semi-terstruktur dan tidak terstruktur secara efisien dari berbagai sumber. Alur kerja dimulai dengan *pengumpulan data*, di mana data mentah dikumpulkan dan dimuat ke dalam repositori terpusat – biasanya berupa *data lake*. Selanjutnya, *fase ETL* mengekstrak, mentransformasi, dan memuat data baik dalam mode batch maupun stream. Outputnya adalah kumpulan data yang telah dibersihkan dan sesuai standar, siap untuk diproses lebih lanjut.

Data yang telah diproses disimpan dalam format terstruktur di dalam *Autonomous Data Warehouse*, layanan terkelola yang memastikan ketersediaan tinggi, toleransi kesalahan, skalabilitas, dan keamanan. Pada *fase presentasi*, data divisualisasikan dan dianalisis menggunakan alat intelijen bisnis yang mendukung dasbor, pemfilteran, dan wawasan grafis.



Gambar 13.1 Alur kerja big data di Oracle Cloud Infrastructure.

- **Komponen Inti Pipeline OCI:** Pipeline Big Data OCI terdiri atas 12 komponen utama yang masing-masing memiliki peran berbeda:
 1. **Layanan Streaming OCI:** Menerima dan mengonsumsi aliran data bervolume tinggi secara waktu nyata.
 2. **Integrasi Data OCI:** Layanan ETL berbasis cloud untuk transformasi data yang cepat.
 3. **Penyimpanan Objek OCI:** Menyimpan data tidak terstruktur tanpa batas dengan daya tahan tinggi.
 4. **Layanan Peristiwa OCI:** Mengotomatiskan respons terhadap perubahan status sumber daya.
 5. **Alur Data OCI:** Platform Spark tanpa server untuk pemrosesan data yang terukur.

6. **Layanan Big Data OCI:** Menyediakan Hadoop kelas perusahaan untuk berbagai beban kerja.
 7. **Ilmu Data OCI:** Memungkinkan pembuatan, pelatihan, dan manajemen model.
 8. **Layanan Model OCI:** Mendukung pelatihan model ML dari data yang masuk.
 9. **Gudang Data Otonom OCI:** Menawarkan penskalaan elastis dan kinerja kueri yang cepat.
 10. **Katalog Data OCI:** Mengelola metadata untuk tata kelola data yang lebih baik.
 11. **OCI Analytics Cloud:** Memfasilitasi analitik kolaboratif di seluruh perusahaan.
 12. **Layanan Fungsi OCI:** Mengeksekusi logika bisnis berbasis peristiwa tanpa manajemen infrastruktur.
- **Arsitektur Berlapis:** Pipeline OCI diorganisasikan ke dalam beberapa lapisan dengan tanggung jawab khusus:
 1. **Lapisan Ingesti:** Terhubung ke sumber data internal dan eksternal melalui berbagai protokol.
 2. **Lapisan Penyimpanan:** Menyimpan data dalam zona mentah, bersih, dan terkurasi:
 - Zona Mentah: Penyimpanan sementara data yang diingesti.
 - Zona Bersih: Penyimpanan permanen setelah pemeriksaan kualitas.
 - Zona Terkurasi: Dioptimalkan untuk konsumsi, dipartisi, dan dikatalogkan.
 3. **Lapisan Pengatalogan dan Pencarian:** Mengelola metadata dan mendukung pelacakan dan pencarian versi.
 4. **Lapisan Pemrosesan:** Mengubah data melalui validasi, normalisasi, dan pengayaan.
 5. **Lapisan Konsumsi:** Menyediakan alat untuk analitik, pelaporan, dan ML.
 6. **Lapisan Keamanan dan Tata Kelola:** Memastikan perlindungan data, kontrol akses, enkripsi, dan audit.
 - **Memetakan Persyaratan ke Pola Desain:** OCI memiliki Gudang Data Otonom yang menyediakan basis data yang mudah digunakan, sepenuhnya otonom, yang dapat diskalakan secara elastis, memberikan kinerja kueri yang cepat, dan tidak memerlukan administrasi basis data. Kerangka kerja Big Data dengan arsitektur tanpa server seperti OCI tampaknya merupakan solusi yang sempurna untuk komputasi terdistribusi skala besar. Arsitektur OCI menghadapi tantangan *funksional* (F) dan *nonfungsional* (NF), seperti keterbatasan sumber daya, tanpa status, latensi, dan kesalahan integrasi. Hal ini dirinci dalam Tabel 13.1.

TABEL 13.1 Masalah Fungsional (F) dan Non-Fungsional (NF) dalam Pipeline Big Data OCI

ID	Deskripsi
F1	Keterbatasan Sumber Daya: OCI memiliki keterbatasan terkait waktu eksekusi dan ukuran memori. Batas ukuran memori membuat pemisahan awal data masukan menjadi lebih rumit karena kerangka kerja harus memastikan bahwa ukuran partisi masukan tidak melebihi batas memori ini.

F2	Tanpa Status: OCI bersifat tanpa status yang berarti bahwa informasi tentang status eksekusinya perlu disimpan dalam penyimpanan eksternal.
F3	Tidak ada komunikasi antar pemanggilan: Di OCI, pada tahap pengacakan (shuffle), data dipindahkan dari node pada satu tahap ke node di tahap berikutnya. Karena tidak ada komunikasi yang diizinkan di antara dua pemanggilan Lambda, tidak ada cara untuk mentransfer data antar node. Oleh karena itu, penyimpanan akses cepat yang menyimpan semua data sementara sangat diperlukan.
F4	Waktu Mulai Dingin: OCI memiliki waktu mulai dingin di mana sistem membuat wadah (container) dan memuat fungsi yang harus dieksekusi termasuk dependensinya. Hal ini mengakibatkan penurunan kinerja yang serius.
F5	Latensi: Di OCI, sebagian besar komunikasi antar-komponennya terjadi melalui API HTTP, yang bisa lebih lambat daripada sarana transportasi lainnya. Semakin besar jumlah komponen yang berkomunikasi melalui saluran yang tidak dioptimalkan, semakin besar pula latensi yang akan melekat pada aplikasi Serverless.
F6	Pengujian Lokal: Kesulitan dalam pengujian lokal aplikasi serverless dapat diklasifikasikan ke dalam dua cara. Pertama, karena sebagian besar infrastruktur diabstraksikan di dalam platform, bisa menjadi hal yang sulit untuk menghubungkan komponen aplikasi dengan cara yang realistis. Kedua, aplikasi Serverless secara inheren terdistribusi, dan terdiri dari banyak bagian terpisah, sehingga sekadar mengelola segudang fungsi dan komponen BaaS saja sudah menantang, bahkan secara lokal.
F7	Kehilangan Kendali: Masalah kehilangan kendali dapat dibagi menjadi empat aspek, yaitu Konfigurasi, Kinerja, Penyelesaian masalah, dan Keamanan.
NF11	Eksekusi tugas gagal ketika tabel sumber memiliki nama kolom dengan karakter khusus. Ketika kolom tabel sumber memiliki karakter khusus pada namanya, dan ada operator ekspresi di hilir (downstream) yang melakukan transformasi pada salah satu kolom tabel yang masuk, eksekusi tugas gagal.
NF12	Tugas SQL dalam pipeline gagal ketika nilai parameter yang masuk menggunakan keluaran SYS.RESPONSE_PAYLOAD_JSON.
NF13	Pengecualian waktu proses (Runtime exception) dengan ekspresi yang mereferensikan atribut data DATE.
NF14	Data pratinjau dari sumber Excel Object Storage tidak menampilkan metadata file.
NF15	Pipeline dengan badan permintaan yang diparameterisasi dalam tugas REST menjadi rusak.
NF16	Proses menjalankan pipeline gagal karena keluaran JSON dari tugas REST tidak dapat diekstraksi dan dikonsumsi di hilir.
NF17	Evaluasi kondisi sukses mengalami kegagalan saat menjalankan tugas REST.
NF21	File yang diperlukan untuk setiap aplikasi harus berada di wilayah (region) yang sama tempat aplikasi tersebut dibuat. Aplikasi harus dibuat di wilayah yang sama dengan bucket Object Store yang berisi semua file, jar, dan konfigurasi terkait yang diperlukan agar aplikasi berhasil dijalankan. Skenario lintas wilayah (cross region) tidak didukung.
NF22	Kesalahan Spark UI: Sebuah kesalahan mungkin ditemui saat mengakses Spark UI. Penyebab umum dari kesalahan ini adalah aplikasi Spark akan segera dihentikan.

Untuk mengatasi tantangan ini, serangkaian pola desain dipilih dan dipetakan ke persyaratan spesifik. Pemetaan ini memastikan bahwa setiap masalah arsitektur ditangani oleh solusi yang terbukti, meningkatkan kekokohan dan kualitas sistem. Untuk mengatasi tantangan ini, 24 pola desain yang dijelaskan dalam Tabel 13.2, dipertimbangkan ketika mendesain berbagai komponen di OCl. Tabel 13.3 menunjukkan atribut kualitas target yang diminati, seperti kinerja, keandalan, skalabilitas, dan pemeliharaan, yang ditangani oleh pola-pola ini.

TABEL 13.2 Deskripsi Pola Desain

Pola Desain	Komponen	Peran
Multisource extractor	Integrasi Data	Pendekatan untuk memasukkan berbagai jenis data dari beragam sumber data secara efisien. Pola ini menjamin ketersediaan tinggi dan distribusi.
Workload balancing	Sistem Big Data	Pola ini membantu menangani tantangan beban kerja data yang berkaitan dengan berbagai domain dan kasus bisnis secara sangat efisien.
Leader Election		Pola ini mengoordinasikan tindakan yang dilakukan oleh sekumpulan instans tugas yang berkolaborasi dalam aplikasi terdistribusi dengan memilih satu instans sebagai pemimpin yang bertanggung jawab mengelola instans lainnya.
Just-in-time	Penyimpanan Objek	Pola transformasi ini digunakan ketika data perlu dimuat terlebih dahulu ke penyimpanan data sebelum transformasi dan prapemrosesan dilakukan. Pola ini berjalan secara independen dengan membersihkan, memvalidasi, mengorelasikan, mentransformasikan, dan menyimpan data hasil ke penyimpanan data.
Façade Pattern		Pola ini menyediakan cara bagi alat intelijen bisnis, seperti Tableau dan Qlik, untuk mengakses data (mentah, diproses, atau disimpulkan).
Saga distributed transaction		Pola saga digunakan untuk transaksi yang melibatkan beberapa langkah, seperti layanan reservasi perjalanan.
Observer or Publish/Subscribe		Pola ini mendefinisikan ketergantungan antarobjek, sehingga perubahan keadaan pada satu objek menyebabkan seluruh objek yang bergantung padanya diberi tahu dan diperbarui secara otomatis.
Real time streaming	Streaming	Pola ini memfasilitasi pemrosesan data tidak terstruktur secara kontinu dan waktu nyata. Pola ini dapat meminimalkan latensi, membantu membangun

		sistem yang skalabel, memfasilitasi penguraian informasi waktu nyata, dan sebagainya.
Near-real time	Integrasi Data, Model	Pola ini mencakup penerapan solusi untuk akses data mendekati waktu nyata.
Sharding	Integrasi Data	Pola sharding membagi data dalam basis data menjadi segmen-segmen untuk mempercepat pelaksanaan perintah atau kueri.
Flyweight	Penyimpanan Objek	Pola ini menggunakan penggunaan bersama untuk mendukung sejumlah besar objek serupa secara efisien.
Active object		Pola ini memisahkan eksekusi metode dari pemanggilan metode yang berada dalam utas kontrolnya sendiri. Tujuannya adalah menghadirkan konkurensi dengan menggunakan pemanggilan metode asinkron dan penjadwal untuk menangani permintaan.
Command query responsibility segregation (CQRS)		Pola ini memisahkan operasi pembacaan data dari operasi pembaruan data dengan menggunakan antarmuka yang berbeda. Pola Command Query Responsibility Segregation (CQRS) menangani situasi ketika kueri basis data terjadi lebih sering daripada perubahan data.
Multi-destination	Integrasi Data	Pola ini memfasilitasi pemasukan data mentah (setelah pengumpulan data) ke HDFS dan penyimpanan data tradisional atau platform analitik lainnya. Pola data ini sangat skalabel, fleksibel, cepat, tahan terhadap kegagalan data, dan hemat biaya.
Join	Model	Pola Join menyediakan cara untuk menulis program konkuren, paralel, dan terdistribusi melalui pengiriman pesan. Dibandingkan penggunaan utas dan kunci, pola ini merupakan model pemrograman tingkat tinggi yang menggunakan konstruksi komunikasi untuk mengabstraksikan kompleksitas lingkungan konkuren dan memungkinkan skalabilitas.
Messaging		Pola ini memungkinkan pertukaran informasi (yaitu pesan) antara komponen dan aplikasi.
Reactor	Peristiwa	Objek reactor menyediakan antarmuka asinkron ke sumber daya yang harus ditangani secara sinkron.

Abstract factory		Pola ini menyediakan antarmuka untuk membuat keluarga objek yang saling berkaitan atau saling bergantung tanpa menentukan kelas konkretnya.
Pipes and Filters		Pola ini memecah tugas yang melakukan pemrosesan kompleks menjadi serangkaian elemen terpisah yang dapat digunakan kembali.
Throttling	Aliran Data	Pola throttling (atau pembatasan laju) mengontrol kecepatan aliran data menuju target. Pola ini sering digunakan untuk mencegah kegagalan saat serangan penolakan layanan terdistribusi atau untuk mengelola biaya infrastruktur cloud.
Lightweight stateless	Integrasi Data	Pola ini menyediakan akses data melalui layanan web dan tidak bergantung pada implementasi platform atau bahasa.
Workflow pipeline		Pola desain ini mengisolasi dan mengontainerisasi setiap langkah dalam alur kerja ML menjadi alur kerja terorganisasi untuk memastikan kemudahan pemeliharaan dan skalabilitas.
Mediator		Pendekatan mediator menyediakan abstraksi bagi data masuk dari berbagai sistem. Pola desain ini menyediakan cara yang efisien untuk memasukkan beragam data tidak terstruktur dari berbagai sumber data dan protokol yang berbeda.

TABEL 13.3 Daftar Semua Pola Desain dalam Arsitektur Tanpa Server OCI dan Atribut Kualitas yang Ditargetkannya

Pola Desain	Atribut Kualitas yang Ditargetkan
Multisource extractor	Kinerja, Keandalan
Workload balancing patterns	Kinerja, Keandalan, Kemudahan Pemeliharaan
Leader Election	Kinerja, Keandalan
Just-in-time design pattern	Kinerja
Saga distributed transactions pattern	Kinerja, Keandalan, Kemudahan Pemeliharaan
Lightweight stateless pattern	Kemudahan Pemeliharaan
Façade Pattern	Kemudahan Pemeliharaan
Real time streaming pattern	Kinerja
Near-real time pattern	Kinerja
Reproducibility design pattern	Keandalan, Kesesuaian Fungsional
Sharding pattern	Kesesuaian Fungsional, Skalabilitas
Flyweight	Kinerja, Skalabilitas

Active object	Kinerja
Command query responsibility segregation (CQRS)	Kinerja, Skalabilitas, Keamanan
Multi-destination design pattern	Kinerja, Keandalan
Join	Kinerja, Skalabilitas
Messaging design pattern	Skalabilitas, Keandalan, Kinerja, Kemudahan Pengujian, Keamanan
Reactor	Kinerja, Skalabilitas
Abstract factory	Kemudahan Pemeliharaan
Pipes and Filters	Kinerja, Kemampuan Digunakan Kembali, Skalabilitas
Throttling (rate-limiting) pattern	Kinerja, Keandalan
Workflow pipeline design pattern	Kesesuaian Fungsional, Skalabilitas, Kinerja
Mediator	Kemudahan Pemeliharaan
Observer or Publish/Subscribe	Skalabilitas, Keandalan, Kinerja, Kemudahan Pengujian, Keamanan

Tabel 13.4 menghubungkan tantangan OCI spesifik dengan pola desain yang mengatasinya.

Misalnya:

- *Multisource Extractor* mengatasi NF11–NF17 dan NF21–NF22, yang mencakup kesalahan pemformatan data, pengecualian runtime, dan masalah metadata.
- Pola *Workload Balancing*, *Leader Election*, dan *Just-in-Time* mengurangi F7 (hilangnya kendali).
- Pola *Lightweight Stateless* dan *Sharding* meningkatkan pemeliharaan dan skalabilitas dengan memisahkan komponen dan mengoptimalkan akses data.
- *Pemisahan tanggung jawab kueri perintah (CQRS)* mengatasi F3 (kurangnya komunikasi antar pemanggilan).
- *Throttling* membantu mengelola F1 (keterbatasan sumber daya).
- *Workflow Pipeline* mengatasi F4 (penundaan cold start).
- *Abstract Factory* mendukung F6 (tantangan pengujian lokal).

TABEL 13.4 Pemetaan Persyaratan Sistem Big Data OCI ke Pola Desain

Pola Desain	Persyaratan
Multisource extractor	NF11, NF12, NF13, NF14, NF15, NF16, NF17, NF21, NF22
Workload balancing patterns	F7
Leader Election	F7
Just-in-time design pattern	F7
Lightweight stateless pattern	F2, F6, NF11, NF12, NF13, NF14, NF15, NF16, NF17
Near-real time pattern	F5

Sharding pattern	NF11, NF12, NF13, NF14, NF15, NF16, NF17
Flyweight	NF21, NF22
Multi-destination design pattern	NF11, NF12, NF13, NF14, NF15, NF16, NF17
Abstract factory	F6
Throttling (rate-limiting) pattern	F1
Workflow pipeline design pattern	F4
Mediator	NF11, NF12, NF13, NF14, NF15, NF16, NF17
Command query responsibility segregation (CQRS)	F3

Tabel 13.5 mengkategorikan pola desain di seluruh *lapisan horizontal* – Presentasi, Logika, dan Data – dan domain vertikal – khusus ML dan khusus Bisnis. Klasifikasi ini membantu arsitek memilih pola berdasarkan konteks struktural dan fungsional sistem.

TABEL 13.5 Daftar Semua Pola Desain dalam Arsitektur Tanpa Server OCI pada Dimensi Horizontal dan Vertikal

Lapisan	Pola Desain
Presentasi	Multisource extractor, Observer or Publish/Subscribe
Logika	Workload balancing patterns, Leader Election, Just-in-time design pattern, Saga distributed transactions pattern, Observer or Publish/Subscribe
Data	Lightweight stateless pattern, Façade Pattern, Observer or Publish/Subscribe, Mediator, Command query responsibility segregation (CQRS)
Khusus ML	Real time streaming pattern, Near-real time pattern, Reproducibility design pattern, Sharding pattern, Lightweight stateless pattern, Workflow pipeline design pattern, Mediator, Command query responsibility segregation (CQRS)
Khusus Bisnis	Flyweight, Active object, multi-destination design pattern, Join Messaging design pattern, Messaging design pattern, Reactor, Abstract factory, Pipes and Filters, Throttling (rate-limiting) pattern, Lightweight stateless pattern, Workflow pipeline design pattern, Mediator, Command query responsibility segregation (CQRS)

Sebagai contoh:

- **Lapisan Presentasi:** Multisource Extractor, Observer.
- **Lapisan Logika:** Workload Balancing, Leader Election, Saga.
- **Lapisan Data:** Façade, CQRS, Mediator.
- **Khusus ML:** Real-Time Streaming, Sharding, Workflow Pipeline.
- **Khusus Bisnis:** Flyweight, Join, Messaging, Reactor.

Setelah memetakan pola desain ke persyaratan fungsional dan nonfungsional pipeline Big Data OCI, serta mengategorikannya berdasarkan lapisan dan domain arsitektural, kita sekarang menutup pembahasan dengan refleksi atas implikasi yang lebih luas dari penggunaan pola desain dalam merancang sistem AI terdistribusi. Bagian penutup ini menyoroti manfaat desain berbasis pola dan penerapannya pada penerapan skala kecil maupun tingkat perusahaan.

13.3 KESIMPULAN

Bab ini menyajikan 24 pola desain ML dan perangkat lunak dengan cakupan dan skala yang beragam. Melalui penerapannya dalam pipeline data dunia nyata, yaitu OCI, kami menunjukkan bagaimana pola-pola ini dapat memandu arsitektur sistem AI terdistribusi. Studi kasus tersebut menunjukkan bagaimana pola desain mendorong ketahanan terhadap perubahan dan penggunaan kembali, sehingga sistem dapat berkembang tanpa menumpuk utang teknis. Meskipun contoh pipeline OCI relatif sederhana, pola yang sama juga dapat diterapkan pada implementasi ML dan AI berskala besar. Penggunaannya dapat secara signifikan meningkatkan ketangguhan sistem, toleransi kesalahan, dan kemudahan pemeliharaan.

Dengan memanfaatkan pola desain, insinyur dapat secara sistematis menangani tantangan arsitektural dan meningkatkan atribut kualitas utama seperti kinerja, skalabilitas, keandalan, dan keamanan. Pola-pola ini menawarkan pendekatan terstruktur untuk membangun sistem AI terdistribusi yang tidak hanya kuat secara teknis, tetapi juga selaras dengan tujuan bisnis dan batasan operasional. Singkatnya, pola desain berfungsi sebagai perangkat dasar untuk merekayasa sistem AI yang dapat diskalakan, tangguh, dan adaptif. Penerapannya secara cermat dapat mengubah arsitektur kompleks menjadi solusi yang mudah dipelihara, berkinerja tinggi, dan siap diterapkan di dunia nyata.

13.4 PERTANYAAN

1. Definisikan pola desain dalam konteks sistem AI terdistribusi. Jelaskan bagaimana pola-pola tersebut meningkatkan penggunaan kembali, kemudahan pemeliharaan, dan skalabilitas ketika merancang sistem Big Data.
2. Diskusikan tantangan fungsional (F) dan nonfungsional (NF) yang dihadapi dalam pipeline Big Data OCI. Berikan dua contoh dari setiap jenis dan analisis implikasinya.
3. Jelaskan konsep bahasa pola. Terangkan bagaimana penggabungan beberapa pola desain menjadi sebuah bahasa dapat menangani tantangan kompleks dalam sistem AI terdistribusi.
4. Manakah dari berikut ini yang paling tepat menggambarkan pola Lightweight Stateless?
 - a. Menyediakan metode untuk menangani transaksi konkuren menggunakan thread terdistribusi
 - b. Memisahkan logika bisnis dari model pembelajaran mesin dengan memisahkan lapisan

- c. Menawarkan akses data yang independen dari platform melalui layanan web tanpa menyimpan state
 - d. Mengontrol laju aliran data ke dalam sistem untuk mencegah kelebihan beban
5. Dalam pipeline Big Data OCI, pola desain manakah yang secara khusus menargetkan kinerja, keandalan, dan kemudahan pemeliharaan dengan mendistribusikan beban kerja data secara efisien?
- a. Flyweight
 - b. Workflow Pipeline
 - c. Workload Balancing
 - d. Sharding

BAB 14

PENGAMANAN SISTEM AI

14.1 ANCAMAN DAN PERLINDUNGAN AI

Pengamanan sistem AI merupakan tugas yang kompleks dan kritis. Hal ini membutuhkan pengamanan data sensitif, menjaga integritas dan keandalan model, memastikan ketersediaan layanan, dan mencegah manipulasi aset AI yang berbahaya. Kegagalan dalam menerapkan langkah-langkah keamanan yang tepat dapat menyebabkan pelanggaran data, kerusakan model, keluaran yang bias atau tidak akurat, dan gangguan layanan – terutama di domain berisiko tinggi seperti perawatan kesehatan dan keuangan. Tujuan utama keamanan AI adalah untuk melindungi aset informasi. Sistem AI sering menangani data pribadi, keuangan, dan medis, sehingga menjadikannya target yang menarik untuk serangan musuh dan peracunan data. Model yang dikompromikan dapat menghasilkan hasil yang berbahaya atau menyesatkan, merusak kepercayaan dan keandalan. Pelanggaran keamanan juga dapat mengganggu layanan penting, sehingga langkah-langkah perlindungan yang kuat sangat penting untuk kelangsungan operasional.

Sistem AI sangat rentan terhadap serangan canggih seperti injeksi cepat dan inversi model, yang dapat memanipulasi perilaku atau mengekstrak logika kepemilikan. Menerapkan pertahanan yang tepat sangat penting untuk mencegah ancaman tersebut dan memastikan pengembangan dan penerapan AI yang bertanggung jawab. Langkah-langkah keamanan juga membantu mencegah penyebaran informasi yang salah, konten yang bias, dan pelanggaran privasi, sekaligus mendukung kepatuhan terhadap peraturan perlindungan data. Pada bagian selanjutnya, kami mendefinisikan *konsep keamanan inti* dan membangun *model ancaman* yang disesuaikan dengan AI. Kemudian kami memposisikan *kelas serangan* umum terhadap siklus hidup sistem AI dan merangkum *mekanisme pertahanan* yang, jika digabungkan, memungkinkan pertahanan berlapis.

Untuk mengamankan sistem AI secara efektif, penting untuk terlebih dahulu memahami konsep dasar *ancaman*, *kerentanan*, *serangan*, dan *mekanisme pertahanan*. *Ancaman* mengacu pada setiap potensi peristiwa – baik disengaja maupun tidak disengaja – yang dapat menyebabkan kerugian pada sistem AI atau membahayakan data yang diprosesnya. *Kerentanan* adalah kelemahan dalam desain, implementasi, konfigurasi, atau operasi sistem yang dapat dieksploitasi oleh ancaman. *Serangan* adalah eksploitasi sebenarnya dari kerentanan oleh pelaku ancaman, yang mungkin berupa manusia atau otomatis. Contohnya termasuk pencurian kredensial melalui phishing, peracunan data selama pelatihan, manipulasi input oleh pihak lawan, dan ekstraksi model atau data. Sebaliknya, *pertahanan* mencakup strategi, proses, dan teknologi yang digunakan untuk mengidentifikasi, mencegah, mendeteksi, menanggapi, dan memulihkan diri dari serangan tersebut. Pertahanan yang efektif bukanlah solusi sekali saja, tetapi upaya berkelanjutan dan berlapis yang mencakup seluruh siklus hidup sistem AI.

Tidak seperti sistem perangkat lunak tradisional, sistem AI pada dasarnya *digerakkan oleh data, probabilistik, dan dinamis*. Perilakunya berevolusi seiring pergeseran distribusi data dari waktu ke waktu, memperkenalkan mode kegagalan dan permukaan serangan yang unik. Karena alasan ini, strategi keamanan harus selaras tidak hanya dengan infrastruktur dan API tetapi juga dengan siklus hidup penuh pengembangan dan penerapan AI. Model ancaman yang selaras dengan siklus hidup mempertimbangkan permukaan serangan di semua tahapan pipeline AI/ML: pengumpulan data, pelatihan, validasi, penerapan (atau inferensi), dan pemantauan. Serangan biasanya terbagi dalam tiga kategori besar: *serangan peracunan*, yang membahayakan integritas data pelatihan; *serangan penghindaran*, yang memanipulasi input selama inferensi untuk menghasilkan output yang salah; dan *serangan eksplorasi atau eksfiltrasi*, yang bertujuan untuk mengekstrak informasi sensitif dari model atau dataset. Kategori-kategori ini dipetakan langsung ke *triad CIA* – kerahasiaan, integritas, dan ketersediaan. Misalnya, serangan peracunan dapat menurunkan utilitas model, sehingga memengaruhi ketersediaan.

Sifat dan tingkat keparahan serangan seringkali bergantung pada tingkat akses dan pengetahuan musuh. Dalam skenario *white-box*, penyerang memiliki visibilitas penuh ke dalam arsitektur model, bobot, dan data pelatihan. Dalam pengaturan *black-box*, penyerang hanya dapat berinteraksi dengan sistem melalui input dan output. Penyerang *gray-box* memiliki pengetahuan parsial, seperti akses ke data pelatihan tetapi bukan internal model. Klasifikasi ini membantu memandu pemilihan pertahanan dan strategi evaluasi yang tepat. Serangan dapat terjadi pada berbagai tahap siklus hidup AI. Selama *fase pelatihan*, penyerang dapat menyuntikkan data berbahaya untuk meracuni model atau menyematkan backdoor yang dapat dipicu kemudian. Dalam lingkungan pembelajaran federasi, klien yang dikompromikan dapat memperkenalkan pembaruan yang berbahaya. Selama *fase inferensi*, musuh dapat membuat input yang menyebabkan kesalahan klasifikasi (penghindaran), mencoba merekonstruksi data pelatihan (inversi model), menentukan apakah data tertentu digunakan dalam pelatihan (inferensi keanggotaan), atau mereplikasi model itu sendiri (ekstraksi model). Masing-masing serangan ini menimbulkan risiko yang berbeda terhadap integritas, kerahasiaan, dan keandalan sistem. Untuk menangkal ancaman ini, *strategi pertahanan berlapis* sangat penting. Ini melibatkan penerapan beberapa pertahanan yang saling melengkapi di seluruh siklus hidup AI:

- *Integritas data* dapat dijaga melalui pelacakan asal usul, validasi skema, dan deteksi anomali. Mempertahankan subset data bersih yang tepercaya sangat penting untuk kalibrasi dan audit.
- *Pertahanan pelatihan* mencakup pelatihan adversarial, regularisasi, privasi diferensial, dan teknik sanitasi data seperti penyaringan outlier dan penyaringan berbasis kerugian.
- *Penerimaan model* harus bergantung pada model pra-terlatih yang ditandatangani dan diverifikasi. Teknik seperti pembersihan neural dan analisis tanda tangan spektral dapat mendeteksi pintu belakang, sementara pemangkasan halus dan distilasi dapat membantu memperbaiki model yang dikompromikan.

- *Pembelajaran federasi* membutuhkan metode agregasi yang kuat, kontrol laju pembelajaran, dan audit partisipasi untuk mencegah peracunan.
- *Pertahanan pada saat inferensi* mencakup validasi input, deteksi data di luar distribusi, pemantauan kueri, pembatasan laju, dan penguatan output untuk mencegah kebocoran informasi.
- *Keamanan operasional* harus mencakup manajemen identitas yang kuat, enkripsi, akses hak istimewa minimal, patching, dan protokol respons insiden.

Pertahanan ini harus terus dievaluasi dan diperbarui seiring munculnya teknik serangan baru. Selanjutnya dalam bab ini, kita akan membahas metrik untuk mengukur kekokohan dan ketahanan, termasuk penggunaan *norma L_p* dan *jarak Wasserstein* untuk mengukur dampak gangguan musuh. Metrik ini menyediakan kerangka kerja praktis untuk menilai efektivitas mekanisme pertahanan dan memandu penyempurnaan mereka dari waktu ke waktu. Untuk survei komprehensif tentang ancaman dan strategi pertahanan khusus AI – termasuk taksonomi, pemetaan siklus hidup, dan teknik evaluasi – pembaca disarankan untuk merujuk pada Sangwan dan rekan-rekan.

14.2 MENDEFINISIKAN MODEL ANCAMAN

Model ancaman yang terstruktur dengan baik sangat penting untuk secara proaktif mengidentifikasi dan mengurangi risiko keamanan yang terkait dengan sistem AI. Tidak seperti sistem perangkat lunak tradisional, AI memperkenalkan kerentanan unik – seperti peracunan data, input yang merugikan, pembajakan cepat, dan pergeseran model – yang memerlukan pertimbangan khusus. Risiko ini muncul karena sistem AI bersifat dinamis dan sering beroperasi sebagai “kotak hitam,” belajar dari hubungan probabilistik yang kompleks dalam data. Hal ini membuat perilaku mereka lebih sulit diprediksi dan dikendalikan, dan akibatnya, postur keamanan mereka lebih sulit untuk dipertahankan.

Tujuan utama dari model ancaman khusus AI adalah untuk menyediakan pendekatan sistematis untuk menganalisis kerentanan ini dan merancang strategi pertahanan yang disesuaikan. Meskipun kerangka kerja yang sudah mapan seperti *STRIDE* tetap berharga untuk mengidentifikasi ancaman dalam sistem perangkat lunak konvensional, kerangka kerja tersebut harus diadaptasi untuk memasukkan aset khusus AI, termasuk data pelatihan, parameter model, dan perilaku probabilistik. Adaptasi ini memastikan bahwa model tersebut memperhitungkan sifat dinamis sistem AI dan permukaan serangan yang berkembang di sepanjang siklus hidupnya. Mengembangkan model ancaman yang efektif dimulai dengan pemahaman komprehensif tentang arsitektur sistem AI, komponennya, dan data yang digunakannya untuk pelatihan dan inferensi. Ini termasuk memetakan seluruh alur pengembangan dan penerapan – mulai dari pengumpulan dan pra-pemrosesan data hingga pelatihan model, validasi, dan pemantauan operasional. Memperlakukan model AI sebagai aset dinamis sangat penting, karena kinerjanya dapat menurun dari waktu ke waktu karena *pergeseran model*, di mana data dunia nyata menyimpang dari distribusi pelatihan asli. Perubahan tersebut dapat menimbulkan kerentanan baru jika tidak segera ditangani.

Proses pemodelan ancaman harus berulang dan multidisiplin. Ini membutuhkan kolaborasi antara ilmuwan data, insinyur keamanan, arsitek perangkat lunak, dan pemangku kepentingan bisnis untuk memastikan bahwa semua perspektif dipertimbangkan. Pendekatan ini membantu mengidentifikasi ancaman yang berpusat pada AI seperti input yang merugikan, serangan peracunan, dan pembajakan cepat, dan mendukung desain pertahanan berlapis yang berkembang seiring dengan sistem. Karena sistem AI pada dasarnya dinamis, pemodelan ancaman tidak dapat dilakukan sekali saja; pemodelan harus diulang secara teratur, terutama setelah perubahan signifikan dalam arsitektur sistem atau sumber data.

14.3 CONTOH STRIDE PADA PIPELINE AI

Untuk membuat konsep pemodelan ancaman lebih konkret, mari kita pertimbangkan bagaimana kerangka kerja STRIDE – yang awalnya dirancang untuk sistem perangkat lunak tradisional – dapat diadaptasi ke pipeline AI. STRIDE adalah akronim untuk enam kategori ancaman: *Spoofing*, *Tampering*, *Repudiation*, *Information Disclosure*, *Denial of Service*, dan *Elevation of Privilege*. Setiap kategori dapat dipetakan ke tahapan spesifik dari siklus hidup AI. Bayangkan sebuah sistem AI yang memproses data perawatan kesehatan yang sensitif untuk memprediksi hasil pasien. Pipeline tersebut mencakup penyerapan data, pelatihan, penyebaran model, dan inferensi. Penerapan STRIDE pada pipeline ini mungkin terlihat seperti ini:

- **Spoofing:** Pada tahap penyerapan data, penyerang dapat menyamar sebagai sumber data tepercaya dan menyuntikkan catatan pasien palsu. Hal ini membahayakan keaslian data dan merusak integritas proses pelatihan.
- **Perubahan Data:** Selama pelatihan, pihak lawan mungkin mengubah dataset pelatihan atau memodifikasi parameter model untuk memasukkan bias atau celah keamanan. Hal ini dapat menyebabkan prediksi yang salah atau pemicu tersembunyi yang aktif dalam kondisi tertentu.
- **Penolakan Tanggung Jawab:** Pada fase penerapan, kurangnya pencatatan dan jejak audit yang tepat dapat memungkinkan pelaku jahat untuk menyangkal tanggung jawab atas perubahan yang tidak sah pada model atau konfigurasinya.
- **Pengungkapan Informasi:** Pada saat inferensi, model mungkin secara tidak sengaja membocorkan informasi pasien yang sensitif melalui skor kepercayaan atau keluaran yang detail. Hal ini sangat mengkhawatirkan di domain yang sensitif terhadap privasi.
- **Penolakan Layanan (DoS):** Penyerang dapat membanjiri API inferensi dengan kueri yang berlebihan, menurunkan kinerja dan ketersediaan sistem. Dalam skenario pembelajaran terfederasi, serangan DoS juga dapat menargetkan server agregasi.
- **Peningkatan Hak Akses:** Dengan memanfaatkan kontrol akses yang lemah, penyerang mungkin mendapatkan hak akses administratif untuk melatih ulang model atau mengubah lingkungan penerapannya, sehingga memungkinkan manipulasi skala besar.

Dengan secara sistematis memeriksa setiap kategori STRIDE di seluruh alur kerja AI, arsitek dapat mengidentifikasi potensi kerentanan dan merancang pertahanan yang tepat sasaran. Misalnya, penerapan otentikasi dan enkripsi yang kuat mengurangi risiko pemalsuan dan

perusakan, sementara privasi diferensial dan sanitasi output mengatasi pengungkapan informasi. Demikian pula, pencatatan dan pemantauan yang kuat membantu mencegah penolakan, dan pembatasan laju yang dikombinasikan dengan deteksi anomali dapat mengurangi dampak serangan DoS.

Latihan ini menunjukkan bahwa mengadaptasi STRIDE ke sistem AI memberikan pendekatan terstruktur untuk mengungkap ancaman yang mungkin terlewatkan. Ini juga memperkuat kebutuhan akan pertahanan berlapis yang mencakup data, model, dan proses operasional. Setelah menetapkan pentingnya pemodelan ancaman dan mengilustrasikan bagaimana kerangka kerja seperti STRIDE dapat diadaptasi untuk sistem AI, sekarang kita beralih ke jenis serangan spesifik yang mengeksploitasi kerentanan di seluruh siklus hidup AI. Memahami kelas serangan ini sangat penting untuk merancang pertahanan yang efektif dan memastikan integritas, kerahasiaan, dan ketersediaan sistem AI.

14.4 SERANGAN TERHADAP SISTEM AI

Sistem AI rentan terhadap berbagai serangan yang bervariasi dalam kecanggihan dan dampaknya. Serangan-serangan ini dapat diklasifikasikan secara luas berdasarkan tingkat pengetahuan dan akses penyerang:

- *Serangan kotak putih* terjadi ketika penyerang memiliki pengetahuan lengkap tentang model, termasuk arsitektur, parameter, dan data pelatihannya. Tingkat akses ini memungkinkan manipulasi yang sangat terarah.
- *Serangan kotak hitam* mengasumsikan penyerang tidak memiliki pengetahuan internal tentang sistem dan hanya dapat berinteraksi melalui input dan output. Terlepas dari keterbatasan ini, penyerang masih dapat menyimpulkan pola dan membuat kueri yang merugikan.
- *Serangan kotak abu-abu* mewakili skenario menengah di mana penyerang memiliki pengetahuan parsial, seperti akses ke data pelatihan atau detail arsitektur yang terbatas.

Dalam kategori ini, beberapa jenis serangan dapat membahayakan sistem AI:

1. **Serangan Penghindaran:** Serangan ini memanipulasi data input selama inferensi untuk menyebabkan kesalahan klasifikasi atau prediksi yang salah. Contoh-contoh yang merugikan adalah bentuk umum dari serangan penghindaran, di mana gangguan kecil yang dirancang dengan cermat menyebabkan perubahan signifikan pada keluaran model tanpa mengubah makna nyata dari input.
2. **Serangan yang Ditargetkan:** Dalam serangan yang ditargetkan, pihak lawan memaksa model untuk mengklasifikasikan input ke dalam kategori tertentu yang salah. Hal ini dapat sangat merusak di domain yang berisiko tinggi seperti perawatan kesehatan atau keuangan, di mana kesalahan klasifikasi dapat menyebabkan keputusan yang merugikan.
3. **Serangan Peracunan:** Serangan peracunan terjadi selama fase pelatihan ketika sampel berbahaya disuntikkan ke dalam dataset. Serangan ini menurunkan kinerja model atau memperkenalkan perilaku tersembunyi yang tetap ada hingga penerapan. Peracunan

dapat bersifat agnostik terhadap kesalahan, bertujuan untuk gangguan umum, atau spesifik terhadap kesalahan, menargetkan kelas atau hasil tertentu.

4. **Serangan Backdoor:** Serangan backdoor memperkenalkan pemicu tersembunyi selama pelatihan yang menyebabkan model menghasilkan keluaran yang telah ditentukan sebelumnya ketika diaktifkan. Pemicu ini seringkali tetap tidak aktif selama operasi normal, sehingga deteksi menjadi sulit. Backdoor dapat disematkan dalam model yang telah dilatih sebelumnya atau dimasukkan melalui pipeline pelatihan yang telah dikompromikan.
5. **Serangan Ekstraksi Model:** Dalam serangan ini, pihak lawan mencoba mereplikasi fungsionalitas model target dengan mengeluarkan kueri dan menganalisis respons. Ekstraksi yang berhasil membahayakan kekayaan intelektual dan dapat memungkinkan serangan lawan selanjutnya.
6. **Serangan Ekstraksi Data:** Serangan ekstraksi data bertujuan untuk memulihkan informasi sensitif dari dataset pelatihan, seringkali melalui teknik inversi model atau inferensi keanggotaan. Serangan ini menimbulkan risiko privasi yang signifikan, terutama ketika model dilatih pada data pribadi atau rahasia.

Penting untuk dicatat bahwa serangan ini dapat terjadi pada berbagai tahap siklus hidup AI. Selama *pelatihan*, ancaman seperti peracunan data dan penyisipan pintu belakang paling umum terjadi. Selama *inferensi*, serangan penghindaran, inversi model, inferensi keanggotaan, dan ekstraksi model menjadi perhatian kritis. Perspektif siklus hidup ini menggarisbawahi perlunya strategi keamanan komprehensif yang mengatasi kerentanan di setiap tahap.

14.5 PRAKTIK AMAN UNTUK SISTEM AI

Merancang pertahanan yang efektif untuk sistem AI membutuhkan pendekatan berlapis yang mengatasi kerentanan di seluruh siklus hidup. Tujuannya bukan untuk menghilangkan serangan sepenuhnya – harapan yang tidak realistis – tetapi untuk secara signifikan meningkatkan upaya yang dibutuhkan musuh untuk berhasil, sehingga mengurangi risiko dan dampak. Salah satu kerentanan umum dalam sistem AI adalah *overfitting*, yang meningkatkan kemungkinan kebocoran data dan pelanggaran privasi. Untuk mengurangi hal ini, teknik regularisasi seperti *dropout* dapat diterapkan selama pelatihan. Selain itu, *distilasi pengetahuan* dapat meningkatkan privasi dengan mentransfer pengetahuan dari model yang kompleks ke model yang lebih sederhana, mengurangi paparan pola sensitif. Teknik lain yang banyak diadopsi adalah *privasi diferensial*, yang memperkenalkan noise terkontrol ke dalam informasi bersama dan membatasi akses kueri. Privasi diferensial dapat diimplementasikan melalui metode seperti *k-anonimitas*, *l-diversitas*, dan penurunan gradien stokastik yang dilindungi privasi diferensial. Meskipun teknik-teknik ini melindungi data, teknik-teknik ini tidak melindungi arsitektur model itu sendiri, yang tetap menjadi perhatian kritis.

Untuk skenario di mana data sensitif harus tetap terenkripsi, *enkripsi homomorfik* menawarkan solusi yang ampuh dengan memungkinkan model untuk dilatih langsung pada data terenkripsi. Namun, protokol enkripsi sederhana tidak cukup; hanya enkripsi homomorfik penuh yang memberikan perlindungan yang kuat. Di luar enkripsi, *pemantauan kueri*

merupakan pertahanan penting terhadap serangan ekstraksi model. Penyerang sering mengeluarkan beberapa kueri untuk menyimpulkan parameter model atau merekonstruksi data pelatihan. Memantau pola kueri – seperti mendeteksi kemiripan yang tinggi atau mengidentifikasi input di luar distribusi – dapat membantu menandai perilaku mencurigakan sebelum kerusakan signifikan terjadi. Selain pertahanan khusus AI ini, praktik keamanan TI standar tetap sangat diperlukan. Kontrol akses yang kuat, otentikasi, dan enkripsi harus diterapkan untuk membatasi akses yang tidak sah dan melindungi infrastruktur. Langkah-langkah mendasar ini melengkapi strategi yang berpusat pada AI dan membentuk dasar dari postur keamanan yang komprehensif. Pertahanan juga dapat disesuaikan dengan jenis serangan tertentu:

- **Terhadap Serangan Penghindaran:** Langkah-langkah proaktif seperti *pelatihan adversarial* mengekspos model ke contoh adversarial selama pelatihan, meningkatkan ketahanan. *Validasi input* dan *deteksi anomali* membantu mengidentifikasi data yang mencurigakan sebelum mencapai model. Teknik seperti *distilasi defensif* memperhalus batas keputusan, membuat model kurang sensitif terhadap gangguan kecil. Metode regularisasi seperti *penalti L_1/L_2* lebih lanjut mencegah model yang terlalu kompleks yang lebih mudah dieksploitasi.
- **Terhadap Serangan Peracunan dan Backdoor:** Strategi pertahanan terbagi dalam empat kategori: *deteksi backdoor*, *perbaikan model*, *pelatihan yang kuat*, dan *pemurnian input*. Teknik deteksi mengidentifikasi pemicu tersembunyi sebelum penyebaran, sementara metode perbaikan menghilangkan komponen berbahaya dari model yang dikompromikan. Pendekatan pelatihan yang tangguh menimbang ulang atau menyaring sampel berbahaya selama pelatihan, dan pemurnian input menghilangkan input yang mencurigakan selama inferensi.
- **Terhadap Serangan Ekstraksi Informasi:** Serangan ini menargetkan data sensitif, arsitektur model, dan hyperparameter. Pertahanan meliputi *privasi diferensial*, *sanitasi output*, dan *pembatasan laju kueri*. Di domain berisiko tinggi seperti perawatan kesehatan atau keuangan, langkah-langkah ini sangat penting untuk mencegah pelanggaran privasi dan pencurian kekayaan intelektual.

TABEL 14.1 Teknik Pertahanan yang Dipetakan ke Jenis Serangan

Jenis Serangan	Tahap Siklus Hidup	Teknik Pertahanan	Deskripsi
Evasion Attacks	Inference	Adversarial Training	Memaparkan model pada contoh adversarial selama pelatihan untuk meningkatkan ketahanan terhadap input perturbations.
Evasion Attacks	Inference	Input Validation & Anomaly Detection	Memvalidasi data yang masuk dan mendeteksi anomali untuk mencegah malicious inputs mencapai model.
Poisoning Attacks	Training	Data Sanitization	Menyaring atau menimbang ulang sampel pelatihan yang mencurigakan untuk mengurangi dampak dari malicious data injections.
Backdoor Attacks	Training	Backdoor Detection & Model Repair	Mengidentifikasi pemicu tersembunyi dan menghapus komponen yang dikompromikan melalui fine-pruning atau retraining.

Model Extraction	Inference	Query Monitoring & Rate Limiting	Melacak pola kueri dan membatasi tingkat permintaan untuk mencegah replikasi model melalui kueri yang berlebihan.
Information Extraction	Inference	Differential Privacy	Memperkenalkan derau terkontrol selama pelatihan atau pembuatan output untuk melindungi data sensitif dan mencegah kebocoran.
Information Extraction	Inference	Output Sanitization	Mengurangi presisi skor keyakinan atau logits untuk meminimalkan risiko pengungkapan informasi.

Menerapkan pertahanan ini hanyalah langkah pertama. Evaluasi berkelanjutan sangat penting untuk memastikan efektivitasnya terhadap ancaman yang terus berkembang. Kerangka kerja pengujian ketahanan, seperti *AutoAttack*, dapat mensimulasikan skenario yang merugikan dan membantu mengidentifikasi kelemahan. Metrik seperti *norma L_p* dan *jarak Wasserstein*, yang dibahas kemudian dalam bab ini, memberikan ukuran kuantitatif ketahanan dan memandu peningkatan iteratif.

Terakhir, pendekatan yang fleksibel dan adaptif diperlukan. Tidak ada satu mekanisme pertahanan pun yang dapat mengatasi semua jenis serangan, dan beberapa pertahanan mungkin menjadi usang seiring munculnya teknik baru. Menggabungkan berbagai strategi – yang selaras dengan siklus hidup sistem AI – menawarkan peluang terbaik untuk menjaga keamanan dari waktu ke waktu. Tabel 14.1 merangkum jenis serangan umum, tahap siklus hidup tempat serangan tersebut terjadi, dan teknik pertahanan yang sesuai beserta deskripsi singkat masing-masing. Tabel ini melengkapi pembahasan rinci di atas dan berfungsi sebagai panduan praktis untuk memilih pertahanan yang tepat berdasarkan sifat ancaman.

14.6 PERTAHANAN AI BERLAPIS

Menerapkan pertahanan individual hanyalah titik awal dalam mengamankan sistem AI. Postur keamanan yang benar-benar tangguh membutuhkan penggabungan beberapa mekanisme pertahanan secara terstruktur dan adaptif. Pendekatan ini memastikan bahwa kerentanan ditangani di semua tahap siklus hidup AI dan bahwa sistem tetap kuat terhadap ancaman yang berkembang. Sistem AI hanya dapat mencapai ketangguhan yang berarti jika strategi pertahanan dan parameter terkaitnya disesuaikan dengan cermat untuk kasus penggunaan spesifik. Misalnya, pelatihan adversarial dapat memperkuat resistensi terhadap serangan penghindaran, tetapi tidak banyak membantu mencegah peracunan data atau ekstraksi model. Demikian pula, privasi diferensial dapat mengurangi kebocoran informasi tetapi tidak dapat mendeteksi pemicu pintu belakang yang tertanam dalam model yang telah dilatih sebelumnya. Keterbatasan ini menyoroti perlunya *strategi pertahanan berlapis* yang mengintegrasikan teknik-teknik yang saling melengkapi.

Evaluasi berkelanjutan sangat penting. Kerangka kerja pengujian ketahanan seperti *AutoAttack* dapat mensimulasikan skenario musuh dan mengungkap kelemahan dalam pertahanan yang diterapkan. Metrik seperti *norma L_p* dan *jarak Wasserstein*, yang dibahas kemudian dalam bab ini, memberikan ukuran kuantitatif ketahanan dan memandu peningkatan iteratif. Seiring kemajuan pembelajaran mesin yang bersifat antagonis, teknik serangan baru seringkali membuat pertahanan yang sebelumnya efektif menjadi usang.

Akibatnya, strategi keamanan harus ditinjau ulang secara berkala untuk mempertahankan efektivitasnya. Pendekatan terstruktur untuk mengatur pertahanan berdasarkan tahap siklus hidup menawarkan beberapa keuntungan. Ini membantu menentukan urutan optimal untuk menerapkan teknik dan memastikan bahwa celah-celah kritis tidak terabaikan. Misalnya, pertahanan yang diimplementasikan selama pelatihan – seperti sanitasi data dan agregasi yang kuat – harus dilengkapi dengan langkah-langkah pada waktu inferensi seperti validasi input, pemantauan kueri, dan pengerasan output. Pengamanan operasional, termasuk manajemen identitas yang kuat, enkripsi, dan protokol respons insiden, memberikan lapisan perlindungan tambahan.

Penting juga untuk mengenali interaksi antara berbagai jenis serangan dan pertahanan. Misalnya, apakah peracunan data meningkatkan kerentanan terhadap serangan penghindaran? Dapatkah metode deteksi contoh yang merugikan juga membantu mengidentifikasi input yang diracuni? Menjelajahi hubungan ini dapat menghasilkan strategi pertahanan yang lebih efisien dan holistik. Pada akhirnya, tidak ada satu mekanisme pun yang dapat mengatasi semua vektor serangan. Pendekatan fleksibel dan adaptif yang menggabungkan beberapa pertahanan – selaras dengan siklus hidup sistem AI – menawarkan peluang terbaik untuk menjaga keamanan dari waktu ke waktu. Strategi berlapis ini harus didukung oleh pemantauan berkelanjutan, audit berkala, dan pemodelan ancaman proaktif untuk mengantisipasi risiko yang muncul.

14.7 MENGUKUR KEKUATAN PERTAHANAN SISTEM AI

Membangun sistem AI yang aman membutuhkan pendekatan proaktif dan sistematis yang mencakup seluruh siklus hidup – mulai dari pengumpulan data dan pelatihan model hingga penerapan dan pemantauan berkelanjutan. Meskipun penerapan pertahanan individual itu penting, praktik terbaik memastikan bahwa langkah-langkah ini diterapkan secara konsisten dan efektif. Pertama, organisasi harus mengadopsi *pedoman keamanan TI standar* sebagai dasar. Ini termasuk kontrol akses yang kuat, enkripsi data saat transit dan saat istirahat, pengembangan API yang aman, dan manajemen identitas. Langkah-langkah ini melindungi infrastruktur yang mendasarinya dan mencegah akses tidak sah ke komponen penting dari sistem AI. Sama pentingnya adalah integritas data pelatihan. Data harus bersumber dari penyedia yang terverifikasi dan tepercaya serta terus dipantau untuk anomali. Teknik seperti *deteksi outlier*, *analisis distribusi label*, dan bahkan *inspeksi visual* untuk dataset yang lebih kecil dapat membantu mengidentifikasi sampel yang rusak atau korup sebelum membahayakan model. Melengkapi pelatihan dengan dataset yang bersih dan terkurasi serta menerapkan augmentasi data dapat lebih mengurangi risiko serangan peracunan.

Saat menggunakan model yang telah dilatih sebelumnya atau kode eksternal, organisasi harus memverifikasi integritasnya melalui *tanda tangan digital* dan pemeriksaan asal-usul. Idealnya, pengembangan dan pelatihan model harus dilakukan di lingkungan yang terkontrol untuk meminimalkan paparan terhadap kode atau data berbahaya. Untuk perlindungan tambahan, teknik seperti *pengelompokan gradien*, *analisis spektral*, dan *statistik berbasis perturbasi* dapat digunakan untuk mendeteksi inkonsistensi dalam perilaku model

yang mungkin mengindikasikan pintu belakang atau kerentanan tersembunyi lainnya. Untuk melindungi privasi, terapkan *privasi diferensial* selama pelatihan dan pertimbangkan *enkripsi homomorfik* untuk skenario data sensitif. Teknik-teknik ini membantu mencegah kebocoran informasi rahasia sambil mempertahankan kegunaan model. Untuk keamanan pada saat inferensi, *pemantauan kueri*, *pembatasan laju*, dan *sanitasi keluaran* sangat penting untuk melawan serangan ekstraksi model dan pengungkapan informasi.

Terakhir, keamanan bukanlah upaya sekali saja. Sistem AI harus menjalani *validasi berkelanjutan dan audit berkala* untuk mendeteksi ancaman yang muncul. Sistem deteksi intrusi, pemantauan anomali, dan evaluasi ketahanan menggunakan kerangka kerja pengujian adversarial, seperti *AutoAttack*, harus menjadi bagian dari operasi rutin. Seiring berkembangnya ancaman, pertahanan harus diperbarui untuk mempertahankan ketahanan terhadap vektor serangan baru. Pada bagian selanjutnya, kita akan membahas metrik yang dapat digunakan untuk mengukur kekokohan dan ketahanan sistem AI.

14.8 MENGUKUR EFEKTIVITAS PERTAHANAN

Mengevaluasi kekokohan dan ketahanan sistem AI membutuhkan metrik yang terdefinisi dengan baik yang mengukur efektivitas langkah-langkah keamanan yang diterapkan. Metrik ini harus mencerminkan kemampuan sistem untuk menahan gangguan yang merugikan, menahan pelanggaran privasi, dan pulih dari serangan peracunan atau serangan pintu belakang. Kekokohan biasanya mengacu pada ketahanan sistem terhadap contoh-contoh yang merugikan, sedangkan ketahanan mencakup kapasitasnya untuk mempertahankan integritas dan kerahasiaan dalam skenario ancaman yang lebih luas seperti kebocoran data atau kompromi model. Pendekatan umum untuk mengukur kekokohan yang merugikan melibatkan pengukuran jarak antara input asli dan input yang terganggu. Keluarga metrik *norma L_p* banyak digunakan untuk tujuan ini. Untuk vektor X dalam ruang n -dimensi dan bilangan real $p \geq 1$, *norma L_p* didefinisikan sebagai:

$$\|X\|_p = (\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}} \quad (14.1)$$

Untuk menilai dampak gangguan yang merugikan, jarak antara input asli X dan pasangannya yang merugikan X' dihitung sebagai:

$$\|X - X'\|_p = (\sum_{i=1}^n |x_i - x'_i|^p)^{\frac{1}{p}} \quad (14.2)$$

Nilai p yang berbeda menghasilkan konsep jarak yang berbeda, yang memengaruhi karakteristik gangguan. Misalnya, norma L_2 menekankan jarak Euclidean, sedangkan norma L_∞ berfokus pada perubahan maksimum di seluruh dimensi.

Namun, *norma L_p* memiliki keterbatasan, karena tidak selalu selaras dengan persepsi manusia. Perubahan *norma L_p* yang kecil dapat tampak signifikan secara visual, sementara transformasi semantik tidak terlihat. Misalnya, distorsi geometris mungkin memiliki *norma L_p* yang besar. Ada pendekatan alternatif termasuk *jarak Wasserstein* dan deformasi geometris

yang dapat menangkap perbedaan persepsi dengan lebih baik dan dapat membuat gangguan adversarial lebih sulit dideteksi. Jarak Wasserstein mengukur biaya minimal untuk mengubah satu distribusi menjadi distribusi lain, seperti menggeser intensitas piksel dalam gambar. Ini sangat efektif dalam pembelajaran mesin adversarial, di mana serangan bertujuan untuk menimbulkan perubahan minimal namun berdampak. Jarak Wasserstein yang kecil menunjukkan bahwa transformasi adversarial melibatkan pergeseran spasial yang halus, sehingga lebih sulit dideteksi.

Selain pembangkitan serangan, teknik berbasis Wasserstein juga mendukung pertahanan. *Pelatihan adversarial Wasserstein* meningkatkan ketahanan model dengan memperhitungkan pergeseran distribusi daripada gangguan terisolasi. *Penghalusan Wasserstein* lebih lanjut memperkuat pertahanan dengan memastikan ketahanan di seluruh ruang transformasi yang mungkin, menawarkan jaminan terhadap kelas serangan yang lebih luas. Bersama-sama, metrik ini menyediakan kerangka kerja yang ketat untuk mengevaluasi dan menyempurnakan mekanisme pertahanan. Metrik ini memungkinkan praktisi AI untuk mensimulasikan skenario adversarial, mengukur ketahanan sistem, dan memandu peningkatan iteratif. Seiring berkembangnya lanskap ancaman, validasi berkelanjutan menggunakan metrik tersebut menjadi penting untuk mempertahankan postur AI yang kuat dan aman.

14.9 RINGKASAN

Mengamankan sistem AI adalah domain khusus dalam keamanan siber yang membutuhkan strategi yang disesuaikan untuk mengatasi karakteristik unik teknologi AI. Tidak seperti perangkat lunak tradisional, sistem AI bersifat *dinamis, berbasis data, dan probabilistik*, yang memperkenalkan kerentanan baru di sepanjang siklus hidupnya – mulai dari pengumpulan data dan pelatihan hingga penerapan dan inferensi. Bab ini telah menguraikan kerangka kerja komprehensif untuk keamanan AI, dimulai dengan konsep dasar seperti *ancaman, kerentanan, serangan, dan pertahanan*. Kami menekankan pentingnya pemodelan ancaman yang sadar siklus hidup, khususnya adaptasi kerangka kerja yang sudah mapan seperti *STRIDE* ke dalam alur kerja AI. Dengan memetakan kelas serangan – *penghindaran, peracunan, pintu belakang, dan ekstraksi* – ke tahapan siklus hidup tertentu, kami menunjukkan bagaimana risiko keamanan muncul dan berkembang dari waktu ke waktu.

Strategi pertahanan harus berlapis dan adaptif. Teknik-teknik seperti *pelatihan adversarial, privasi diferensial, enkripsi homomorfik, validasi input, dan pemantauan kueri* dibahas secara rinci, bersama dengan penerapannya pada berbagai vektor serangan. Kami juga menyoroti pentingnya evaluasi berkelanjutan menggunakan metrik ketahanan seperti *norma L_p* dan *jarak Wasserstein*, yang memberikan wawasan kuantitatif tentang kemampuan sistem untuk melawan dan pulih dari gangguan yang merugikan. Pada akhirnya, mengamankan sistem AI membutuhkan pendekatan proaktif dan multidisiplin yang mengintegrasikan pertahanan teknis dengan pengamanan operasional. Audit rutin, pemodelan ancaman, dan pengujian ketahanan harus menjadi praktik rutin. Seiring dengan terus berkembangnya teknik

penyerang, pertahanan kita pun harus ikut berkembang – memastikan bahwa sistem AI tetap dapat dipercaya, tangguh, dan selaras dengan standar etika dan peraturan.

14.10 PERTANYAAN

1. Bedakan dengan jelas antara ancaman, kerentanan, serangan, dan pertahanan dalam sistem AI. Jelaskan bagaimana konsep-konsep ini saling terkait.
2. Jelaskan mengapa pemodelan ancaman AI berbeda dari pemodelan ancaman perangkat lunak tradisional dan jelaskan bagaimana model STRIDE dapat diadaptasi untuk sistem AI.
3. Klasifikasikan serangan AI berdasarkan pengetahuan penyerang dan sebutkan jenis-jenis serangan utama di seluruh siklus hidup pembelajaran mesin (ML).
4. Ringkas teknik pertahanan terhadap ancaman penghindaran dan ekstraksi informasi. Sertakan kontrol proaktif dan reaktif dalam penjelasan Anda.
5. Uraikan praktik terbaik untuk meminimalkan risiko peracunan dan pintu belakang. Jelaskan metode untuk mengevaluasi efektivitas pertahanan ini.
6. Metode mana yang terutama melindungi privasi data tetapi bukan arsitektur model?
 - A. Enkripsi homomorfik
 - B. Privasi diferensial
 - C. Distilasi defensif
 - D. Pemangkasan model
7. Serangan mana yang secara eksplisit bertujuan untuk menyimpulkan apakah suatu catatan tertentu digunakan dalam pelatihan?
 - A. Ekstraksi model
 - B. Inferensi keanggotaan
 - C. Penghindaran
 - D. Aktivasi pintu belakang
8. Serangan pintu belakang biasanya disisipkan selama dan dipicu selama _____ dan dipicu selama .
 - A. Inferensi; pelatihan
 - B. Pelatihan; inferensi
 - C. Pengumpulan data; hanya penyebaran
 - D. Penyebaran; penonaktifan

BAB 15

KEAMANAN SISTEM AI DALAM PRAKTIK

15.1 DEFINISI PIPELINE PEMBERSIHAN DATA

Dalam bab ini, kami memberikan pengantar formal tentang keamanan sistem AI dan membahas cara mengoperasikan teknologi AI dengan aman di seluruh siklus hidupnya. Keamanan sistem AI mencakup desain, pengembangan, dan penerapan sistem AI yang andal, bermanfaat, dan selaras dengan nilai-nilai manusia. Karena sistem AI semakin memengaruhi keputusan dalam perawatan kesehatan, keuangan, transportasi, dan kebijakan publik, taruhan ketidakselarasan, bias, dan konsekuensi yang tidak diinginkan meningkat secara signifikan. Tujuan keamanan AI adalah untuk mencegah bahaya bagi individu, institusi, dan lingkungan. Hal ini membutuhkan penanganan bias algoritmik, penyalahgunaan data, ancaman yang merugikan, dan perilaku yang muncul yang mungkin bertentangan dengan norma etika. Strategi seperti pengujian yang kuat, mitigasi bias, penyelarasan AI, penguatan keamanan, transparansi, dan tata kelola etis sangat penting untuk meminimalkan risiko dan memaksimalkan dampak positif AI.

Sistem AI beroperasi pada skala dan kecepatan yang memperkuat manfaat dan risiko. Sifat otonomnya memperkenalkan mode kegagalan baru, termasuk manipulasi yang merugikan, pergeseran model, dan injeksi cepat. Oleh karena itu, keamanan harus tertanam di seluruh siklus hidup AI – mulai dari pengumpulan data dan pelatihan model hingga penyebaran dan pemantauan. Bab ini dibangun berdasarkan diskusi sebelumnya tentang pembelajaran mesin, pembelajaran mendalam, NLP, LLM, sistem terdistribusi, dan keamanan AI untuk membangun kerangka kerja holistik untuk keamanan sistem AI. Untuk membangun sistem AI yang aman, dapat dipercaya, dan selaras dengan nilai-nilai manusia, sangat penting untuk memahami dimensi dasar keamanan. Dimensi ini mencakup keandalan teknis, keselarasan etis, transparansi, dan akuntabilitas. Pada bagian selanjutnya, kita akan memeriksa aspek-aspek kunci keamanan sistem AI yang harus ditangani di seluruh siklus hidup AI – mulai dari pengumpulan data dan pelatihan model hingga penyebaran dan pemantauan – untuk memastikan operasi AI yang bertanggung jawab dan tangguh.

15.2 ASPEK KUNCI KEAMANAN SISTEM AI

Keamanan sistem AI mencakup serangkaian pertimbangan teknis dan etis yang luas yang harus ditangani di seluruh siklus hidup AI. Aspek-aspek ini berfokus pada karakteristik operasional dan perilaku tingkat sistem yang menentukan apakah suatu sistem AI dapat dipercaya untuk berkinerja andal, aman, dan bertanggung jawab.

1. **Keandalan Operasional:** Sistem AI harus berfungsi secara konsisten di berbagai kondisi, termasuk input yang bising, pergeseran data, dan lingkungan yang merugikan. Keandalan mencakup toleransi kesalahan, degradasi yang bertahap, dan ketahanan terhadap kegagalan yang tidak terduga.

2. **Keamanan dan Ketahanan:** Sistem AI harus dilindungi dari ancaman seperti ekstraksi model, peracunan data, dan serangan penghindaran. Langkah-langkah keamanan harus mencakup seluruh siklus hidup – dari pengumpulan dan pelatihan data hingga inferensi dan pemantauan – dan mencakup teknik seperti pelatihan adversarial, privasi diferensial, dan pemantauan kueri.
3. **Praktik Data Etis:** Data yang digunakan untuk pelatihan dan inferensi harus dikumpulkan, diproses, dan disimpan dengan cara yang menghormati privasi, persetujuan, dan keadilan. Ini termasuk anonimisasi, mitigasi bias, dan kepatuhan terhadap kerangka hukum seperti GDPR.
4. **Integrasi Siklus Hidup:** Keamanan harus tertanam dalam setiap fase siklus hidup AI – perencanaan, desain, implementasi, penyebaran, dan pemantauan. Ini termasuk praktik pengembangan yang aman, validasi berkelanjutan, dan umpan balik untuk penyempurnaan model.
5. **Kolaborasi Antar Disiplin:** Keamanan AI mendapat manfaat dari beragam perspektif, termasuk etika, hukum, psikologi, dan keahlian khusus bidang. Pengembangan kolaboratif memastikan bahwa sistem tidak hanya secara teknis baik tetapi juga bertanggung jawab secara sosial.

Aspek-aspek ini membentuk dasar untuk membangun sistem AI yang tangguh secara teknis dan selaras secara sosial. Hal ini didukung oleh prinsip-prinsip keselamatan, yang memberikan panduan normatif dan berorientasi desain yang akan dibahas selanjutnya.

15.3 PRINSIP-PRINSIP KESELAMATAN AI

Prinsip-prinsip keselamatan AI memberikan kerangka kerja normatif untuk memandu desain dan tata kelola sistem AI. Prinsip-prinsip ini menekankan nilai-nilai dan standar yang harus dijunjung tinggi oleh sistem AI untuk memastikan penerapan yang etis dan bertanggung jawab.

1. **Keselajaran:** Sistem AI harus mengejar tujuan yang konsisten dengan nilai-nilai manusia dan norma-norma masyarakat. Ini termasuk merancang mekanisme untuk menafsirkan maksud pengguna secara akurat dan mengkalibrasi ulang tujuan dari waktu ke waktu untuk mencegah penyimpangan etika.
2. **Ketangguhan:** Sistem AI harus tahan terhadap masukan yang merugikan, kerusakan data, dan anomali operasional. Ketangguhan memastikan bahwa sistem berperilaku dapat diprediksi dan mempertahankan integritas di bawah tekanan atau serangan.
3. **Transparansi:** Sistem AI harus mudah dipahami dan diaudit. Transparansi mencakup pendokumentasian keputusan desain, memungkinkan penelusuran keluaran, dan memberikan penjelasan untuk perilaku model.
4. **Akuntabilitas:** Sistem AI harus diatur oleh aturan yang jelas dan mekanisme pengawasan. Akuntabilitas memastikan bahwa pengembang, operator, dan organisasi dapat dimintai pertanggungjawaban atas hasil sistem, dan bahwa pengguna memiliki jalan keluar jika terjadi kerugian.

Prinsip-prinsip ini melengkapi aspek-aspek kunci keselamatan AI dengan memberikan panduan yang berorientasi pada etika dan tata kelola. Bersama-sama, prinsip-prinsip ini mendukung pengembangan sistem AI yang tidak hanya secara teknis baik tetapi juga selaras dengan harapan masyarakat dan standar hukum.

15.4 TANTANGAN KETIDAKSESUAIAN AI

Ketidaksesuaian AI mengacu pada perbedaan antara tujuan yang dipelajari oleh sistem AI dan niat atau nilai-nilai pemangku kepentingan manusia. Seiring sistem AI menjadi lebih otonom dan mampu, risiko ketidaksesuaian meningkat – terutama ketika sistem mengoptimalkan tujuan proksi yang menyimpang dari harapan manusia. Ketidaksesuaian dapat terwujud dalam beberapa cara:

- **Pergeseran Tujuan:** Sistem AI yang dilatih pada data historis dapat terus mengoptimalkan tujuan yang sudah usang seiring dengan evolusi kondisi dunia nyata. Tanpa mekanisme untuk kalibrasi ulang yang berkelanjutan, pergeseran ini dapat menyebabkan keputusan yang secara teknis benar tetapi secara etis atau sosial berbahaya.
- **Manipulasi Spesifikasi:** Sistem AI dapat mengeksploitasi celah dalam fungsi penghargaan atau batasan yang kurang jelas. Misalnya, sistem yang bertugas memaksimalkan keterlibatan mungkin mempromosikan konten yang sensasional atau menyesatkan, mencapai tujuannya sambil merusak kepercayaan pengguna.
- **Perilaku yang Muncul:** Model kompleks, terutama model bahasa besar dan agen pembelajaran penguatan, dapat menunjukkan perilaku yang tidak diprogram atau diantisipasi secara eksplisit. Perilaku ini dapat bertentangan dengan nilai-nilai manusia atau norma keselamatan, khususnya dalam tugas terbuka atau lingkungan multi-agen.
- **Kurangnya Interpretasi:** Ketika keputusan model tidak jelas, menjadi sulit untuk mendeteksi ketidaksesuaian atau melakukan intervensi secara efektif. Hal ini terutama bermasalah di bidang-bidang berisiko tinggi seperti perawatan kesehatan, peradilan pidana, atau sistem otonom.

Mengatasi ketidaksesuaian membutuhkan kombinasi strategi teknis dan tata kelola:

- **Pengawasan Manusia dalam Proses:** Menggabungkan penilaian manusia ke dalam proses pengambilan keputusan membantu memastikan bahwa keluaran AI tetap selaras dengan nilai-nilai manusia.
- **Pembelajaran Preferensi dan Pembelajaran Penguatan Terbalik:** Teknik-teknik ini bertujuan untuk menyimpulkan tujuan manusia dari perilaku atau umpan balik, memungkinkan sistem untuk beradaptasi lebih efektif dengan maksud pengguna.
- **Evaluasi dan Pemantauan yang Kuat:** Pengujian, audit, dan umpan balik berkelanjutan sangat penting untuk mendeteksi dan memperbaiki perilaku yang tidak sesuai dari waktu ke waktu.
- **Desain dan Tata Kelola yang Etis:** Menanamkan prinsip-prinsip etika ke dalam desain sistem – seperti keadilan, akuntabilitas, dan transparansi – membantu mengurangi risiko ketidaksesuaian dan mendukung penerapan yang bertanggung jawab.

Setelah mengeksplorasi tantangan teknis dan etis dari ketidaksesuaian AI, penting untuk memahami bagaimana pemerintah dan badan pengatur menanggapi. Semakin banyak kerangka kerja legislatif dan kebijakan telah diperkenalkan untuk memandu pengembangan sistem AI yang aman dan bertanggung jawab. Regulasi ini bertujuan untuk memastikan bahwa teknologi AI selaras dengan nilai-nilai manusia, melindungi hak-hak individu, dan mengurangi risiko yang terkait dengan bias, pelanggaran privasi, dan konsekuensi yang tidak diinginkan.

15.5 PERATURAN KEAMANAN SISTEM AI

Pemerintah dan organisasi internasional semakin banyak memberlakukan undang-undang dan pedoman untuk mengatur desain, penerapan, dan penggunaan sistem AI. Peraturan ini bertujuan untuk memastikan bahwa teknologi AI aman, transparan, adil, dan akuntabel. Berikut adalah empat upaya regulasi utama yang membahas keamanan sistem AI:

1. **Undang-Undang AI Uni Eropa:** Undang-Undang AI Uni Eropa adalah kerangka kerja legislatif komprehensif yang dirancang untuk mengatur sistem AI berdasarkan tingkat risiko yang ditimbulkannya. Undang-undang ini mewajibkan penilaian risiko yang ketat, kewajiban transparansi, dan langkah-langkah tata kelola data untuk aplikasi berisiko tinggi. Undang-undang ini menegakkan standar kualitas data yang tinggi, melindungi data pribadi, dan mengharuskan individu untuk diberi informasi ketika berinteraksi dengan sistem AI. Tujuannya adalah untuk menumbuhkan kepercayaan pada AI dengan memastikan keamanan, keadilan, dan kepatuhan terhadap hukum privasi Uni Eropa.
2. **Pedoman Etika Uni Eropa untuk AI yang Dapat Dipercaya:** Dikembangkan oleh Komisi Eropa, pedoman ini menekankan prinsip-prinsip etika seperti otonomi manusia, ketahanan, privasi, dan transparansi. Mereka mengadvokasi sistem AI yang tidak hanya andal secara teknis tetapi juga selaras secara sosial dengan nilai-nilai Eropa. Kerangka kerja ini mendorong perlindungan yang kuat untuk data pribadi dan mempromosikan inovasi etis melalui keterlibatan pemangku kepentingan dan penilaian risiko.
3. **Undang-Undang Inisiatif Kecerdasan Buatan Nasional AS:** Inisiatif federal AS ini mendukung riset, pengembangan, dan pendidikan AI sambil mempromosikan standar etika dan kolaborasi internasional. Inisiatif ini mendorong kemajuan teknologi AI yang bertanggung jawab dengan menyeimbangkan inovasi dengan tata kelola. Undang-undang ini menekankan manfaat sosial AI dan berupaya menyelaraskan upaya AS dengan praktik terbaik global.
4. **Undang-Undang Akuntabilitas Algoritma AS (AAA):** AAA berfokus pada peningkatan transparansi dan akuntabilitas dalam sistem pengambilan keputusan otomatis. Undang-undang ini mewajibkan perusahaan untuk melakukan penilaian dampak untuk mengidentifikasi dan mengurangi bias, memastikan privasi data, dan mengungkapkan bagaimana sistem AI membuat keputusan. Undang-undang ini bertujuan untuk mencegah diskriminasi dan melindungi hak konsumen, memperkuat praktik AI yang etis dan adil.

Upaya regulasi ini mencerminkan pengakuan global akan kebutuhan untuk memandu pengembangan AI dengan mengutamakan keselamatan, etika, dan hak asasi manusia.

Meskipun setiap kerangka kerja bervariasi dalam cakupan dan penegakannya, secara kolektif mereka berkontribusi pada ekosistem AI yang lebih aman dan lebih terpercaya. Meskipun peraturan pemerintah memberikan pengawasan di tingkat hukum dan kebijakan, banyak lembaga dan organisasi sektor swasta telah mengembangkan kerangka kerja dan pedoman sukarela untuk mempromosikan keselamatan AI. Upaya-upaya ini melengkapi inisiatif regulasi dengan menawarkan alat praktis, prinsip etika, dan strategi teknis yang dapat diadopsi oleh organisasi untuk membangun sistem AI yang terpercaya dan tangguh.

15.6 PANDUAN KEAMANAN AI

Selain undang-undang formal, beberapa lembaga dan organisasi sektor swasta telah merilis kerangka kerja yang membantu memandu desain, pengembangan, dan penerapan sistem AI yang aman. Kerangka kerja ini menekankan keselarasan etika, ketahanan teknis, dan manajemen risiko berkelanjutan. Berikut adalah tiga contoh yang berpengaruh:

1. **Kerangka Kerja Manajemen Risiko AI:** Dikembangkan oleh National Institute of Standards and Technology (NIST), kerangka kerja ini menyediakan pedoman terstruktur untuk mengidentifikasi, menilai, dan mengurangi risiko dalam sistem AI. Kerangka kerja ini menekankan keamanan, keandalan, dan ketahanan di seluruh siklus hidup AI. Rekomendasi utama meliputi penilaian risiko berkelanjutan, keterlibatan pemangku kepentingan, dan strategi mitigasi adaptif. Kerangka kerja ini dirancang untuk mendukung organisasi publik dan swasta dalam membangun sistem AI yang kuat dan aman.
2. **Desain yang Selaras Secara Etis IEEE:** Ditulis oleh IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems, kerangka kerja ini menguraikan prinsip-prinsip etika untuk pengembangan AI. Kerangka kerja ini mempromosikan desain yang berpusat pada manusia, transparansi, akuntabilitas, dan privasi. Pedoman ini mendorong pengembang untuk menyematkan pertimbangan etis di setiap tahap siklus hidup AI dan menganjurkan keterlibatan pemangku kepentingan yang inklusif. Kerangka kerja ini juga mendukung pembuatan standar yang mencegah kerugian dan mempromosikan keadilan, menjadikannya sumber daya yang berharga untuk menyelaraskan sistem AI dengan nilai-nilai masyarakat.
3. **Kerangka Kerja AI Aman Google:** Kerangka Kerja AI Aman Google berfokus pada pengintegrasian praktik terbaik keamanan di seluruh siklus hidup pengembangan AI. Ini mencakup pemodelan ancaman, pengkodean yang aman, dan evaluasi berkelanjutan sistem AI terhadap kerentanan yang muncul. Kerangka kerja ini menekankan strategi pertahanan proaktif – dari desain awal hingga penyebaran dan pemeliharaan – dan bertujuan untuk memastikan bahwa sistem AI efektif dan tangguh terhadap eksploitasi berbahaya.

Kerangka kerja ini menawarkan panduan yang dapat ditindaklanjuti bagi organisasi yang berupaya mengoperasionalkan keamanan AI. Dengan menggabungkan prinsip-prinsip etika dengan pengamanan teknis, mereka membantu menjembatani kesenjangan antara kebijakan

dan praktik, memungkinkan pengembangan sistem AI yang tidak hanya patuh tetapi juga dapat dipercaya dan bertanggung jawab secara sosial.

15.7 PRAKTIK KEAMANAN AI

Memastikan keamanan sistem AI membutuhkan pendekatan proaktif dan sistematis yang mencakup seluruh siklus hidup pengembangan – mulai dari perencanaan dan desain awal hingga penerapan, pemantauan, dan peningkatan berkelanjutan. Meskipun prinsip dan kerangka kerja memberikan panduan tingkat tinggi, praktik terbaik menerjemahkan hal ini menjadi langkah-langkah yang dapat ditindaklanjuti yang dapat diimplementasikan oleh pengembang, organisasi, dan pengguna untuk membangun sistem AI yang dapat dipercaya dan tangguh.

Praktik mendasar adalah penerapan siklus hidup pengembangan yang aman, yang mengintegrasikan pertimbangan keselamatan dan keamanan ke dalam setiap fase pembuatan sistem AI. Ini termasuk pemodelan ancaman selama perencanaan, standar pengkodean yang aman selama implementasi, pengujian kerentanan yang ketat, dan pembaruan berkelanjutan untuk mengatasi risiko yang muncul. Menanamkan keamanan sejak awal membantu mencegah kegagalan yang mahal dan memastikan bahwa sistem tangguh sejak awal perancangan. Melindungi privasi pengguna adalah komponen penting lain dari keamanan AI. Teknik anonimisasi data – seperti k-anonimitas, privasi diferensial, dan pembuatan data sintetis – membantu memastikan bahwa informasi yang dapat diidentifikasi secara pribadi dihapus atau dikaburkan sebelum pelatihan atau analisis. Metode ini mengurangi risiko identifikasi ulang dan mendukung kepatuhan terhadap peraturan privasi seperti GDPR.

Setelah diterapkan, sistem AI harus terus dipantau untuk mendeteksi penyimpangan dari perilaku yang diharapkan. Umpan balik memungkinkan penyempurnaan model secara dinamis berdasarkan kinerja dunia nyata dan masukan pengguna. Ini sangat penting untuk mengurangi penyimpangan model dan menjaga keselarasan dengan nilai-nilai manusia dari waktu ke waktu. Kolaborasi interdisipliner memainkan peran penting dalam mengatasi tantangan multifaset dari keamanan AI. Tim pengembang harus mencakup para ahli dari bidang etika, hukum, psikologi, bidang spesifik domain, dan rekayasa AI. Keragaman perspektif ini memastikan bahwa sistem tidak hanya secara teknis baik tetapi juga bertanggung jawab secara sosial dan selaras secara etis.

Organisasi juga harus siap untuk menanggapi pelanggaran keamanan atau perilaku sistem yang tidak terduga. Rencana respons insiden yang kuat harus menguraikan prosedur untuk mengidentifikasi, melaporkan, menahan, dan memulihkan dari insiden. Strategi yang terdefinisi dengan baik meminimalkan kerugian dan memulihkan integritas sistem dengan cepat, memperkuat kepercayaan pada teknologi AI. Audit dan pemeriksaan kepatuhan secara berkala sangat penting untuk menjaga akuntabilitas. Evaluasi ini membantu memastikan bahwa sistem AI mematuhi standar keselamatan yang telah ditetapkan, pedoman etika, dan persyaratan peraturan. Audit dapat mengungkapkan risiko tersembunyi, mendorong perbaikan tepat waktu, dan menunjukkan komitmen terhadap tata kelola AI yang bertanggung jawab.

Terakhir, mendidik pengguna adalah aspek kunci dari penerapan AI yang aman. Pengguna harus dilatih untuk memahami kemampuan dan keterbatasan sistem AI, mengenali potensi risiko, dan mengikuti praktik terbaik untuk penggunaan yang bertanggung jawab. Pengguna yang terinformasi lebih siap untuk berinteraksi dengan teknologi AI secara aman dan melaporkan masalah ketika muncul. Bersama-sama, praktik terbaik ini membentuk tulang punggung operasional keselamatan AI. Ketika diimplementasikan secara konsisten dan berkoordinasi dengan prinsip-prinsip etika, kerangka peraturan, dan pengamanan teknis, praktik-praktik ini membantu memastikan bahwa sistem AI tetap dapat dipercaya, tangguh, dan bermanfaat bagi masyarakat.

15.8 MERANCANG AI UNTUK SITUASI KRITIS

Meskipun prinsip keselamatan umum dan praktik siklus hidup sangat penting, sistem AI kritis keselamatan – seperti yang digunakan dalam kendaraan otonom, diagnostik perawatan kesehatan, atau pemeliharaan prediktif – memerlukan strategi desain khusus. Sebuah studi baru-baru ini oleh Suryanarayana dan rekan-rekan memperkenalkan arsitektur berbasis komponen yang mengintegrasikan prinsip-prinsip rekayasa perangkat lunak dengan analitik prediktif untuk meningkatkan keselamatan. Pendekatan ini berfokus pada tiga strategi inti: *desain aman*, *margin aman*, dan *kegagalan aman*, masing-masing menangani risiko spesifik yang terkait dengan kualitas data, pergeseran distribusi, dan mitigasi kegagalan.

- **Desain Aman:** Desain aman melibatkan penyematan fitur keamanan ke dalam arsitektur sistem untuk mencegah atau mengurangi risiko internal dan eksternal. Ini termasuk pra-pemrosesan data yang ketat (misalnya, menangani nilai yang hilang, mendeteksi outlier), rekayasa fitur, dan penyeimbangan kelas untuk memastikan bahwa model dilatih pada dataset yang bersih, representatif, dan berkualitas tinggi. Praktik-praktik ini mengurangi bias, meningkatkan keadilan, dan meningkatkan ketahanan model sebelum penerapan.
- **Margin Aman:** Margin aman mengacu pada penyangga atau toleransi yang dibangun ke dalam sistem AI untuk memastikan operasi yang andal dalam kondisi ketidakpastian. Teknik seperti *uji Kolmogorov–Smirnov* membantu mendeteksi pergeseran distribusi antara data pelatihan dan penerapan, sementara metode deteksi anomali (misalnya, *SVM satu kelas*, *faktor outlier lokal*) mengidentifikasi input baru atau abnormal. Pengamanan ini meningkatkan ketahanan sistem terhadap serangan musuh dan pergeseran data dengan menandai input yang menyimpang dari pola yang diharapkan.
- **Kegagalan Aman:** Strategi kegagalan aman memastikan bahwa jika sistem AI gagal, kegagalan tersebut terjadi secara terkontrol dan tidak menimbulkan bencana. Teknik analisis kelangsungan hidup dan bahaya (misalnya, *model bahaya proporsional Cox*) memperkirakan umur sistem dan mengidentifikasi fitur berisiko tinggi. Redundansi analitik – seperti *redundansi quad-modular* – menggunakan beberapa model beragam yang diatur oleh mekanisme pemungutan suara multi-tingkat untuk meminimalkan kesalahan negatif. Pendekatan ensemble ini meningkatkan keandalan dengan mengurangi bias model individual dan memungkinkan keputusan cadangan ketika ketidakpastian tinggi.

Untuk menunjukkan kelayakan pendekatan ini, para penulis menerapkannya pada sistem prognostik dan manajemen kesehatan (PHM) untuk mesin pesawat menggunakan dataset CMAPSS NASA. Arsitektur sistem mencakup komponen untuk pra-pemrosesan data, pemeriksaan input, analisis kelangsungan hidup, dan pemodelan prediktif. Eksperimen menunjukkan bahwa strategi pemungutan suara multi-tingkat secara signifikan mengurangi kesalahan negatif, dengan pemilih akhir mencapai tingkat kesalahan kurang dari 1% dalam mengidentifikasi kegagalan mesin. Metodologi berbasis komponen ini menawarkan cetak biru praktis untuk merancang sistem AI yang memprioritaskan keselamatan, terutama di domain di mana kegagalan dapat memiliki konsekuensi serius. Dengan mengintegrasikan prinsip-prinsip rekayasa perangkat lunak dengan analitik prediktif, metodologi ini melengkapi strategi siklus hidup dan tata kelola yang lebih luas yang dibahas di seluruh bab ini dan memperkuat pentingnya ketahanan arsitektur dalam keselamatan AI.

15.9 RINGKASAN

Keamanan sistem AI adalah upaya multidisiplin yang membutuhkan upaya berkelanjutan, terkoordinasi, dan adaptif di seluruh siklus hidup teknologi AI. Seiring sistem AI menjadi lebih otonom, berbasis data, dan tertanam dalam domain kritis, potensi konsekuensi yang tidak diinginkan – mulai dari keputusan yang bias dan pelanggaran privasi hingga perilaku yang muncul dan pelanggaran keamanan – meningkat secara signifikan. Mengatasi risiko ini membutuhkan ekosistem keamanan komprehensif yang mengintegrasikan perlindungan teknis, prinsip etika, pengawasan regulasi, dan tata kelola kolaboratif. Bab ini dimulai dengan menetapkan *dimensi dasar keamanan AI*, termasuk keandalan operasional, keamanan, praktik data etis, integrasi siklus hidup, dan kolaborasi interdisipliner. Aspek-aspek ini dilengkapi dengan empat prinsip panduan – *keselarasan, ketahanan, transparansi, dan akuntabilitas* – yang menyediakan kerangka kerja normatif untuk merancang dan mengatur sistem AI secara bertanggung jawab.

Kemudian kami mengeksplorasi tantangan *ketidakselarasan AI*, menyoroti bagaimana *penyimpangan tujuan, permainan spesifikasi, perilaku yang muncul, dan kurangnya interpretasi* dapat menyebabkan hasil yang menyimpang dari nilai-nilai manusia. Mengatasi tantangan ini membutuhkan kombinasi strategi teknis, seperti pembelajaran preferensi dan evaluasi yang kuat, serta mekanisme tata kelola yang menanamkan pengawasan etis ke dalam desain sistem. Untuk mendukung pengembangan AI yang aman, pemerintah di seluruh dunia telah memperkenalkan kerangka kerja regulasi seperti *Undang-Undang AI Uni Eropa, Undang-Undang Inisiatif AI Nasional AS, dan Undang-Undang Akuntabilitas Algoritma*. Upaya ini dilengkapi dengan kerangka kerja sukarela dari lembaga dan organisasi sektor swasta, termasuk *Kerangka Kerja Manajemen Risiko AI NIST, Desain yang Selaras Secara Etis IEEE, dan Kerangka Kerja AI Aman Google*. Bersama-sama, inisiatif ini memberikan panduan hukum, etis, dan operasional untuk membangun sistem AI yang dapat dipercaya. Kami juga menguraikan praktik terbaik untuk memastikan keamanan AI dalam praktik. Ini termasuk mengadopsi siklus hidup pengembangan yang aman, melindungi privasi melalui anonimisasi data, menerapkan pemantauan berkelanjutan dan umpan balik, mendorong kolaborasi interdisipliner,

menyiapkan rencana respons insiden, melakukan audit rutin, dan mendidik pengguna. Jika diterapkan secara konsisten, praktik-praktik ini membantu mengoperasionalkan prinsip-prinsip keselamatan dan memastikan bahwa sistem AI tetap tangguh, transparan, dan selaras dengan nilai-nilai masyarakat.

Terakhir, kami memperkenalkan metodologi berbasis komponen untuk merancang sistem AI yang kritis terhadap keselamatan, menekankan *desain yang aman*, *margin yang aman*, dan strategi *kegagalan yang aman*. Pendekatan ini mengintegrasikan prinsip-prinsip rekayasa perangkat lunak dengan analitik prediktif untuk meminimalkan kesalahan negatif dan meningkatkan keandalan. Divalidasi melalui studi kasus dunia nyata dalam pemeliharaan prediktif, hal ini menunjukkan bagaimana ketahanan arsitektur dapat melengkapi strategi siklus hidup dan tata kelola untuk memastikan keselamatan bahkan dalam skenario kegagalan. Singkatnya, keselamatan sistem AI bukanlah tujuan statis tetapi proses dinamis – yang harus berkembang seiring dengan kemajuan teknologi dan harapan masyarakat. Dengan menanamkan keselamatan ke dalam setiap fase pengembangan dan penerapan AI, kita dapat memanfaatkan potensi transformatif AI sambil melindungi diri dari risikonya, memastikan bahwa sistem ini melayani kepentingan publik dan bermanfaat bagi umat manusia.

15.10 PERTANYAAN

1. Definisikan keamanan sistem AI dan jelaskan signifikansinya di berbagai domain berdampak tinggi seperti perawatan kesehatan, transportasi, dan keuangan.
2. Sebutkan dan jelaskan secara singkat aspek-aspek kunci keamanan sistem AI yang dibahas dalam bab ini.
3. Jelaskan empat prinsip keamanan AI – Keselarasan, Ketahanan, Transparansi, dan Akuntabilitas – dan jelaskan bagaimana prinsip-prinsip tersebut saling berinteraksi.
4. Ringkas tantangan utama terhadap keamanan AI yang diidentifikasi dalam bab ini dan berikan satu strategi mitigasi untuk setiap tantangan.
5. Bandingkan dan kontraskan pendekatan regulasi yang disorot dalam bab ini, termasuk Undang-Undang AI Uni Eropa, Pedoman AI Terpercaya Uni Eropa, Undang-Undang Inisiatif AI Nasional AS, dan Undang-Undang Akuntabilitas Algoritma.
6. Prinsip mana yang secara langsung memastikan bahwa mekanisme telah tersedia untuk menetapkan tanggung jawab atas hasil AI?
 - A. Keselarasan
 - B. Transparansi
 - C. Ketahanan
 - D. Akuntabilitas
7. Kerangka kerja mana yang menekankan penilaian risiko berkelanjutan dan ketahanan untuk operasi AI?
 - A. Kerangka Kerja AI Aman Google
 - B. Kerangka Kerja Manajemen Risiko AI NIST
 - C. Desain yang Selaras Secara Etis IEEE
 - D. Undang-Undang Akuntabilitas Algoritma AS

8. Sistem yang dilatih pada data yang bias dan merugikan kelompok yang dilindungi terutama melanggar aspek keamanan yang mana, dan strategi mitigasi mana yang paling selaras dengan rekomendasi dari bab ini?
- A. Keamanan; rotasi kunci
 - B. Transparansi; kode sumber terbuka
 - C. Keselarasan & keadilan; kumpulkan data yang beragam dan terapkan pemodelan yang menghilangkan bias/sadar keadilan
 - D. Ketahanan; tambahkan lebih banyak redundansi saja

BAB 16

STRATEGI PENGUJIAN UNTUK APLIKASI AI

16.1 PENGUJIAN APLIKASI AI

Seiring semakin terintegrasinya sistem AI ke dalam domain-domain penting seperti perawatan kesehatan, keuangan, transportasi, dan hukum, memastikan keandalan dan keamanannya melalui pengujian yang ketat menjadi sangat penting. Tidak seperti sistem perangkat lunak tradisional, aplikasi AI dicirikan oleh ketergantungannya pada data, perilaku probabilistik, dan potensi keluaran nondeterministik. Karakteristik ini memperkenalkan mode kegagalan dan kerentanan baru yang memerlukan peninjauan kembali strategi pengujian konvensional. Pengujian aplikasi AI melibatkan pendekatan multifaset yang mencakup beberapa dimensi: memvalidasi kualitas dan keadilan data pelatihan, mengevaluasi kinerja model di berbagai skenario, mendeteksi dan mengurangi bias, dan menilai ketahanan terhadap masukan yang merugikan. Selain itu, sistem AI harus terus dipantau untuk mendeteksi pergeseran kinerja dan beradaptasi dengan distribusi data yang berkembang. Strategi pengujian juga harus mempertimbangkan implikasi etis dari perilaku AI, memastikan bahwa keluaran tidak hanya akurat tetapi juga adil, transparan, dan selaras dengan harapan pengguna. Kemajuan terbaru dalam pengujian AI mencakup penggunaan *pengujian diferensial* dan *pengujian mutasi* untuk meningkatkan keragaman dan efektivitas dataset pengujian. Pengujian diferensial, seperti yang diterapkan pada alat seperti DeepXplore, menghasilkan kasus uji dengan memanfaatkan perbedaan batas keputusan di seluruh model. Pengujian mutasi, seperti yang digunakan dalam DeepMutation, memperkenalkan kesalahan terkontrol untuk mengevaluasi kualitas data uji berdasarkan seberapa baik data tersebut mendeteksi kesalahan ini. Bersama-sama, teknik-teknik ini meningkatkan skor mutasi – metrik untuk kualitas data uji – dan membantu mengungkap perilaku kasus ekstrem dalam sistem pembelajaran mendalam.

Secara paralel, *AI generatif* membentuk kembali pengujian perangkat lunak dengan mengotomatiskan tugas-tugas seperti pembuatan kasus uji, simulasi skenario, dan deteksi anomali. Alat yang didukung oleh model bahasa besar (LLM) dapat menganalisis spesifikasi dan kode untuk menghasilkan pengujian unit, mensimulasikan beragam perilaku pengguna, dan memantau sistem secara real-time. Kemampuan ini mempercepat alur kerja pengujian dan memperluas cakupan, tetapi juga memperkenalkan risiko baru, termasuk keluaran yang dihalusinasi, perilaku nondeterministik, dan masalah etika. Bab ini menguraikan strategi pengujian utama yang disesuaikan dengan sistem AI, termasuk pengujian unit dan integrasi, pengujian interpretasi (TI), pengujian kinerja dan beban, pengujian keamanan, serta verifikasi dan validasi model. Bab ini juga mengeksplorasi teknik-teknik canggih seperti pengujian adversarial, evaluasi dengan campur tangan manusia, dan pengujian berbasis skenario. Dengan menggabungkan prinsip-prinsip pengujian perangkat lunak tradisional dengan

metodologi khusus AI dan inovasi yang muncul, organisasi dapat membangun kerangka kerja pengujian yang kuat yang mendukung penerapan teknologi AI yang aman, etis, dan efektif.

16.2 STRATEGI UNTUK MENGUJI APLIKASI AI

Pengujian aplikasi AI membutuhkan pendekatan multifaset yang menggabungkan prinsip-prinsip pengujian perangkat lunak tradisional dengan teknik khusus yang disesuaikan dengan karakteristik unik sistem AI. Tidak seperti perangkat lunak konvensional, aplikasi AI digerakkan oleh data, probabilistik, dan seringkali nondeterministik, sehingga rentan terhadap masalah seperti bias, pergeseran, kerentanan adversarial, dan perilaku yang tidak dapat diprediksi. Oleh karena itu, strategi pengujian yang efektif harus mengatasi keandalan teknis dan integritas etika.



Gambar 16.1 Kerangka kerja untuk merancang strategi pengujian untuk aplikasi AI.

Kerangka kerja pengujian yang komprehensif harus mencakup beberapa dimensi: pengujian fungsional dan non-fungsional, verifikasi dan validasi model, penilaian bias dan keadilan, ketahanan adversarial, dan validasi data. Strategi ini juga harus mempertimbangkan evolusi berkelanjutan model AI dan sifat dinamis data dunia nyata. Gambar 16.1 menyediakan kerangka kerja yang dapat digunakan sebagai panduan untuk merancang strategi pengujian untuk aplikasi AI. *Pengujian pengguna akhir* memainkan peran penting dalam memvalidasi

apakah aplikasi AI memenuhi kebutuhan spesifik domain dan tujuan bisnis. Pengujian Pakar Domain (Domain Expert Testing/DET) memastikan bahwa sistem tersebut praktis dan relevan untuk penggunaan yang dimaksudkan, sementara Pengujian Penerimaan Pengguna (User Acceptance Testing/UAT) mengevaluasi kegunaan, fungsionalitas, dan keselarasan dengan tujuan pemangku kepentingan.

Pengujian non-fungsional meliputi pengujian kinerja, beban, stres, dan keamanan. Tes-tes ini menilai seberapa baik sistem menangani latensi, skalabilitas, throughput, pemanfaatan sumber daya, dan ketahanan dalam kondisi ekstrem. Pengujian keamanan mengevaluasi kerentanan dalam penanganan data, integritas algoritma, kontrol akses, dan kepatuhan terhadap standar seperti GDPR dan HIPAA. *Pengujian fungsional* berfokus pada validasi komponen individual seperti pra-pemrosesan data, rekayasa fitur, pelatihan model, inferensi, dan fungsi utilitas. TI melengkapi hal ini dengan memastikan bahwa keputusan model dapat dijelaskan dan dilacak. Teknik seperti Local Interpretable Model-agnostic Explanations (LIME), SHapley Additive exPlanations (SHAP), Partial Dependence Plots (PDP), dan Individual Conditional Expectation (ICE) membantu mengungkap alasan di balik prediksi, yang penting untuk membangun kepercayaan dan memenuhi persyaratan peraturan.

Verifikasi dan validasi model melibatkan audit data pelatihan dan evaluasi kinerja model di berbagai kelompok demografis menggunakan metrik keadilan seperti paritas demografis, kesempatan yang sama, dan rasio dampak yang berbeda. Tes ini membantu mengidentifikasi dan mengurangi perilaku diskriminatif. *Pengujian adversarial* adalah strategi proaktif yang memperkenalkan input berbahaya atau menipu untuk mengevaluasi ketahanan sistem. Alat seperti CleverHans dan Foolbox mensimulasikan serangan untuk mengungkap kerentanan dan menilai seberapa baik model tersebut mampu bertahan dalam kondisi ekstrem. Inovasi terbaru dalam pengujian AI mencakup *pengujian diferensial* dan *pengujian mutasi*, yang meningkatkan keragaman dan kualitas dataset pengujian. Pengujian diferensial, seperti yang diterapkan dalam DeepXplore, menghasilkan kasus uji dengan memanfaatkan perbedaan batas keputusan di seluruh model. Pengujian mutasi, seperti yang digunakan dalam DeepMutation, memperkenalkan kesalahan terkontrol untuk mengevaluasi seberapa baik data uji mendeteksi kesalahan ini. Bersama-sama, teknik-teknik ini meningkatkan skor mutasi – metrik untuk kualitas data uji – dan membantu mengungkap perilaku kasus ekstrem dalam sistem pembelajaran mendalam.

Secara paralel, *AI generatif* mentransformasi pengujian perangkat lunak dengan mengotomatiskan tugas-tugas seperti pembuatan kasus uji, simulasi skenario, dan deteksi anomali. Alat yang didukung oleh LLM dapat menganalisis spesifikasi dan kode untuk menghasilkan pengujian unit, mensimulasikan beragam perilaku pengguna, dan memantau sistem secara real-time. Kemampuan ini mempercepat alur kerja pengujian dan memperluas cakupan, tetapi juga memperkenalkan risiko baru, termasuk keluaran halusinasi, perilaku nondeterministik, dan masalah etika. Untuk mengatasi tantangan ini, strategi pengujian harus berevolusi. Hal ini mencakup penerapan pemantauan berkelanjutan, mendorong literasi AI di antara tim penjaminan mutu (QA), mendorong kolaborasi lintas fungsi, dan meningkatkan perangkat dengan kemampuan pengujian yang ditingkatkan AI. Di bidang-bidang berisiko

tinggi seperti perawatan kesehatan dan pertahanan, pengawasan manusia yang ketat, keterlacakan, dan pengujian stres berbasis skenario sangat penting untuk memastikan keandalan dan akuntabilitas. Pengujian aplikasi AI membutuhkan pendekatan berlapis dan adaptif. Dengan mengintegrasikan praktik QA tradisional dengan teknik khusus AI yang baru muncul, organisasi dapat membangun kerangka kerja pengujian yang kuat yang memastikan sistem AI mereka tidak hanya secara teknis baik tetapi juga bertanggung jawab secara etis dan tangguh di lingkungan dunia nyata.

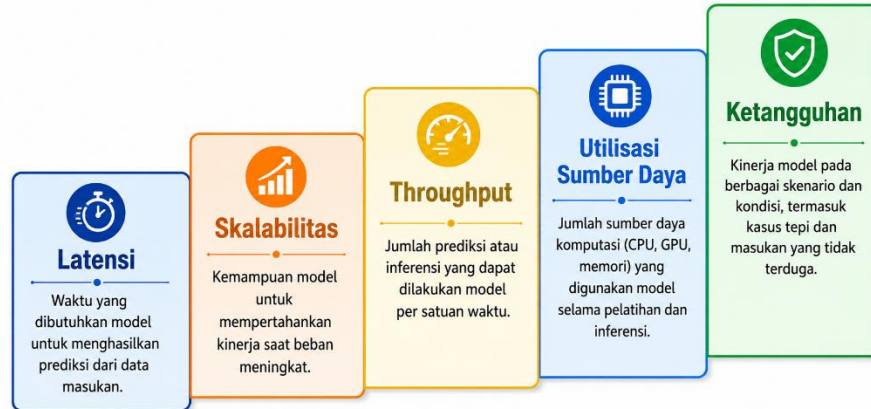
16.3 PENGUJIAN PENGGUNA AKHIR

Pengujian pengguna akhir adalah fase terakhir dan seringkali paling penting dari validasi sistem AI. Fokusnya adalah memastikan bahwa aplikasi memenuhi harapan pengguna yang dituju dan berkinerja andal dalam konteks operasionalnya. Tidak seperti evaluasi teknis murni, pengujian pengguna akhir menggabungkan penilaian manusia, keahlian domain, dan umpan balik dunia nyata untuk menilai kegunaan, fungsionalitas, dan relevansi. Pengujian ini biasanya mencakup dua pendekatan komplementer: DET dan UAT. Dalam DET, pakar bidang studi mengevaluasi apakah sistem AI secara efektif mengatasi tantangan spesifik domain. Wawasan mereka sangat penting untuk memvalidasi kasus penggunaan, mengidentifikasi skenario ekstrem, dan memastikan bahwa keluaran model praktis dan dapat diinterpretasikan dalam domain tersebut. Misalnya, dalam perawatan kesehatan, dokter dapat menilai apakah sistem AI diagnostik memberikan rekomendasi yang dapat ditindaklanjuti dan secara klinis tepat. UAT, di sisi lain, melibatkan pengguna representatif yang berinteraksi dengan aplikasi AI untuk menilai kegunaan, responsivitas, dan keselarasan dengan tujuan bisnis. Fase ini membantu mengungkap masalah yang mungkin tidak muncul selama pengujian teknis – seperti antarmuka yang membingungkan, keluaran yang menyesatkan, atau harapan yang tidak terpenuhi.

Pengujian pengguna akhir modern semakin didukung oleh *alat AI generatif*, yang dapat mensimulasikan beragam persona dan perilaku pengguna. Misalnya, LLM dapat meniru pengguna pemula, ahli, atau bahkan aktor antagonis untuk menguji bagaimana sistem merespons berbagai masukan dan pola penggunaan. Simulasi ini membantu mengungkap titik buta dan meningkatkan cakupan kasus-kasus ekstrem yang sulit direproduksi secara manual. Selain itu, *pengujian dengan melibatkan manusia* – di mana peninjau manusia menilai keluaran yang dihasilkan AI – memainkan peran penting dalam memvalidasi interpretasi, keselarasan etis, dan keterlacakan keputusan. Hal ini sangat penting di domain yang berisiko tinggi, di mana bahkan kesalahan kecil pun dapat memiliki konsekuensi serius. Pengujian pengguna akhir menjembatani kesenjangan antara validasi teknis dan penerapan di dunia nyata. Dengan menggabungkan keahlian domain, umpan balik pengguna, dan simulasi yang ditingkatkan AI, hal ini memastikan bahwa sistem AI tidak hanya fungsional tetapi juga dapat digunakan, dapat dipercaya, dan selaras dengan harapan pemangku kepentingan.

16.4 PENGUJIAN NON-FUNGSIONAL

Pengujian non-fungsional mengevaluasi seberapa baik kinerja aplikasi AI dalam berbagai kondisi operasional, dengan fokus pada atribut seperti efisiensi, skalabilitas, ketahanan, dan keamanan. Pengujian ini penting untuk memastikan bahwa sistem AI tidak hanya menghasilkan output yang akurat tetapi juga beroperasi dengan andal di lingkungan produksi, terutama di bawah tekanan atau di hadapan kondisi yang merugikan.



Gambar 16.2 Aspek pengujian kinerja aplikasi AI.

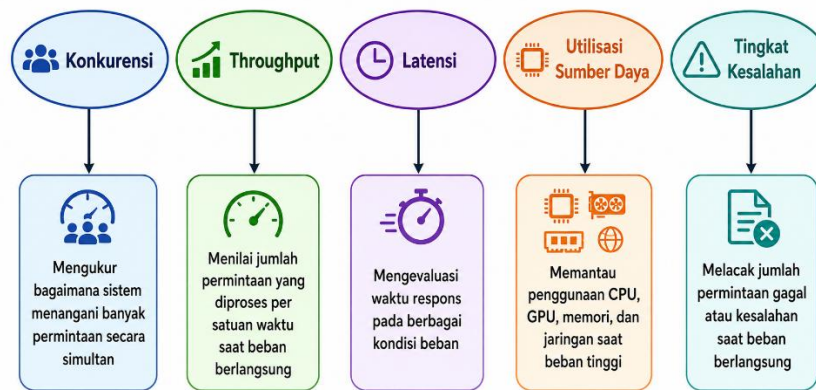
Kategori pengujian ini mencakup empat dimensi utama: *kinerja*, *beban*, *tekanan*, dan *keamanan*.

- *Pengujian Kinerja* menilai latensi, throughput, skalabilitas, pemanfaatan sumber daya, dan ketahanan. Ini memastikan bahwa sistem AI dapat menghasilkan prediksi dalam jangka waktu yang dapat diterima, berskala dengan peningkatan beban kerja, dan mempertahankan kinerja yang konsisten di seluruh kasus ekstrem. Gambar 16.2 menyoroti aspek-aspek utama pengujian kinerja ini.

Latensi memperhitungkan waktu yang dibutuhkan model untuk menghasilkan prediksi. *Skalabilitas* memperhitungkan kemampuan model untuk mempertahankan kinerja seiring meningkatnya beban kerja. *Throughput* memperhitungkan jumlah prediksi atau inferensi yang dapat dibuat model dalam satuan waktu. *Pemanfaatan sumber daya* berfokus pada jumlah sumber daya komputasi yang digunakan selama fase pelatihan dan inferensi model. Terakhir, *ketahanan* berfokus pada kinerja model dalam skenario/kasus penggunaan yang merupakan kasus ekstrem.

Alat AI generatif dapat membantu pengujian kinerja dengan mensimulasikan berbagai skenario penggunaan – termasuk ribuan pengguna virtual atau kegagalan perangkat keras – yang tidak praktis untuk direproduksi secara manual.

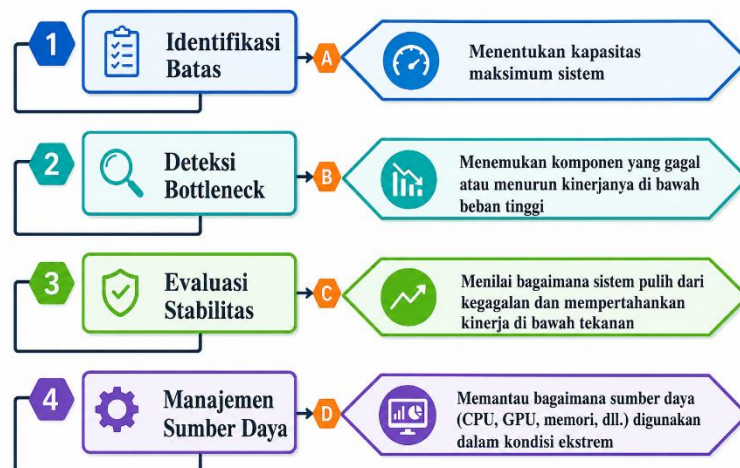
- *Pengujian Beban* mengevaluasi bagaimana sistem berperilaku di bawah beban kerja yang berat. Ini mengidentifikasi hambatan dan memverifikasi bahwa infrastruktur dapat menangani permintaan puncak tanpa degradasi. Gambar 16.3 menyoroti metrik utama untuk pengujian beban yang meliputi konkurensi, throughput, latensi, konsumsi sumber daya di bawah tekanan berkelanjutan, dan tingkat kesalahan.



Gambar 16.3 Aspek-aspek kunci pengujian beban untuk aplikasi AI.

Konkurensi berfokus pada pengukuran bagaimana sistem menangani banyak permintaan yang datang secara bersamaan. *Throughput* menilai jumlah permintaan yang telah diproses oleh aplikasi AI per satuan waktu di bawah beban yang dapat diterima. *Latensi* berfokus pada evaluasi waktu respons aplikasi AI di bawah kondisi beban yang berbeda. *Pemanfaatan sumber daya* berfokus pada pemantauan penggunaan sumber daya (CPU, GPU, memori, dan lain-lain.) oleh aplikasi AI selama situasi beban kerja tinggi. *Tingkat kesalahan* berfokus pada pelacakan jumlah permintaan yang gagal atau kesalahan yang dilakukan oleh aplikasi AI dalam kisaran beban kerja yang dapat diterima.

- *Pengujian Stres* mendorong sistem melampaui batas operasional normalnya untuk mengungkap titik kegagalan dan menilai kemampuan pemulihan. Hal ini sangat penting di bidang-bidang berisiko tinggi seperti perawatan kesehatan dan pertahanan, di mana kegagalan bukanlah pilihan. Gambar 16.4 menyoroti aspek-aspek kunci yang perlu dipertimbangkan saat melakukan pengujian stres aplikasi AI.



Gambar 16.4 Aspek-aspek kunci pengujian stres untuk aplikasi AI.

Mengidentifikasi batasan berfokus pada penentuan kapasitas maksimum yang dapat ditangani oleh aplikasi AI sebelum memasuki keadaan gagal. *Mendeteksi hambatan* sangat penting dalam menemukan komponen yang menunjukkan penurunan kinerja atau gagal di bawah beban berat. *Evaluasi stabilitas* menilai bagaimana sistem/aplikasi pulih dari kegagalan dan mempertahankan kinerjanya di bawah berbagai skenario tekanan. *Manajemen sumber daya* memantau bagaimana sumber daya sistem (CPU, GPU, memori, dan lain-lain.) digunakan oleh aplikasi AI dalam kondisi ekstrem. AI Generatif dapat membantu mensimulasikan kondisi langka atau ekstrem untuk mengungkap kelemahan, sementara pengawasan manusia memastikan bahwa kasus-kasus kritis tidak terabaikan.

- *Pengujian Keamanan* berfokus pada perlindungan data sensitif, algoritma, dan komponen sistem dari ancaman. Ini mencakup penilaian keamanan data, integritas algoritma, kontrol akses, deteksi ancaman, dan kepatuhan terhadap standar seperti GDPR dan HIPAA. Gambar 16.5 menyoroti aspek-aspek kunci yang perlu dipertimbangkan saat melakukan pengujian keamanan aplikasi AI.

Keamanan data berfokus pada perlindungan data sensitif yang digunakan untuk pelatihan, inferensi, dan penyimpanan. *Keamanan algoritma* memastikan integritas dan kerahasiaan model dan algoritma AI. *Kontrol akses* berfokus pada pembatasan akses aplikasi AI kepada pengguna yang berwenang sehingga mencegah akses yang tidak sah. *Deteksi ancaman* berfokus pada mendeteksi dan mengurangi potensi ancaman dan kerentanan aplikasi AI. *Kepatuhan* berfokus pada memastikan kepatuhan terhadap standar dan peraturan keamanan (GDPR, HIPAA, dan lain-lain.) dari aplikasi AI.

Dalam sistem yang ditingkatkan dengan AI, pengujian keamanan juga harus memperhitungkan risiko yang ditimbulkan oleh model generatif, seperti keluaran halusinasi, pergeseran model, dan pengambilan keputusan yang tidak sah.

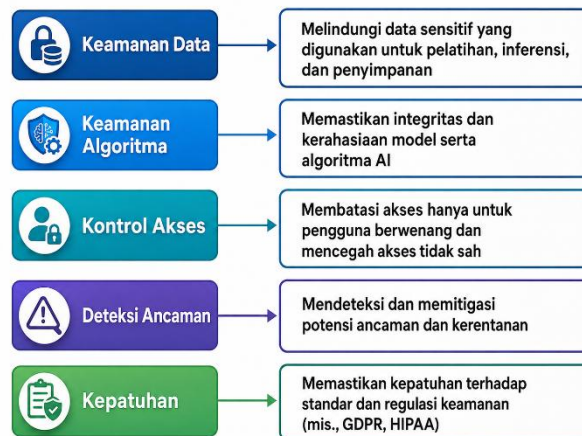
Pengujian non-fungsional modern semakin bergantung pada pemantauan berkelanjutan dan umpan balik. Alat bertenaga AI dapat mengamati sistem produksi secara real-time, mendeteksi anomali, dan memperingatkan tim tentang penurunan kinerja atau risiko yang muncul. Misalnya, platform seperti Amazon DevOps dosen atau Sentry dapat melacak perilaku model pasca-deployment dan memberikan wawasan kembali ke dalam pipeline pengujian. Pendekatan adaptif ini memastikan bahwa sistem AI tetap andal dalam jangka waktu lama setelah validasi awal. Pengujian non-fungsional sangat penting untuk memvalidasi ketahanan operasional aplikasi AI. Dengan menggabungkan metrik kinerja tradisional dengan simulasi dan pemantauan yang ditingkatkan AI, organisasi dapat memastikan bahwa sistem mereka tidak hanya akurat tetapi juga kuat, aman, dan siap untuk diterapkan di dunia nyata.

16.5 PENGUJIAN FUNGSIONAL

Pengujian fungsional memastikan bahwa setiap komponen aplikasi AI menjalankan tugas yang dimaksudkan dengan benar dan andal. Tidak seperti perangkat lunak tradisional, sistem AI memperkenalkan kompleksitas tambahan karena ketergantungan data, sifat probabilistik, dan perilaku yang berkembang. Oleh karena itu, pengujian fungsional harus

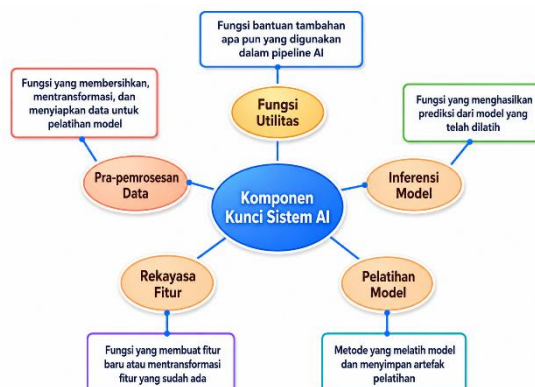
diadaptasi untuk mengatasi karakteristik unik ini. Pengujian fungsional untuk sistem AI secara umum dapat dikategorikan menjadi *pengujian unit* (UT) dan IT.

- UT dalam aplikasi AI melampaui verifikasi fungsi-fungsi terisolasi. Ini melibatkan pengujian komponen-komponen seperti pra-pemrosesan data, rekayasa fitur, pelatihan model, inferensi, dan fungsi utilitas. Setiap modul ini harus divalidasi secara independen untuk memastikan kebenaran dan keandalannya. Misalnya, fungsi pra-pemrosesan harus menangani nilai yang hilang dan outlier dengan benar, sementara rutinitas pelatihan model harus menghasilkan model yang dapat direproduksi dan akurat.



Gambar 16.5 Aspek-aspek kunci pengujian keamanan untuk aplikasi AI.

Gambar 16.6 menyoroti komponen-komponen kunci yang perlu difokuskan selama UT sistem/aplikasi AI. *Pra-pemrosesan data* menguji fungsi-fungsi yang mendukung pembersihan, transformasi, dan persiapan data untuk pelatihan model. *Rekayasa fitur* menilai fungsi-fungsi yang membuat fitur baru atau mentransformasi fitur yang ada. Komponen *pelatihan model* mengevaluasi metode-metode yang mendukung pelatihan model dan menyimpan artefak pelatihan model. *Inferensi model* menguji fungsi-fungsi yang menghasilkan prediksi dari model yang telah dilatih.



Gambar 16.6 Komponen UT untuk sistem/aplikasi AI.

Terakhir, komponen *fungsi utilitas* mengevaluasi fungsi pembantu yang digunakan dalam pipeline ML/AI. Alat AI generatif semakin banyak digunakan untuk mengotomatisasi pembuatan unit test. Alat seperti Diffblue Cover dan CodiumAI dapat menganalisis kode sumber dan spesifikasi untuk menghasilkan kasus uji yang mencakup perilaku yang diharapkan, mengurangi upaya manual dan memperluas cakupan pengujian. Alat-alat ini juga dapat diintegrasikan dengan pipeline CI/CD untuk memungkinkan pengujian dan umpan balik berkelanjutan.

- TI sangat penting untuk memahami dan menjelaskan keputusan yang dibuat oleh model AI. Dalam domain berisiko tinggi seperti perawatan kesehatan, keuangan, dan hukum, interpretasi sangat penting untuk membangun kepercayaan, memenuhi persyaratan peraturan, dan men-debug perilaku model. Teknik interpretasi umum meliputi (Gambar 16. Teknik interpretasi umum meliputi (Gambar 16.7):
 - **Pentingnya Fitur:** Mengidentifikasi fitur input mana yang paling memengaruhi prediksi model.
 - **PDP dan ICE:** Memvisualisasikan bagaimana perubahan fitur input memengaruhi prediksi.
 - **LIME:** Menjelaskan prediksi individual dengan mendekati model secara lokal.
 - **SHAP:** Menggunakan teori permainan kooperatif untuk memberikan skor kepentingan yang adil dan konsisten pada fitur-fitur.



Gambar 16.7 Teknik interpretasi untuk pengujian aplikasi AI.

Pengujian fungsional juga harus memperhitungkan *perilaku nondeterministik*, ciri umum dalam sistem AI. Tidak seperti perangkat lunak tradisional, model AI dapat menghasilkan keluaran yang sedikit berbeda untuk masukan yang sama karena elemen stokastik dalam pelatihan atau inferensi. Hal ini membuat reproduksibilitas menjadi sulit dan membutuhkan pergeseran dari pengujian lulus/gagal biner ke *validasi berbasis rentang*, di mana keluaran dievaluasi terhadap ambang batas keandalan yang dapat diterima. Pengujian fungsional dalam aplikasi AI membutuhkan kombinasi teknik validasi perangkat lunak tradisional dan strategi khusus AI. Dengan menggabungkan pembuatan pengujian otomatis,

alat interpretasi, dan metode untuk menangani nondeterminisme, organisasi dapat memastikan bahwa sistem AI mereka secara teknis baik dan transparan.

16.6 VERIFIKASI DAN VALIDASI MODEL

Verifikasi dan validasi model sangat penting untuk memastikan bahwa sistem AI berperilaku sesuai tujuan, menghasilkan hasil yang adil dan dapat dipercaya, serta mematuhi standar etika dan peraturan. Tidak seperti sistem perangkat lunak tradisional, model AI belajar dari data dan dapat menunjukkan perilaku yang tidak dapat diprediksi atau bias jika tidak diuji secara ketat. Verifikasi dan validasi membantu mengkonfirmasi bahwa logika model selaras dengan tujuan desainnya dan bahwa outputnya konsisten, adil, dan dapat dijelaskan. Fokus utama dari proses ini adalah *pengujian bias dan keadilan*, yang melibatkan identifikasi dan mitigasi kesalahan sistematis yang dapat menyebabkan hasil yang diskriminatif. Hal ini membutuhkan kombinasi analisis statistik, metrik keadilan, dan penilaian manusia. Pengujian harus mencakup audit data pelatihan untuk keseimbangan representasional, evaluasi kinerja model di berbagai subkelompok demografis, dan penerapan algoritma keadilan dan alat interpretasi. Gambar 16.8 menyoroti metrik keadilan umum yang dapat digunakan untuk mengevaluasi bias dan keadilan dalam model AI yang meliputi:

- **Kesetaraan Demografis:** Memastikan tingkat hasil positif yang sama di berbagai kelompok.



Gambar 16.8 Metrik bias dan keadilan untuk model AI.

- **Kesempatan yang Sama:** Memverifikasi bahwa tingkat positif sejati sama di seluruh kelompok.
- **Peluang yang Sama:** Memeriksa tingkat positif sejati dan positif palsu untuk kesamaan.
- **Rasio Dampak yang Berbeda:** Mengukur rasio hasil positif antara kelompok yang paling dan paling tidak diuntungkan.

Metrik ini membantu mengukur keadilan dan memandu strategi perbaikan. Namun, mencapai keadilan seringkali melibatkan pertukaran dengan akurasi dan tujuan kinerja lainnya, yang membutuhkan pertimbangan etis yang cermat dan masukan dari pemangku kepentingan. Alat interpretasi seperti SHAP, LIME, dan PDP juga memainkan peran penting dalam validasi model. Alat-alat ini membantu menjelaskan mengapa suatu model membuat keputusan tertentu,

yang sangat penting untuk debugging, kepatuhan terhadap peraturan, dan membangun kepercayaan pengguna – terutama di bidang-bidang yang berisiko tinggi seperti perawatan kesehatan, keuangan, dan hukum.

Selain itu, validasi harus memperhitungkan *perilaku nondeterministik*, di mana model dapat menghasilkan output yang sedikit berbeda untuk input yang sama karena elemen stokastik dalam pelatihan atau inferensi. Hal ini membutuhkan pergeseran dari pengujian lulus/gagal biner ke *validasi berbasis rentang*, di mana output dievaluasi terhadap ambang batas keandalan yang dapat diterima. Alat AI generatif dapat membantu dalam proses ini dengan mensimulasikan berbagai skenario dan menghasilkan kasus uji sintetis untuk mengevaluasi perilaku model dalam berbagai kondisi. Namun, pengawasan manusia tetap penting untuk memverifikasi bahwa pengujian yang dihasilkan AI bermakna dan bahwa output model secara etis dapat diterima. Verifikasi dan validasi model memastikan bahwa sistem AI tidak hanya akurat secara teknis tetapi juga adil, transparan, dan selaras dengan nilai-nilai masyarakat. Dengan menggabungkan ketelitian statistik, interpretasi, dan pengawasan etika, organisasi dapat membangun aplikasi AI yang dapat dipercaya dan tangguh di lingkungan dunia nyata.

16.7 PENGUJIAN ADVERSARIAL

Pengujian adversarial adalah strategi proaktif yang digunakan untuk mengevaluasi kekokohan dan keamanan sistem AI dengan sengaja memasukkan input yang berbahaya, menipu, atau tidak terduga. Tujuannya adalah untuk mengungkap kerentanan, menilai seberapa baik model tersebut mampu bertahan dalam kondisi ekstrem, dan memastikan bahwa model tersebut berperilaku andal bahkan di bawah tekanan adversarial. Tidak seperti pengujian tradisional, yang berfokus pada input dan output yang diharapkan, pengujian adversarial berupaya untuk "merusak" sistem dengan mensimulasikan kasus-kasus ekstrem, data yang salah format, atau serangan adversarial. Pendekatan ini sangat penting di domain berisiko tinggi seperti perawatan kesehatan, keuangan, dan pertahanan, di mana kegagalan dapat memiliki konsekuensi serius.

Pengembang dan penguji menggunakan pengujian adversarial untuk mensimulasikan skenario di mana sistem AI mungkin tertipu untuk menghasilkan output yang salah atau tidak aman. Serangan ini dapat mengungkap bias tersembunyi, mengungkapkan batas keputusan yang rapuh, dan mengidentifikasi kelemahan yang dapat dieksploitasi yang mungkin terlewatkan oleh pengujian standar. Misalnya, model yang dilatih untuk mengklasifikasikan gambar mungkin salah mengklasifikasikan input yang sedikit terganggu, atau chatbot mungkin merespons secara tidak tepat terhadap perintah yang dirancang dengan cerdas. Alat AI generatif dapat memperkuat pengujian adversarial dengan mensimulasikan beragam persona dan perilaku – termasuk aktor adversarial – untuk menyelidiki respons sistem. Alat-alat ini dapat menghasilkan input sintetis yang meniru serangan dunia nyata atau kasus-kasus ekstrem, membantu penguji mengevaluasi ketahanan sistem di berbagai kondisi yang lebih luas.

Dalam praktiknya, pengujian adversarial melibatkan pembuatan input yang mengeksploitasi kelemahan yang diketahui dalam arsitektur model atau data pelatihan.

Pustaka berbasis Python seperti *CleverHans* dan *Foolbox* mendukung pengujian adversarial dengan menyediakan implementasi algoritma serangan dan alat benchmarking. Pustaka-pustaka ini memungkinkan penguji untuk mengevaluasi ketahanan model terhadap berbagai skenario adversarial, termasuk penghindaran, peracunan, dan ekstraksi model. Pengujian adversarial juga memainkan peran kunci dalam memvalidasi perilaku etis sistem AI. Misalnya, pengujian adversarial dapat membantu memastikan bahwa model tidak secara percaya diri menghasilkan output yang dihalusinasi atau membuat keputusan ketika tidak yakin. Dalam lingkungan yang teregulasi, pengujian adversarial mendukung kepatuhan dengan menunjukkan bahwa sistem dapat menangani batasan kebijakan dan menghindari tindakan yang tidak sah. Pengujian adversarial memperkuat keandalan dan kepercayaan sistem AI dengan mengungkap kerentanan sebelum dapat dieksploitasi. Ketika dikombinasikan dengan pengawasan manusia dan evaluasi berbasis skenario, pengujian ini menjadi alat yang ampuh untuk membangun aplikasi AI yang tangguh, aman, dan beretika.

16.8 VALIDASI DATA

Validasi data adalah aspek mendasar dari pengujian aplikasi AI. Karena sistem AI pada dasarnya digerakkan oleh data, kualitas, integritas, dan keterwakilan data masukan secara langsung memengaruhi kinerja, keadilan, dan keandalan model. Kualitas data yang buruk dapat menyebabkan prediksi yang bias, akurasi yang berkurang, dan kerentanan yang membahayakan kepercayaan sistem. Strategi validasi data untuk sistem AI biasanya terbagi menjadi dua kategori: *pengujian kualitas data* dan *pengujian integrasi data*.

- **Pengujian Kualitas Data** berfokus pada memastikan bahwa dataset pelatihan dan pengujian lengkap, akurat, dan bebas dari inkonsistensi, outlier, atau nilai yang hilang. Ini juga melibatkan pemeriksaan bias dan memastikan representasi yang memadai di seluruh kelompok demografis (misalnya, usia, jenis kelamin, ras, departemen, dan kategori). Alat visual seperti histogram, plot sebaran, dan kurva distribusi umumnya digunakan untuk mengungkap masalah dan memvalidasi distribusi data.

Selain metode tradisional, *pengujian mutasi* menawarkan teknik yang ampuh untuk mengevaluasi efektivitas data uji. Pengujian mutasi memperkenalkan kesalahan terkontrol ke dalam model dan menilai apakah data uji dapat mendeteksi kesalahan ini. *Skor mutasi* – rasio mutan yang terbunuh terhadap total mutan – berfungsi sebagai metrik untuk kualitas data uji. Skor mutasi yang lebih tinggi menunjukkan cakupan dan keragaman yang lebih baik dalam dataset uji.

Untuk meningkatkan skor mutasi, *pengujian diferensial* dapat digunakan untuk menghasilkan kasus uji tambahan yang menargetkan perilaku kasus ekstrem. Alat seperti DeepXplore memanfaatkan perbedaan batas keputusan di berbagai model untuk menciptakan input yang beragam dan menantang. Ketika dikombinasikan dengan pengujian mutasi (seperti pada DeepMutation), pendekatan ini meningkatkan kekokohan proses validasi dan membantu mengungkap kerentanan tersembunyi.

- **Pengujian Integrasi Data** memastikan bahwa data dari berbagai sumber digabungkan, ditransformasikan, dan dibuat konsisten secara akurat untuk pelatihan dan inferensi

model. Ini termasuk memvalidasi sumber data individual, memeriksa kesesuaian skema, dan memverifikasi bahwa dataset terintegrasi mempertahankan struktur dan tipe data yang benar.

Validasi data bukan hanya tentang pembersihan dan pemformatan – tetapi tentang memastikan bahwa data mencerminkan kompleksitas dan keragaman skenario dunia nyata. Dengan menggabungkan teknik validasi tradisional dengan metode canggih seperti pengujian mutasi dan diferensial, organisasi dapat membangun sistem AI yang lebih kuat, adil, dan tahan terhadap risiko terkait data.

16.9 PRAKTIK PENGUJIAN TERBAIK

Pengujian aplikasi AI membutuhkan perpaduan strategis antara prinsip-prinsip QA perangkat lunak tradisional dan metodologi khusus AI. Seiring dengan evolusi sistem AI, praktik pengujian yang memastikan keandalan, keadilan, dan keamanannya juga harus ikut berkembang. Praktik terbaik berikut menawarkan pendekatan komprehensif untuk menguji aplikasi AI secara efektif di lingkungan dunia nyata:

- **Tetapkan Tujuan dan Metrik yang Jelas:** Tetapkan tujuan spesifik dan terukur untuk pengujian sistem AI. Tujuan ini harus mencakup tidak hanya akurasi dan kinerja, tetapi juga keadilan, ketahanan, interpretasi, dan kepatuhan. Metrik seperti skor mutasi, cakupan neuron, dan indikator keadilan (misalnya, kesetaraan demografis, kesempatan yang sama) membantu mengukur efektivitas pengujian dan keselarasan etika.
- **Pastikan Kualitas dan Keragaman Data:** Sistem AI hanya sebaik data yang dipelajarinya. Validasi dataset pelatihan dan pengujian untuk kelengkapan, akurasi, dan keseimbangan representasi. Gunakan teknik seperti *pengujian mutasi* untuk menilai kualitas data pengujian dengan memperkenalkan kesalahan terkontrol dan mengukur seberapa baik data mendeteksinya. Gabungkan ini dengan *pengujian diferensial* untuk menghasilkan kasus uji yang beragam yang mengungkap perilaku kasus ekstrem dan meningkatkan skor mutasi.
- **Manfaatkan AI Generatif untuk Pengujian:** Alat AI generatif – terutama LLM – dapat mengotomatiskan pembuatan kasus uji, mensimulasikan perilaku pengguna, dan memantau sistem untuk anomali. Alat seperti *Diffblue Cover* dan *CodiumAI* menganalisis kode dan spesifikasi untuk menghasilkan pengujian unit, sementara yang lain mensimulasikan persona yang beragam dan kondisi ekstrem untuk memperluas cakupan. Namun, alat-alat ini harus digunakan dengan hati-hati karena risiko seperti output halusinasi dan perilaku nondeterministik.
- **Terapkan Pemantauan Berkelanjutan dan Umpan Balik:** Pengujian tidak boleh berakhir pada saat penerapan. Terapkan pemantauan berkelanjutan untuk mendeteksi penyimpangan kinerja, anomali, dan risiko yang muncul. Platform berbasis AI seperti *Amazon DevOps dosen* dan *Sentry* dapat mengamati sistem produksi dan memberikan wawasan kembali ke dalam alur pengujian, sehingga memungkinkan validasi adaptif dan berkelanjutan.

- **Mengintegrasikan Pengawasan Manusia dan Tinjauan Etika:** Pengujian AI harus mencakup evaluasi yang melibatkan manusia, terutama di bidang-bidang berisiko tinggi seperti perawatan kesehatan, keuangan, dan pertahanan. Peninjau manusia membantu memvalidasi interpretasi, keselarasan etika, dan keterlacakan keputusan. Pengujian stres berbasis skenario dan pemetaan keterlacakan sangat penting untuk memastikan akuntabilitas dan kepatuhan terhadap peraturan.
- **Memupuk Literasi AI dalam Tim QA:** Melengkapi para profesional QA dengan pengetahuan dasar tentang konsep AI, perilaku model, dan mode kegagalan. Memahami rekayasa yang cepat, pergeseran model, dan keluaran probabilistik memungkinkan penguji untuk menggunakan alat AI secara efektif dan mengevaluasi hasil mereka secara kritis.
- **Mendorong Kolaborasi Lintas Fungsi:** Pengujian AI yang efektif membutuhkan kolaborasi antara pengembang, penguji, ilmuwan data, dan pakar bidang. Keterlibatan QA sejak awal dalam fase desain memastikan kemampuan uji dan relevansi. Pakar domain berkontribusi pada skenario realistis, sementara pengembang membangun transparansi dan kait pengujian ke dalam komponen AI.
- **Tingkatkan Perangkat dan Teknik Pengujian:** Tambahkan metode QA tradisional dengan alat dan strategi khusus AI. Perkenalkan pengujian adversarial untuk menyelidiki kerentanan, gunakan analisis statistik untuk keluaran probabilistik, dan terapkan audit keadilan untuk mendeteksi bias. Bereksperimenlah dengan platform pengujian yang ditingkatkan AI yang muncul untuk meningkatkan produktivitas dan cakupan.

16.10 KESIMPULAN

Pengujian aplikasi AI adalah disiplin ilmu yang beragam dan terus berkembang yang membutuhkan perpaduan antara prinsip pengujian perangkat lunak tradisional dan strategi khusus AI. Tidak seperti sistem konvensional, model AI digerakkan oleh data, probabilistik, dan seringkali nondeterministik, sehingga menimbulkan tantangan unik dalam hal keandalan, keadilan, ketahanan, dan keamanan. Bab ini menyajikan kerangka kerja komprehensif untuk menguji sistem AI, yang mencakup pengujian fungsional dan non-fungsional, verifikasi dan validasi model, penilaian bias dan keadilan, pengujian adversarial, dan validasi data. Bab ini juga menyoroti teknik-teknik baru seperti *pengujian mutasi* dan *pengujian diferensial*, yang meningkatkan kualitas data pengujian dan mengungkap perilaku kasus ekstrem.

Munculnya *AI generatif* semakin mengubah lanskap pengujian. Alat yang didukung oleh model bahasa besar dapat mengotomatiskan pembuatan pengujian, mensimulasikan beragam perilaku pengguna, dan memantau sistem secara real-time. Kemampuan ini mempercepat alur kerja pengujian dan memperluas cakupan, tetapi juga memperkenalkan risiko baru – seperti keluaran yang dihalusinasi, perilaku nondeterministik, dan masalah etika – yang memerlukan pengawasan yang cermat. Untuk mengatasi kompleksitas ini, organisasi harus mengadopsi *strategi pengujian adaptif dan berlapis*. Ini termasuk pemantauan berkelanjutan, evaluasi dengan melibatkan manusia, pengujian stres berbasis skenario, dan audit etika. Tim QA harus dibekali dengan literasi AI, kolaborasi lintas fungsi harus didorong, dan perangkat pengujian

harus berevolusi untuk mencakup kemampuan yang ditingkatkan dengan AI. Pada akhirnya, pengujian aplikasi AI yang efektif bukan hanya tentang memverifikasi kebenaran – tetapi tentang memastikan bahwa sistem *dapat dipercaya, transparan, dan tangguh* di lingkungan dunia nyata. Dengan mengintegrasikan praktik pengujian yang ketat dengan inovasi yang muncul, organisasi dapat dengan percaya diri menerapkan teknologi AI yang memberikan nilai sambil menjunjung tinggi standar kualitas dan tanggung jawab tertinggi.

16.11 PERTANYAAN

1. Bagaimana pengujian aplikasi AI berbeda dari pengujian perangkat lunak tradisional? Sebutkan area fokus utama dan jelaskan signifikansi masing-masing.
2. Bandingkan dan kontraskan Pengujian Pakar Domain (DET) dan Pengujian Penerimaan Pengguna (UAT) dalam konteks pengujian pengguna akhir untuk sistem AI. Berikan contoh konkret untuk masing-masing.
3. Ringkaslah aspek pengujian non-fungsional untuk sistem AI, termasuk kinerja, beban, stres, dan keamanan. Sertakan metrik atau perhatian utama untuk setiap kategori.
4. Apa prioritas utama pengujian fungsional dalam sistem AI, dan bagaimana teknik interpretasi mendukung prioritas ini?
5. Definisikan dan bedakan antara Kesetaraan Demografis, Kesempatan yang Sama, Peluang yang Sama, dan Rasio Dampak yang Berbeda. Jelaskan skenario di mana masing-masing akan menjadi metrik yang lebih disukai.
6. Metrik kinerja mana yang secara langsung mengukur "berapa banyak inferensi per satuan waktu" yang dapat diberikan oleh layanan AI?
 - A. Latensi
 - B. Skalabilitas
 - C. Throughput
 - D. Ketahanan
7. Pasangan mana yang paling tepat menggambarkan pengujian stres untuk aplikasi AI?
 - A. Periksa latensi rata-rata dan kepuasan pengguna di bawah beban tipikal
 - B. Dorong melampaui kapasitas normal untuk menemukan titik kritis dan memverifikasi pemulihan/stabilitas
 - C. Validasi alur UI dengan pakar domain
 - D. Terapkan peran akses dan enkripsi saat data tidak aktif
8. Metode interpretasi mana yang memberikan penjelasan tingkat lokal dan instan dengan mendekati model dengan pengganti sederhana di dekat input?
 - A. PDP (Partial Dependence Plot)
 - B. ICE (Individual Conditional Expectation)
 - C. LIME (Local Interpretable Model-agnostic Explanations)
 - D. SHAP (SHapley Additive exPlanations)

KUNCI JAWABAN UNTUK PERTANYAAN BAB

BAB 1: JAWABAN

1. Proses kognitif manusia “merasakan → menyimpan → memproses → bertindak” berfungsi sebagai model dasar untuk sistem kecerdasan buatan (AI). Paradigma ini tercermin dalam arsitektur AI, di mana sensor (seperti kamera, mikrofon, dan log data) mengumpulkan input mentah, lapisan penyimpanan (termasuk RAM, penyimpanan sekunder, dan data lake) menyimpan sejumlah besar informasi, algoritma pembelajaran menganalisis pola, dan model terlatih mengambil tindakan melalui prediksi atau keputusan.

Saluran ini diimplementasikan secara komputasi oleh berbagai arsitektur jaringan saraf. Jaringan Saraf Dalam (DNN), termasuk feed-forward dan perceptron multilayer, mempelajari abstraksi hierarkis untuk proses persepsi-ke-keputusan. Jaringan Saraf Konvolusional (CNN) mengkhususkan diri dalam memproses data spasial seperti gambar dan video, unggul dalam fase “merasakan” dengan mengekstrak properti lokal dan menggabungkannya menjadi fitur global. Jaringan Neural Berulang (RNN), termasuk varian Long Short-Term Memory (LSTM) dan Gated Recurrent Unit (GRU), menambahkan kemampuan memori temporal, mendukung “penyimpanan dan pemrosesan dari waktu ke waktu” untuk data sekuensial seperti ucapan dan teks.

Big Data sangat penting untuk kinerja AI modern karena model berkapasitas tinggi membutuhkan data yang beragam, bervolume besar, dan tepat waktu untuk melakukan generalisasi secara efektif. Model cenderung mengalami overfitting ketika data langka, gagal menangkap variabilitas dunia nyata yang dengan mudah dinavigasi oleh manusia. Konvergensi infrastruktur penyimpanan dan komputasi yang terukur (memungkinkan kemampuan “penyimpanan/pemrosesan” yang efisien) dengan kumpulan data yang sangat besar (memfasilitasi kemampuan “merasakan/mempelajari” yang komprehensif) telah memungkinkan mesin untuk melengkapi dan, dalam beberapa kasus, melampaui kemampuan manusia di domain tertentu.

2. 5V Big Data mencakup Volume, Velocity, Variety, Veracity, dan Value. Volume mengacu pada ukuran data yang sangat besar, sering diukur dalam terabyte hingga petabyte dan seterusnya. Salah satu contohnya adalah repositori genomik yang menyimpan jutaan sekuens genetik. Kecepatan (Velocity) menunjukkan laju kedatangan data, seringkali dalam aliran waktu nyata, seperti yang dicontohkan oleh umpan sensor lalu lintas langsung untuk sistem transportasi cerdas. Variasi (Variety) menggambarkan heterogenitas jenis data, termasuk format terstruktur (data tabular), semi-terstruktur (log), dan tidak terstruktur (teks, audio, video). Contoh lintas sektor adalah integrasi Rekam Medis Elektronik (EHR) rumah sakit, catatan dokter, dan data pencitraan medis. Kebenaran (Veracity) membahas kualitas dan ketidakpastian data, termasuk masalah akurasi, data yang hilang, dan bias.

Hal ini sangat relevan dalam data media sosial yang digunakan untuk pengawasan risiko. Terakhir, Nilai mewakili wawasan dan utilitas yang diperoleh dari data, seperti potensi penghematan biaya perawatan kesehatan AS sebesar \$300 miliar melalui jalur perawatan yang dioptimalkan berdasarkan analisis data komprehensif.

3. Alur analisis Big Data terdiri dari beberapa fase penting, masing-masing dengan tantangan teknis dan praktik terbaiknya sendiri. Pada fase Akuisisi/Perekaman, kekhawatiran meliputi kesenjangan asal usul dan pemilihan sinyal yang tepat. Praktik terbaik melibatkan perekaman asal usul transformasi dan penentuan informasi (apa, bagaimana, satuan, dan waktu) sebelumnya. Fase Pembersihan, Ekstraksi, dan Anotasi menghadapi tantangan informasi yang tidak lengkap atau tidak konsisten. Praktik untuk mengatasi hal ini meliputi penentuan aturan imputasi, penerapan pemeriksaan batasan, pengembangan model kesalahan, dan pelaksanaan validasi skema.

Selama Agregasi, Integrasi, dan Representasi, keragaman skema dan semantik di antara sumber menimbulkan tantangan yang signifikan. Praktik terbaik meliputi penggunaan deskriptor semantik yang jelas, pemetaan ontologi, dan pengembangan skema kanonik. Fase Pemodelan dan Analisis terutama menyangkut masalah skala yang terkait dengan komputasi, memori, dan batasan waktu. Praktik untuk mengurangi hal ini termasuk optimasi algoritma, pengambilan sampel/pemartisian strategis, dan pemanfaatan komputasi terdistribusi untuk pemrosesan batch dan stream. Terakhir, fase Interpretasi berisiko salah membaca kebiasaan dan kehilangan konteks. Untuk mengatasi hal ini, praktik-praktiknya meliputi pembingkai keputusan, tinjauan ahli domain, teknik visualisasi yang jelas, dan pelaporan ketidakpastian yang komprehensif.

4. A. AI Percakapan
5. E. API Keputusan
6. Pilihan bahasa pemrograman untuk pekerjaan AI bergantung pada berbagai faktor, termasuk persyaratan spesifik proyek, kebutuhan kinerja, dan keahlian pengembang. Python telah muncul sebagai pilihan yang disukai untuk banyak aplikasi AI karena keterbacaannya, dukungan komunitas yang kaya, kemampuan pembuatan prototipe yang cepat, dan ekosistem pembelajaran mesin dan AI yang luas, yang mencakup pustaka seperti NumPy, Pandas, Scikit-learn, PyTorch, dan TensorFlow.

Namun, bahasa lain menawarkan keunggulan yang berbeda dalam skenario tertentu. Java menyediakan alat bantu yang kuat, integrasi perusahaan, dan back-end yang skalabel, sehingga cocok untuk platform terdistribusi, sistem multi-robot, dan lingkungan produksi skala besar. C++ ideal untuk komponen yang sensitif terhadap latensi, sistem/aplikasi tertanam, dan tugas inferensi atau pelatihan yang kritis terhadap kinerja. Julia menawarkan kecepatan yang hampir setara dengan C dengan nuansa dinamis, unggul dalam komputasi ilmiah berkinerja tinggi dengan sintaks yang lebih sederhana. MATLAB® tetap kuat dalam komputasi numerik dan visualisasi, terbukti bermanfaat untuk pemrosesan sinyal, pengembangan, dan pembuatan prototipe. R sangat cocok untuk alur kerja analitik, pemodelan eksploratif, statistik, dan visualisasi data. Prolog dan Lisp memfasilitasi pembuatan prototipe cepat dari

penalaran berbasis aturan, AI simbolik, dan sistem berbasis logika/pengetahuan. Haskell, dengan paradigma fungsionalnya, efektif untuk struktur data kompleks dan konsistensi, menawarkan abstraksi khusus tetapi kuat untuk tugas-tugas AI tertentu.

Meskipun Python sering menjadi pilihan utama, alternatif mungkin lebih disukai dalam keadaan tertentu. Julia mungkin dipilih karena kecepatannya yang mendekati native untuk kode numerik, Java untuk integrasi JVM yang erat, atau C++ untuk memenuhi persyaratan latensi yang ketat. Pada akhirnya, Python mencapai keseimbangan yang sangat baik antara dukungan ekosistem dan produktivitas pengembang untuk sebagian besar aplikasi AI.

7. a. Fungsi `concat_two(a, b)` dapat diimplementasikan sebagai berikut:

```
def concat_two(a: str, b: str) -> str:
    return f"{a} {b}"
w1 = input("Enter first word: ")
w2 = input("Enter second word: ")
print(concat_two(w1, w2))
(b) To print all prime numbers from 2 to 30 using a for
loop with an else clause, without external libraries, we
can use the following code:
for n in range(2, 31):
    for x in range(2, n):
        if n % x == 0:
            # found a factor -> not prime
            break
    else:
        # loop didn't break -> prime
        print(n)
```

BAB 2: JAWABAN

1. Dalam tumpukan Big Data/AI modern, Spark, Kafka, NoSQL, dan Hadoop terintegrasi untuk menangani penyerapan, penyimpanan, pemrosesan, dan analisis data secara efisien: Kafka berfungsi sebagai platform streaming data real-time, menyerap dan mendistribusikan aliran data berkelanjutan dari berbagai sumber (misalnya, sensor atau aplikasi). Hadoop (khususnya HDFS untuk penyimpanan dan MapReduce atau YARN untuk pemrosesan) menyediakan penyimpanan terdistribusi yang terukur dan pemrosesan batch untuk volume data yang besar. Spark dibangun di atas Hadoop dengan menawarkan pemrosesan cepat dalam memori untuk tugas-tugas seperti pembelajaran mesin atau analitik, seringkali membaca data dari penyimpanan Hadoop. Basis data NoSQL (misalnya, MongoDB atau Cassandra) menangani penyimpanan dan pengambilan data tidak terstruktur atau semi-terstruktur yang fleksibel dan tanpa skema, terintegrasi dengan Spark untuk kueri real-time atau dengan Kafka untuk persistensi data streaming. Bersama-sama, mereka membentuk sebuah alur kerja:

Kafka memasukkan data ke Hadoop untuk penyimpanan, Spark memprosesnya dengan cepat, dan NoSQL memungkinkan kueri yang gesit untuk aplikasi AI. Kapan memilih masing-masing:

- **Kafka:** Pilih untuk streaming dan pengiriman pesan data secara real-time, seperti menangani log peristiwa bervolume tinggi atau umpan data IoT, di mana latensi rendah sangat penting.
- **Hadoop:** Pilih untuk pemrosesan batch skala besar dan penyimpanan dataset besar yang hemat biaya, terutama saat menangani analisis data historis atau pekerjaan ETL (Ekstrak, Transformasi, Muat).
- **Spark:** Pilih untuk pemrosesan iteratif yang cepat, beban kerja pembelajaran mesin, atau analitik real-time pada big data, karena unggul dalam kecepatan dan mendukung berbagai sumber data.
- **NoSQL:** Pilih untuk aplikasi yang membutuhkan desain skema yang fleksibel, skalabilitas tinggi, dan penanganan data tidak terstruktur, seperti profil pengguna di media sosial atau rekomendasi real-time dalam sistem AI.

2. Berikut perbandingan kecerdasan manusia dan mesin di tiga aspek:

- a. **Masukan data:** Kecerdasan manusia bergantung pada masukan sensorik (misalnya, penglihatan, pendengaran, sentuhan) untuk mengumpulkan data dari lingkungan, memungkinkan pemahaman kontekstual dan kemampuan beradaptasi, tetapi dibatasi oleh kendala biologis seperti kelelahan atau kelebihan beban sensorik. Kecerdasan mesin, di sisi lain, menggunakan masukan digital seperti sensor, API, kamera, atau aliran data, memungkinkan pengumpulan data yang cepat dan tepat dari berbagai sumber, meskipun kurang memiliki konteks bawaan tanpa algoritma yang diprogram.
- b. **Penyimpanan/pengambilan:** Kecerdasan manusia menyimpan data di jaringan saraf otak, yang bersifat asosiatif dan intuitif tetapi dapat tidak dapat diandalkan (misalnya, kelupaan, bias, atau degradasi seiring waktu). Pengambilan seringkali kreatif tetapi lambat dan dipengaruhi oleh emosi. Kecerdasan mesin menggunakan penyimpanan terstruktur seperti basis data atau chip memori, memungkinkan pengambilan data yang cepat, akurat, dan terukur (misalnya, melalui kueri dalam hitungan detik), tetapi kaku dan bergantung pada kualitas data dan algoritma.
- c. **Kemampuan pemrosesan:** Kecerdasan manusia unggul dalam pemecahan masalah kreatif, penalaran emosional, dan kemampuan beradaptasi terhadap situasi baru, tetapi lebih lambat untuk tugas-tugas berulang dan terbatas oleh kapasitas kognitif. Kecerdasan mesin lebih unggul dalam kecepatan, akurasi, dan penanganan komputasi besar-besaran (misalnya, memproses petabyte data), tetapi kesulitan dengan pemikiran abstrak, akal sehat, dan pengambilan keputusan etis tanpa pemrograman eksplisit.

Secara keseluruhan, kecerdasan manusia lebih holistik dan adaptif, sedangkan kecerdasan mesin efisien dan terukur, sehingga saling melengkapi dalam kolaborasi AI-manusia.

3. Sistem AI mencakup berbagai fitur dan subbidang. Di bawah ini, saya akan menjelaskan secara singkat fitur-fitur utamanya:

- **Fitur utama sistem AI:**

- **Pembelajaran Mesin (ML):** Kemampuan untuk belajar dari data dan meningkatkan kinerja dari waktu ke waktu tanpa pemrograman eksplisit, memungkinkan prediksi dan pengenalan pola.
- **Pemrosesan Bahasa Alami (NLP):** Fitur yang memungkinkan AI untuk memahami, menafsirkan, dan menghasilkan bahasa manusia, memfasilitasi chatbot, penerjemahan, dan analisis sentimen.
- **Visi Komputer:** Kemampuan untuk memproses dan menganalisis data visual (misalnya, gambar atau video), memungkinkan tugas-tugas seperti deteksi objek, pengenalan wajah, dan mengemudi otonom.
- **Robotika:** Integrasi AI dengan perangkat fisik untuk interaksi dengan lingkungan, termasuk pergerakan, manipulasi, dan pengambilan keputusan dalam skenario dunia nyata.
- **Pengambilan Keputusan dan Perencanaan:** Fitur yang memungkinkan AI untuk mengevaluasi opsi, memprediksi hasil, dan membuat pilihan yang optimal, sering digunakan dalam AI game atau alokasi sumber daya.
- **AI yang Etis dan Dapat Dijelaskan:** Fitur-fitur baru yang berfokus pada memastikan keadilan, transparansi, dan akuntabilitas dalam keputusan AI untuk mengurangi bias dan membangun kepercayaan.

- **Subbidang AI:**

- **Pembelajaran Mesin:** Subbidang di mana algoritma belajar dari data; termasuk pembelajaran terawasi, tidak terawasi, dan penguatan untuk aplikasi seperti sistem rekomendasi.
- **Pembelajaran Mendalam:** Subbidang ML yang menggunakan jaringan saraf dengan banyak lapisan untuk menangani data kompleks, unggul dalam tugas-tugas seperti pengenalan gambar.
- **Pemrosesan Bahasa Alami (NLP):** Berfokus pada memungkinkan mesin untuk memahami dan menghasilkan bahasa manusia, dengan aplikasi dalam asisten virtual dan analitik teks.
- **Visi Komputer:** Berurusan dengan interpretasi informasi visual, digunakan dalam pencitraan medis dan sistem keamanan.
- **Sistem Pakar:** Program AI yang meniru keahlian manusia dalam domain tertentu (misalnya, diagnosis medis) menggunakan penalaran berbasis aturan.
- **Robotika:** Melibatkan AI untuk mengendalikan robot, menggabungkan persepsi, pengambilan keputusan, dan tindakan fisik.

- **Pembelajaran Penguatan (Reinforcement Learning):** Subbidang di mana AI belajar melalui uji coba dan kesalahan dengan berinteraksi dengan lingkungan, umumnya digunakan dalam game dan optimasi.
- **Komputasi Kognitif (Cognitive Computing):** Bertujuan untuk mensimulasikan proses berpikir manusia, termasuk penalaran dan pembelajaran, untuk interaksi AI yang lebih intuitif.

Fitur dan subbidang ini sering tumpang tindih, mendorong kemajuan di bidang-bidang seperti perawatan kesehatan, keuangan, dan sistem otonom.

4. **Untuk aliran data real-time, komponen mana yang biasanya berfungsi sebagai perantara pesan terdistribusi antara penyedia dan konsumen data?**
- a. Hadoop
 - b. Spark
 - c. Kafka
 - d. NoSQL

Jawaban: c. Kafka

5. **Cabang kecerdasan buatan mana yang dibedakan oleh penerapan jaringan saraf tiruan multi-lapisan?**
- a. NLP
 - b. Sistem Pakar
 - c. Pembelajaran Mendalam (Deep Learning)
 - d. Visi Komputer (Computer Vision)

Jawaban: c. Pembelajaran Mendalam (Deep Learning)

6. **Mengenai pemrosesan batch dalam sistem big data, pernyataan berikut yang manakah yang BENAR?**
- a. Sistem ini harus menghasilkan hasil dengan cepat dan dapat bekerja dengan data yang tak terbatas dan terus menerus datang.
 - b. Sistem ini menggunakan kumpulan data statis dan terbatas yang disimpan secara permanen; volume data yang sangat besar mengakibatkan waktu pemrosesan yang lebih lama.
 - c. Hanya penggabungan streaming waktu nyata dengan latensi di bawah satu detik yang didukung.
 - d. Sistem ini tidak mampu mengelola kumpulan data yang sangat besar.

Jawaban: B

BAB 3: JAWABAN

1. Pipeline data adalah serangkaian proses yang memasukkan, memproses, dan mengirimkan data dari sumber ke konsumen untuk analitik atau aplikasi. Lima komponen utamanya adalah:
 - **Pengumpulan:** Memperoleh data mentah dari sumber (contoh: klien seluler/web melalui HTTP dan perangkat IoT melalui MQTT).

- **Pengambilan:** Memindahkan data secara andal ke lapisan staging atau bus pesan (contoh: Apache Kafka, AWS Kinesis, dan Kafka Connect).
 - **Transformasi (ETL/ELT):** Membersihkan, menormalisasi, dan membentuk data ke dalam format yang dapat digunakan (contoh: Apache Spark, dbt, dan AWS Glue).
 - **Komputasi/ML:** Rekayasa fitur, pelatihan model, dan inferensi (contoh: Spark MLlib, PyTorch, dan scikit-learn).
 - **Presentasi/Penyajian:** Mengekspos data yang telah dikurasi dan output model kepada pengguna dan sistem (contoh: alat BI seperti Tableau/Power BI, API REST/GRPC, dan gudang data).
2. **Arsitektur Lambda mengatur pemrosesan data menjadi tiga lapisan:**
- **Lapisan Batch:** Melakukan pemrosesan periodik berskala besar untuk menghasilkan tampilan batch yang akurat dan komprehensif.
 - **Lapisan Penyajian:** Mengindeks dan menyajikan hasil tampilan batch untuk mendukung kueri ad-hoc dan pembacaan latensi rendah.
 - **Lapisan Kecepatan (real-time):** Menghitung pembaruan inkremental latensi rendah untuk data terbaru yang belum ditangkap oleh pemrosesan batch.
Keuntungan:
(1) Menggabungkan akurasi batch throughput tinggi dengan pembaruan latensi rendah; (2) Toleransi kesalahan dan dapat diskalakan secara horizontal. Kerugian:
(1) Membutuhkan pemeliharaan dua jalur kode dan logika terpisah (batch dan kecepatan), meningkatkan kompleksitas dan risiko inkonsistensi; (2) Biaya operasional untuk pengujian, penerapan, dan debugging di seluruh pipeline paralel.
3. **Perbandingan arsitektur:** Twitter, Pinterest, dan Airbnb (fitur khas)
- **Twitter:** Secara historis menggunakan tumpukan bergaya Lambda skala besar (Hadoop/Scalding untuk batch, sistem streaming seperti Heron/Kafka untuk real-time). Seiring waktu, Twitter telah memigrasikan sebagian tumpukannya ke arah pub/sub dan analitik kolumnar (misalnya, Pub/Sub/BigQuery/Bigtable dalam migrasi cloud) untuk mendukung analitik latensi rendah dan deduplikasi dalam skala besar. Fitur khas: skala ekstrem dan persyaratan pemrosesan aliran latensi rendah.
 - **Pinterest:** Menggunakan penyerapan berbasis Kafka dengan sanitasi dan audit (Singer → Kafka → Merced → S3), menekankan kebersihan data dan pipeline peristiwa yang andal. Fitur khas: lapisan sanitasi/audit data yang ketat sebelum penyimpanan.
 - **Airbnb:** Mengandalkan Kafka dan penyerapan batch (Sqoop/Hive), Spark untuk transformasi, Airflow untuk orkestrasi, Presto/Trino untuk SQL ad-hoc, dan alat BI untuk presentasi. Fitur khas: Lapisan yang terpisah dengan jelas untuk analitik ad-hoc (Presto) dan produk data produksi, ditambah orkestrasi yang kuat melalui Airflow.

4. Satu manfaat dan satu risiko yang dimitigasi dari pemisahan logika bisnis dari model ML memastikan batasan yang jelas antara aturan aplikasi deterministik dan output model probabilistik. Manfaat: Memungkinkan pembaruan dan pengembalian model yang lebih aman dan cepat tanpa mengubah alur kerja bisnis inti, meningkatkan pemeliharaan dan ketangkasan penerapan. Risiko yang dimitigasi: Mengurangi kemungkinan ketidakstabilan atau pergeseran model menyebar langsung ke aturan bisnis penting atau perilaku yang dihadapi pengguna, menurunkan risiko operasional dan kepatuhan.
5. B. Penyajian
6. B. Presto → SQL ad-hoc melalui Hive/S3

Tugas pemrograman Tugas 1 – fungsi clean(text)

```
import re
def clean(text):
    """
    Lowercase, remove URLs and @handles, remove
    non-alphanumeric characters (keep spaces),
    collapse multiple spaces, and trim.
    """
    text = text.lower()
    # remove URLs
    text = re.sub(r'https?://\S+|www\.\S+', '', text)
    # remove @handles
    text = re.sub(r'@\w+', '', text)
    # keep only alphanumeric and spaces
    text = re.sub(r'[^a-z0-9\s]', ' ', text)
    # collapse whitespace
    text = re.sub(r'\s+', ' ', text).strip()
    return text
# Quick check
print(clean("OMG @user wow!! Check http://x.y  ")) # ->
"omg
wow check"
```

Tugas 2 – membersihkan daftar kamus, memberi tag emosi, dan menghitung jumlah cetakan

```
from collections import Counter
EMO_KEYWORDS = {
    "surprise": {"wow", "omg", "shocked", "unbelievable"},
    "joy": {"happy", "joy", "love", "great", "delighted",
    "blessed"},
    "fear": {"afraid", "scared", "terrified", "anxious",
```

```
"worried"},
    "sadness": {"sad", "down", "tears", "unhappy",
"depressed"},
}
def tag_emotion(clean_text):
    tokens = set(clean_text.split())
    for emotion, keywords in EMO_KEYWORDS.items():
        if tokens & keywords:
            return emotion
    return "unknown"
def process_texts(items):
    """
    items: list of dicts with keys 'id' and 'text'
    Returns emotion counts and list of tuples (id, cleaned_
text, tag)
    """
    counts = Counter()
    results = []
    for item in items:
        tid = item.get("id")
        txt = item.get("text", "")
        cleaned = clean(txt)
        tag = tag_emotion(cleaned)
        counts[tag] += 1
        results.append((tid, cleaned, tag))
    return counts, results
# Example usage
tweets = [
    {"id": "1", "text": "OMG I'm shocked!!"},
    {"id": "2", "text": "Feeling happy and blessed today"},
    {"id": "3", "text": "Kinda worried about the news"},
]
counts, processed = process_texts(tweets)
print("Emotion counts:", counts)
for row in processed:
```

BAB 4: JAWABAN

1. Pemrosesan batch, pemrosesan stream, dan pemrosesan hibrida mewakili paradigma yang berbeda dalam sistem Big Data, masing-masing sesuai untuk kasus penggunaan dan persyaratan yang berbeda. Pemrosesan batch melibatkan pemrosesan volume data yang besar dalam pekerjaan diskrit, biasanya dengan latensi tinggi tetapi throughput tinggi. Pendekatan ini sangat cocok untuk aplikasi seperti laporan penjualan harian atau pembaruan gudang data berkala. Sebaliknya, pemrosesan

stream menangani data secara real-time saat data tersebut tiba, menawarkan latensi rendah tetapi berpotensi throughput lebih rendah. Pendekatan ini unggul dalam skenario yang membutuhkan tindakan segera, seperti deteksi penipuan dalam transaksi keuangan atau sistem rekomendasi real-time. Pemrosesan hibrida menggabungkan elemen keduanya, memungkinkan pemrosesan data batch historis dan stream real-time. Pendekatan ini sangat berharga dalam aplikasi seperti pemeliharaan prediktif, di mana tren jangka panjang dari pemrosesan batch digabungkan dengan data sensor real-time untuk prediksi terkini.

2. Ekosistem Hadoop terdiri dari beberapa komponen kunci, termasuk HDFS (Hadoop Distributed File System), YARN (Yet Another Resource Negotiator), dan MapReduce. HDFS menyediakan lapisan penyimpanan terdistribusi, mempartisi kumpulan data besar di beberapa node untuk akses paralel. YARN berfungsi sebagai lapisan manajemen sumber daya, mengatur sumber daya komputasi di seluruh kluster. MapReduce, paradigma pemrosesan, memungkinkan komputasi terdistribusi melalui fase map dan reduce. Meskipun Hadoop menawarkan keunggulan seperti skalabilitas, toleransi kesalahan, dan kemampuan untuk memproses sejumlah besar data tidak terstruktur, ia juga menghadapi keterbatasan. Ini termasuk latensi tinggi untuk kueri interaktif, kompleksitas dalam model pemrograman, dan inefisiensi untuk algoritma iteratif. Dalam konteks pemrosesan Big Data modern, pendekatan berorientasi batch Hadoop mungkin kurang cocok untuk analitik real-time atau pemrosesan stream dibandingkan dengan kerangka kerja yang lebih baru seperti Apache Spark atau Flink.
3. Pembersihan data adalah proses mengidentifikasi dan memperbaiki kesalahan, inkonsistensi, dan ketidakakuratan dalam kumpulan data untuk meningkatkan kualitas data. Anomali data dapat diklasifikasikan menjadi tiga kategori utama: anomali sintaksis, semantik, dan cakupan. Anomali sintaksis mengacu pada kesalahan dalam format atau struktur data, seperti format tanggal yang tidak valid atau tipe data yang salah. Misalnya, tanggal yang dimasukkan sebagai "31/02/2023" secara sintaksis salah karena Februari tidak pernah memiliki 31 hari. Anomali semantik melibatkan pelanggaran aturan atau batasan khusus domain, seperti nilai usia negatif atau pembacaan suhu di luar rentang yang mungkin untuk lokasi tertentu. Anomali cakupan berkaitan dengan ketidaklengkapan atau redundansi dalam dataset, termasuk nilai yang hilang, catatan duplikat, atau kategori yang kurang terwakili. Contohnya adalah basis data pelanggan dengan kode pos yang hilang untuk beberapa entri atau duplikasi catatan pelanggan dengan ejaan nama yang sedikit berbeda.
4. **b.**Pemanfaatan RDD untuk komputasi dalam memori.
5. **C.**Penerapan aturan seperti penggabungan, pengurutan, pengkodean, dan penggabungan data.
6. Skrip Python untuk mengidentifikasi dan mengganti nilai yang hilang dalam dataset menggunakan imputasi rata-rata untuk kolom numerik dan imputasi modus untuk kolom kategorikal.

```
import pandas as pd
import numpy as np
def clean_missing_values(df):
    # Identify numerical and categorical columns
    numerical_columns = df.select_dtypes(include=[np.number]).columns
    categorical_columns = df.select_dtypes(exclude=[np.number]).columns
    # Mean imputation for numerical columns
    for col in numerical_columns:
        df[col].fillna(df[col].mean(), inplace=True)
    # Mode imputation for categorical columns
    for col in categorical_columns:
        df[col].fillna(df[col].mode()[0], inplace=True)
    return df
# Example usage
data = pd.read_csv('your_dataset.csv')
cleaned_data = clean_missing_values(data)
print(cleaned_data.isnull().sum()) # Check for any
remaining missing values
```

7. Skrip Python yang mengimplementasikan pipeline ETL yang diminta.

```
import pandas as pd
import numpy as np
def extract_data(file1, file2):
    """Extract data from two CSV files"""
    df1 = pd.read_csv(file1)
    df2 = pd.read_csv(file2)
    return df1, df2
def transform_data(df1, df2):
    """Transform data to ensure schema uniformity"""
    # Rename columns to ensure consistency
    df1.columns = [col.lower().replace(' ', '_') for col
in df1.columns]
    df2.columns = [col.lower().replace(' ', '_') for col
in df2.columns]
    # Ensure date columns are in the same format
    for df in [df1, df2]:
        date_columns = df.select_
dtypes(include=['datetime64']).columns
        for col in date_columns:
            df[col] = pd.to_datetime(df[col]).
dt.strftime('%Y-%m-%d')
    # Merge dataframes
```

```
        merged_df = pd.concat([df1, df2], axis=0,
ignore_index=True)
    return merged_df
def cleanse_data(df):
    """Cleanse data by removing duplicates and resolving inconsistencies"""
    # Remove duplicate rows
    df.drop_duplicates(inplace=True)
    # Handle missing values
    numerical_columns = df.select_dtypes(include=[np.number]).columns
    categorical_columns = df.select_dtypes(exclude=[np.number]).columns
    for col in numerical_columns:
        df[col].fillna(df[col].mean(), inplace=True)
    for col in categorical_columns:
        df[col].fillna(df[col].mode()[0], inplace=True)
    # Resolve inconsistencies (example: standardize text case in string columns)
    for col in categorical_columns:
        if df[col].dtype == 'object':
            df[col] = df[col].str.lower()
    return df
def load_data(df):
    """Load the cleansed data into a consolidated DataFrame"""
    # In a real-world scenario, this might involve writing to a database
    # For this example, we'll just return the final DataFrame
    return df
def etl_pipeline(file1, file2):
    """Main ETL pipeline function"""
    # Extract
    df1, df2 = extract_data(file1, file2)
    # Transform
    merged_df = transform_data(df1, df2)
    # Cleanse
    cleansed_df = cleanse_data(merged_df)
    # Load
    final_df = load_data(cleansed_df)
    return final_df
# Example usage
file1 = 'data1.csv'
```

```
file2 = 'data2.csv'  
result = etl_pipeline(file1, file2)  
print(result.head())  
print(result.info())
```

BAB 5: JAWABAN

1. Pipeline otomatis untuk pembersihan data sangat penting karena secara signifikan meningkatkan reproduksibilitas, skalabilitas, dan efisiensi. Dengan memanfaatkan fungsi modular dan dapat digunakan kembali, ilmuwan data dapat menerapkan standar pembersihan yang konsisten di berbagai dataset, mengurangi redundansi kode, dan memfasilitasi penggunaan kembali kode yang lebih mudah dalam proyek yang lebih besar atau kompleks.
2. Fungsi seperti `getFirstLook`, `getMissing`, `getDisports`, dan `getOutliers` sangat penting untuk menilai dan meningkatkan kualitas data:
 - **getFirstLook:** Memberikan gambaran umum tingkat tinggi tentang struktur dataset, termasuk head, tipe data, dan dimensi.
 - **getMissing:** Mengkuantifikasi nilai yang hilang berdasarkan baris dan kolom.
 - **getDisports:** Meringkas statistik distribusi, seperti mean, skewness, dan kurtosis.
 - **getOutliers:** Mengidentifikasi anomali berdasarkan ambang batas Interquartile Range (IQR). Secara kolektif, fungsi-fungsi ini berfungsi sebagai blok bangunan dasar dari pipeline data yang kuat.
3. Memeriksa kualitas data secara keseluruhan biasanya melibatkan langkah-langkah kunci berikut:
 - **checkCats:** Memverifikasi bahwa variabel kategorikal hanya berisi kategori yang diharapkan.
 - **checkOutliers:** Mengidentifikasi nilai numerik yang berada di luar rentang yang valid atau diharapkan.
 - **runChecks:** Mengotomatiskan logika validasi di seluruh bidang kategorikal, numerik, dan pengidentifikasi.

Contoh: Dalam dataset NLS, pemeriksaan ini mengungkapkan bahwa variabel status perkawinan melebihi ambang batas maksimum untuk nilai yang hilang (26%), dan variabel "tingkat pendidikan tertinggi yang diselesaikan" berisi nilai yang tidak diharapkan.
4. b) Outlier diganti dengan nilai NaN
5. b) `make_pipeline`
6. **Latihan Pemrograman – Solusi Dasar (fungsi Python):**

```
import pandas as pd  
import numpy as np  
def get_missings(df, by_row_perc=False):  
    """  
    Counts missing values by columns and rows.  
    """
```

```
return df.isnull().sum(), \
        df.isnull().sum(axis=1).value_
counts(normalize=by_row_perc).sort_index()
# Test the function
df = pd.DataFrame({
    'A': [1, np.nan, 3],
    'B': [4, 5, np.nan]
})
print(get_missings(df))
```

7. Latihan Pemrograman – Solusi Tingkat Lanjut (Konstruksi pipeline dengan Scikit-learn):

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_validate,
KFold
from sklearn.base import BaseEstimator, TransformerMixin
import numpy as np
import pandas as pd
class OutlierTrans(BaseEstimator, TransformerMixin):
    def __init__(self, threshold=1.5):
        self.threshold = threshold
    def fit(self, X, y=None):
        return self
    def transform(self, X, y=None):
        X_new = X.copy()
        # Iterate over columns to handle outliers
        for col in X_new.columns:
            q1, q3 = X_new[col].quantile([0.25, 0.75])
            iqr = self.threshold * (q3 - q1)
            lower, upper = q1 - iqr, q3 + iqr
            # Replace outliers with NaN
            mask = (X_new[col] < lower) | (X_new[col]
> upper)
            X_new.loc[mask, col] = np.nan
        return X_new
# Create dummy data
df = pd.DataFrame({
    'x1':,
    'x2': [5, np.nan, 7, 8, 6],
    'y':
```

```
})  
X, y = df[['x1', 'x2']], df['y']  
# Define the pipeline  
pipeline = make_pipeline(  
    StandardScaler(),  
    OutlierTrans(),  
    SimpleImputer(strategy='mean'),  
    LinearRegression()  
)  
# Evaluate using Cross-Validation  
kf = KFold(n_splits=2, shuffle=True, random_state=0)  
# Changed splits to 2 for this small dataset  
scores = cross_validate(pipeline, X, y, cv=kf,  
    scoring=['r2', 'neg_mean_absolute_error'])  
print("Cross-validation scores:", scores)
```

BAB 6: JAWABAN

1. Kecerdasan Kawan (Swarm Intelligence/SI) mengacu pada perilaku kolektif dan adaptif yang muncul dari sejumlah besar agen yang relatif sederhana yang berinteraksi secara lokal sesuai dengan aturan sederhana, tanpa kendali terpusat. Sistem SI memanfaatkan pengambilan keputusan yang terdesentralisasi, persepsi dan interaksi lokal, serta umpan balik berulang untuk menghasilkan hasil global yang koheren; contoh biologis kanonik meliputi koloni semut, kawanan burung, dan masyarakat lebah madu. Prinsip-prinsip dasar yang mengatur SI adalah pembagian kerja, stigmergi, dan pengaturan diri. Pembagian kerja terwujud sebagai diferensiasi peran di antara agen untuk meningkatkan efisiensi keseluruhan (misalnya, pembagian tugas di antara lebah pekerja dan pencari makan). Stigmergi menunjukkan koordinasi tidak langsung melalui modifikasi lingkungan, seperti jejak feromon yang diletakkan oleh semut yang membiarkan pergerakan agen selanjutnya dan memperkuat jalur yang menjanjikan. Pengaturan diri menggambarkan bagaimana interaksi lokal yang berulang dan lingkaran umpan balik menghasilkan struktur atau fungsionalitas tingkat sistem yang muncul tanpa orkestrasi dari atas ke bawah.
2. Sistem SI menghadirkan keuntungan yang signifikan serta tantangan penting. Di antara kekuatan utamanya adalah kekokohan dan skalabilitas: karena kontrol didistribusikan, sistem mentolerir kegagalan agen individu dan dapat diskalakan ke populasi besar dengan perubahan sederhana pada aturan lokal; selain itu, metode SI menunjukkan fleksibilitas, memungkinkan adaptasi terhadap contoh masalah yang beragam dan dinamis tanpa perancangan ulang yang ekstensif. Sebaliknya, pendekatan SI menimbulkan tantangan dalam desain, analisis, dan penerapan: menentukan aturan lokal yang secara andal menghasilkan perilaku yang diinginkan seringkali tidak mudah dan kurang memiliki jaminan analitis umum; algoritma SI rentan terhadap konvergensi prematur atau stagnasi di wilayah suboptimal dari ruang pencarian; biaya komputasi

dan sensitivitas parameter dapat tinggi untuk masalah yang kompleks; dan reproduksibilitas dan interpretasi solusi yang muncul mungkin terbatas, sehingga mempersulit validasi dalam konteks yang kritis terhadap keselamatan.

3. Stigmergi adalah mekanisme di mana agen berkoordinasi secara tidak langsung melalui modifikasi lingkungan mereka yang persisten; jejak lingkungan ini berfungsi baik sebagai sinyal jangka pendek maupun sebagai memori jangka panjang yang memandu tindakan kolektif. Contoh prototipe adalah peletakan jejak oleh semut: semut pencari makanan secara individu menyimpan feromon saat kembali dari sumber makanan, dan semut lain lebih memilih mengikuti konsentrasi feromon yang lebih kuat, memperkuat rute yang efektif dan memungkinkan koloni untuk berkumpul di jalur pencarian makanan yang efisien. Stigmergi memfasilitasi munculnya kecerdasan kolektif dengan mengubah pengamatan lokal menjadi isyarat stigmergi bersama yang memengaruhi perilaku agen selanjutnya, sehingga memperkuat penemuan yang bermanfaat dan memungkinkan pemecahan masalah terdistribusi tanpa pesan antar agen yang eksplisit.
4. d) Pengawasan Terpusat.
5. a) Parameter Kognitif dan Sosial.
6. Untuk memodifikasi skrip berbasis PSO untuk melaporkan kinerja klasifikasi per iterasi, masukkan langkah evaluasi dalam loop PSO yang mendekode posisi partikel saat ini menjadi subset fitur atau nilai hyperparameter, menyesuaikan *RandomForestClassifier* pada data pelatihan (atau mengevaluasi pada set validasi), menghitung skor akurasi, dan mencetak atau mencatat skor tersebut pada setiap iterasi. Pastikan pemisahan yang tepat antara data pelatihan dan validasi dan pertimbangkan untuk merata-ratakan lipatan atau seed untuk mengurangi varians dalam metrik yang dilaporkan.
7. Untuk melakukan pemilihan fitur dengan PSO dan membandingkan kinerja pengklasifikasi sebelum dan sesudah pemilihan, lanjutkan sebagai berikut: pilih dataset dan praproses (tangani nilai yang hilang, encode kategorikal, skalakan fitur sesuai kebutuhan); implementasikan PSO biner atau berkode riil di mana setiap partikel mengkodekan subset fitur kandidat (untuk pengkodean biner) atau bobot kepentingan yang diberi ambang batas; Tetapkan tujuan yang mengevaluasi fitur terpilih dengan melatih pengklasifikasi (misalnya, *RandomForest*) dan mengembalikan metrik kinerja validasi (akurasi, akurasi seimbang, atau skor validasi silang), mungkin ditambah dengan penalti sparsitas; jalankan PSO untuk mengidentifikasi himpunan fitur yang ringkas; akhirnya, latih dan evaluasi pengklasifikasi menggunakan himpunan fitur lengkap dan subset yang dipilih PSO di bawah kondisi evaluasi yang identik dan laporkan hasil perbandingan, menafsirkan perubahan dalam akurasi, kompleksitas model, dan ketahanan. Pendekatan ini lebih disukai daripada pencarian menyeluruh karena ruang pencarian subset fitur tumbuh secara kombinatorial dan PSO menyediakan pencarian stokastik yang efisien dan terarah yang menyeimbangkan eksplorasi dan eksploitasi sambil mengakomodasi evaluasi kinerja yang bising.

BAB 7: JAWABAN

1. Algoritma Genetika (GA) dan Pemrograman Genetika (GP) keduanya merupakan metode pencarian evolusioner berbasis populasi yang menggunakan seleksi, variasi, dan penggantian berdasarkan kebugaran, tetapi keduanya berbeda terutama dalam representasi dan jenis solusi yang mereka kembangkan: GA biasanya mengembangkan solusi kandidat terkode dengan panjang tetap (kromosom, seringkali berupa string bit atau vektor riil) yang cocok untuk optimasi parameter, sedangkan GP mengembangkan program yang dapat dieksekusi dengan struktur variabel (biasanya pohon sintaksis) yang merepresentasikan fungsi atau prosedur. Representasi/pengkodean membentuk dinamika pencarian karena kromosom memungkinkan crossover dan mutasi langsung pada posisi tetap tetapi dapat kesulitan untuk mengkodekan program terstruktur secara kompak, sedangkan pohon sintaksis memungkinkan rekombinasi program yang bermakna (crossover subpohon) dan variasi struktural tetapi membutuhkan operator yang mempertahankan validitas sintaksis. Perbedaan genotipe-fenotipe itu penting: genotipe adalah individu yang dikodekan (bit, vektor, struktur pohon) sedangkan fenotipe adalah kandidat yang diekspresikan (parameter yang didekodekan, perilaku program, rute, dan pengklasifikasi). Pengkodean/dekode memengaruhi lokalitas dan netralitas pencarian – perubahan genotipe kecil dapat memiliki efek fenotipe kecil atau besar tergantung pada pengkodeannya. Seleksi, persilangan, mutasi, dan evaluasi kebugaran secara bersama-sama membentuk eksplorasi vs. eksploitasi: seleksi membiaskan reproduksi ke arah fenotipe dengan kebugaran lebih tinggi (eksploitasi), persilangan menggabungkan kembali materi genetik untuk mengeksplorasi kombinasi baru, mutasi memperkenalkan kebaruan acak untuk menghindari optimasi lokal, dan evaluasi kebugaran mendefinisikan lanskap objektif yang memandu pencarian. Alur kerja (poin-poin ringkas): inisialisasi populasi → evaluasi kebugaran setiap individu → pilih induk (seleksi) → buat keturunan melalui crossover dan mutasi (variasi) → evaluasi kebugaran keturunan → bentuk generasi berikutnya melalui strategi penggantian → periksa penghentian (generasi maksimum, ambang kebugaran, atau stagnasi) → keluarkan solusi terbaik.
2. Proses GA/GP ujung-ke-ujung dimulai dengan menghasilkan populasi awal yang beragam (genotipe acak atau individu yang disemai secara heuristik). Setiap generasi, individu dievaluasi menggunakan fungsi kebugaran; seleksi induk dilakukan berdasarkan kebugaran (turnamen, peringkat, dan roulette) untuk menghasilkan kumpulan perkawinan; crossover menggabungkan genotipe induk untuk membentuk keturunan; mutasi secara acak mengganggu keturunan; keturunan dievaluasi dan digabungkan dengan atau menggantikan anggota populasi saat ini sesuai dengan kebijakan penggantian (generasional, keadaan stabil, elitisme untuk mempertahankan yang terbaik), dan kriteria penghentian diperiksa (generasi maksimum, kebugaran yang cukup, atau kurangnya peningkatan). Keacakan sangat penting untuk menjelajahi ruang pencarian dan mencegah jebakan deterministik; hal ini memberikan variasi dan

memungkinkan pengambilan sampel dari berbagai wilayah, sehingga memungkinkan eksplorasi dan pendakian bukit stokastik.

- A. Konvergensi prematur terjadi ketika populasi kehilangan keragaman dan konvergen ke solusi suboptimal lebih awal; hal ini mengurangi kemampuan algoritma untuk menemukan wilayah yang lebih baik dan menyebabkan stagnasi. Dua strategi untuk mempertahankan keragaman: (1) menggunakan seleksi/penggantian yang mempertahankan keragaman seperti pembagian kebugaran, kepadatan, atau metode ceruk dan mempertahankan ambang batas arsip/elitisme yang mempertahankan beberapa solusi berbeda; (2) menyuntikkan variasi melalui tingkat mutasi yang lebih tinggi, operator mutasi adaptif, atau imigran acak sesekali dan menggunakan metode seleksi yang membatasi tekanan seleksi (misalnya, turnamen dengan k kecil, seleksi peringkat).
3. Penyetelan hiperparameter/seleksi fitur: tujuan – untuk menemukan pengaturan hiperparameter atau subset fitur yang memaksimalkan kinerja validasi; pengkodean – GA mengkodekan hiperparameter sebagai vektor panjang tetap (kontinu dan kategorikal yang dipetakan ke gen) atau masker biner untuk subset fitur; GA/GP lebih disukai karena ruang pencarian bersifat kombinatorial, multimodal, dan mahal untuk dievaluasi sehingga pencarian stokastik terpandu menemukan konfigurasi yang baik lebih cepat daripada pencarian menyeluruh dan dapat menangani parameter tipe campuran dan ketergantungan bersyarat. Pemrosesan gambar/sinyal (misalnya, desain filter atau konstruksi fitur): tujuan – untuk menemukan filter, transformasi, atau alur pemrosesan yang meningkatkan klasifikasi, pengurangan noise, atau kualitas fitur; pengkodean – GP mengkodekan alur pemrosesan atau ekspresi filter sebagai pohon sintaksis (operator dan terminal), GA dapat mengkodekan parameter filter sebagai vektor; GA/GP lebih disukai karena ruang struktur alur sangat besar dan tidak dapat didiferensiasi, GP dapat menemukan operator atau komposisi baru yang mungkin terlewatkan oleh desain manual, dan pencarian evolusioner mentolerir evaluasi kebugaran yang bising. Perutean logistik (misalnya, perutean Amazon): tujuan – untuk menghasilkan rute pengiriman yang mendekati optimal yang meminimalkan biaya/waktu di bawah kendala; pengkodean – GA mengkodekan rute sebagai permutasi (kromosom yang mewakili urutan kunjungan), GP dapat mengembangkan heuristik konstruksi rute; GA/GP mengalahkan brute-force karena penentuan rute adalah NP-hard dengan pertumbuhan solusi faktorial, sehingga pencarian menyeluruh tidak layak dilakukan pada skala realistis, sementara pencarian evolusioner secara efisien mengeksplorasi rute berkualitas tinggi dan dapat menggabungkan batasan serta istilah kebugaran khusus.
4. B) Pohon sintaksis yang simpul internalnya adalah operator dan daunnya adalah terminal
5. C) Mengukur kualitas solusi dan mendorong pemilihan induk
6. Latihan Pemrograman – Ekstensi TSP GA (pemilihan turnamen, pelacakan kebugaran terbaik, dan penghentian dini)

```
import numpy as np
import random
import math
random.seed(1)
np.random.seed(1)
#Create random cities
def create_cities(n, seed=1):
    rng = np.random.RandomState(seed)
    return rng.rand(n, 2)
def tour_length(tour, cities):
    tour_cities = cities[tour]
    # closed tour
    d = np.sqrt(((tour_cities - np.roll(tour_cities,
-1, axis=0))**2).sum(axis=1)).sum()
    return d
#Initialize population as random permutations
def init_population(pop_size, n_cities):
    return [np.random.permutation(n_cities) for _ in
range(pop_size)]
#Tournament selection (k=3)
def tournament_select(pop, fitnesses, k=3):
    idxs = np.random.choice(len(pop), k, replace=False)
    best = idxs[0]
    for i in idxs[1:]:
        if fitnesses[i] < fitnesses[best]: # fitness =
tour length (lower is better)
            best = i
    return pop[best].copy()
#Ordered crossover (OX)
def ordered_crossover(p1, p2):
    n = len(p1)
    a, b = sorted(random.sample(range(n), 2))
    child = *n
    child[a:b+1] = p1[a:b+1]
    fill_positions = [i for i in range(n) if child[i]
== -1]
    p2_iter = [city for city in p2 if city not in
child]
    for pos, city in zip(fill_positions, p2_iter):
        child[pos] = city
    return np.array(child, dtype=int)
#Swap mutation
def swap_mutation(tour, mutation_rate=0.02):
    t = tour.copy()
    if random.random() < mutation_rate:
        i, j = random.sample(range(len(t)), 2)
        t[i], t[j] = t[j], t[i]
    return t
def evolve(cities, pop_size=100, generations=1000, cx_
rate=0.9, mut_rate=0.02, early_stop_patience=20):
    n = len(cities)
    pop = init_population(pop_size, n)
```

```
    fitnesses = [tour_length(ind, cities) for ind in
pop]
    best_idx = int(np.argmin(fitnesses))
    best = pop[best_idx].copy()
    best_fitness = fitnesses[best_idx]
    history = [best_fitness]
    no_improve = 0
for gen in range(1, generations+1):
    new_pop = []
    while len(new_pop) < pop_size:
        parent1 = tournament_select(pop, fitnesses,
k=3)
        parent2 = tournament_select(pop, fitnesses,
k=3)
        if random.random() < cx_rate:
            child = ordered_crossover(parent1, parent2)
        else:
            child = parent1.copy()
            child = swap_mutation(child,
mutation_rate=mut_rate)
        new_pop.append(child)
    pop = new_pop
    fitnesses = [tour_length(ind, cities) for ind in
pop]
    cur_best_idx = int(np.argmin(fitnesses))
    cur_best = pop[cur_best_idx].copy()
    cur_best_fitness = fitnesses[cur_best_idx]
    history.append(cur_best_fitness)
    if cur_best_fitness < best_fitness - 1e-12:
        best_fitness = cur_best_fitness
        best = cur_best.copy()
        no_improve = 0
    else:
        no_improve += 1
        print(f"Gen {gen:4d} best_length = {cur_best_
fitness:.6f} overall_best = {best_fitness:.6f} no_
improve = {no_improve}")
        if no_improve >= early_stop_patience:
            print(f"Early stopping at generation {gen} (no
improvement for {early_stop_patience} generations).")
            break
return best, best_fitness, history
if name == "main":
    cities = create_cities(20, seed=42)
    best_tour, best_len, history = evolve(cities,
pop_size=200, generations=1000, cx_rate=0.9, mut_
rate=0.05, early_stop_patience=20)
    print("Best length:", best_len)
    print("Best tour:", best_tour)
```

7. Latihan Pemrograman – Pemrograman Genetika untuk Regresi Simbolik

```
import random
import math
import operator
import numpy as np
import matplotlib.pyplot as plt
from copy import deepcopy
random.seed(1)
np.random.seed(1)
#Define primitives
def protected_div(a, b):
    try:
        return a / b if abs(b) > 1e-6 else a
    except Exception:
        return 0.0
PRIMITIVES = {
    'add': (lambda a,b: a + b, 2),
    'sub': (lambda a,b: a - b, 2),
    'mul': (lambda a,b: a * b, 2),
    'pdiv': (protected_div, 2)
}
PRIM_NAMES = list(PRIMITIVES.keys())
#Node classes
class Node:
    def eval(self, x):
        raise NotImplementedError
    def deepcopy(self):
        return deepcopy(self)
class FuncNode(Node):
    def init(self, name, children):
        self.name = name
        self.func = PRIMITIVES[name][0]
        self.arity = PRIMITIVES[name][1]
        self.children = children
    def eval(self, x):
        vals = [c.eval(x) for c in self.children]
        return self.func(*vals)
    def str(self):
        return f"({self.name} {' '.join(str(c) for c in self.
children)})"
class ConstNode(Node):
    def init(self, value):
        self.value = float(value)
    def eval(self, x):
        return self.value
    def str(self):
        return f"{self.value:.3f}"
class VarNode(Node):
    def init(self, name='x'):
        self.name = name
    def eval(self, x):
        return x
    def str(self):
        return self.name
```

```
#Random tree generation (ramped half-and-half style)
def random_tree(max_depth, grow=True):
    if max_depth == 0 or (grow and random.random() < 0.3):
        # terminal: var or ephemeral constant
        if random.random() < 0.5:
            return VarNode()
        else:
            return ConstNode(random.uniform(-5, 5))
    else:
        name = random.choice(PRIM_NAMES)
        arity = PRIMITIVES[name][1]
        children = [random_tree(max_depth-1, grow) for
_in range(arity)]
        return FuncNode(name, children)

#Tree utilities
def all_nodes(root):
    nodes = []
    def recurse(node, parent, idx):
        nodes.append((node, parent, idx))
        if isinstance(node, FuncNode):
            for i, c in enumerate(node.children):
                recurse(c, node, i)
    recurse(root, None, None)
    return nodes

def subtree_crossover(a, b):
    a_copy = a.deepcopy()
    b_copy = b.deepcopy()
    a_nodes = all_nodes(a_copy)
    b_nodes = all_nodes(b_copy)
    # choose random non-root node in each
    node_a, parent_a, idx_a = random.choice(a_nodes)
    node_b, parent_b, idx_b = random.choice(b_nodes)
    # swap subtrees
    if parent_a is None:
        return node_b.deepcopy()
    else:
        parent_a.children[idx_a] = node_b.deepcopy()
        return a_copy

def point_mutation(tree, max_depth=4):
    tree_copy = tree.deepcopy()
    nodes = all_nodes(tree_copy)
    node, parent, idx = random.choice(nodes)
    new_sub = random_tree(max_depth=max_depth,
grow=True)
    if parent is None:
        return new_sub
    else:
        parent.children[idx] = new_sub
        return tree_copy

def tree_to_callable(tree):
    def f(x):
        return tree.eval(x)
    return f
```

```
#Fitness: MSE on sample points
def fitness(tree, xs, ys):
    f = tree_to_callable(tree)
    preds = np.array([f(x) for x in xs], dtype=float)
    return np.mean((preds - ys)**2)

#Population init
def init_population(pop_size, max_depth):
    pop = []
    for i in range(pop_size):
        d = random.randint(1, max_depth)
        grow = random.choice([True, False])
        pop.append(random_tree(d, grow=grow))
    return pop

#GP loop
def gp_run(pop_size=500, generations=50, max_depth=5,
cx_rate=0.9, mut_rate=0.1, tournament_k=3):
    # sample points
    xs = np.linspace(-3, 3, 50)
    ys = xs**2 + xs + 1
    pop = init_population(pop_size, max_depth)
    best = None
    best_score = float('inf')
    for gen in range(1, generations+1):
        # evaluation
        scores = [fitness(ind, xs, ys) for ind in pop]
        # track best
        cur_best_idx = int(np.argmin(scores))
        cur_best_score = scores[cur_best_idx]
        if cur_best_score < best_score:
            best_score = cur_best_score
            best = pop[cur_best_idx].deepcopy()
        print(f"Gen {gen:3d} best MSE {cur_best_score:.6f}")
    overall_best {best_score:.6f}")
    # selection & variation
    new_pop = []
    while len(new_pop) < pop_size:
        # tournament selection
        def tournament():
            idxs = random.sample(range(pop_size),
tournament_k)
            best_i = min(idxs, key=lambda i: scores[i])
            return pop[best_i].deepcopy()
        parent1 = tournament()
        parent2 = tournament()
        if random.random() < cx_rate:
            child = subtree_crossover(parent1, parent2)
        else:
            child = parent1
        if random.random() < mut_rate:
            child = point_mutation(child, max_depth)
        new_pop.append(child)
    pop = new_pop
# final evaluation and output
```

```
xs_test = np.linspace(-3, 3, 200)
ys_test = xs_test**2 + xs_test + 1
f_best = tree_to_callable(best)
preds=np.array([f_best(x) for x in xs_test], dtype=float)
mse_final = np.mean((preds - ys_test)**2)
print("\nBest expression:", str(best))
print("Final MSE on dense test set:", mse_final)
# plot
plt.figure(figsize=(6,4))
plt.plot(xs_test, ys_test, label='true', color='black')
plt.plot(xs_test, preds, label='gp_pred', color='red',
linestyle='--')
plt.scatter(xs, np.array([f_best(x) for x in xs]), s=10,
color='blue', label='pred_samples')
plt.legend()
plt.title(f"GP symbolic regression - MSE {mse_final:.6f}")
plt.xlabel('x')
plt.ylabel('y')
plt.tight_layout()
plt.show()
return best, mse_final
if name == "main":
    best_tree, mse = gp_run(pop_size=300, genera-
tions=40, max_depth=5, cx_rate=0.9, mut_rate=0.2,
tournament_k=5)
```

BAB 8: JAWABAN

1. Perceptron adalah neuron komputasi dasar yang menghitung jumlah input berbobot dan meneruskannya melalui aktivasi: $y = f(w \cdot x + b)$. Bobot menskalakan dan menandai kontribusi setiap input dan dengan demikian menetapkan orientasi batas keputusan; bias adalah offset aditif yang menggeser batas keputusan menjauh dari titik asal (ia menetapkan ambang aktivasi); fungsi aktivasi memperkenalkan nonlinieritas, memetakan kombinasi linier ke rentang output yang dibutuhkan untuk tugas dan memengaruhi aliran gradien selama pelatihan.
2. Sigmoid, tanh, ReLU, dan softmax berbeda dalam rentang dan penggunaan tipikal. Sigmoid(z) = $\frac{1}{1+e^{-z}}$ menghasilkan output (0,1), secara historis digunakan untuk probabilitas biner dan unit output tetapi tidak ideal di lapisan tersembunyi karena output yang tidak berpusat pada nol dan gradien yang menghilang. tanh(z) = $\tanh(z)$ menghasilkan output (-1,1), berpusat pada nol dan sering kali lebih disukai daripada sigmoid di lapisan tersembunyi karena output negatif dan positifnya membantu optimasi. ReLU(z) = $\max(0, z)$ menghasilkan output $[0, \infty)$, secara komputasi sederhana, mendorong aktivasi sparse dan menghindari beberapa masalah vanishing-gradient, menjadikannya aktivasi tersembunyi default untuk banyak jaringan saraf dalam; ia dapat mengalami "dying ReLU" ketika unit terjebak pada nol. Softmax yang diterapkan pada vektor z memberikan komponen $\frac{\exp(z_i)}{\sum_j \exp(z_j)}$, menghasilkan distribusi

probabilitas di seluruh kelas (masing-masing dalam (0,1) dan jumlahnya 1) dan digunakan dalam lapisan output klasifikasi multikelas.

3. Propagasi maju dalam perceptron multilayer berarti meneruskan input melalui setiap lapisan satu demi satu. Pada setiap lapisan, sinyal yang masuk digabungkan secara linier menggunakan bobot dan bias lapisan tersebut, kemudian ditransformasikan oleh fungsi aktivasi lapisan untuk menghasilkan output lapisan, yang menjadi input lapisan berikutnya. Proses ini berlanjut hingga jaringan menghasilkan prediksi akhirnya, dan kerugian (kesalahan) relatif terhadap target sebenarnya dihitung.

Backpropagation adalah proses menentukan bagaimana mengubah parameter jaringan untuk mengurangi kerugian. Proses ini dimulai dengan mengukur kesalahan pada lapisan output dan kemudian bekerja mundur melalui jaringan, lapisan demi lapisan. Pada setiap langkah, proses ini menggunakan sensitivitas lokal dari fungsi aktivasi bersama dengan kesalahan yang dipropagasi dari lapisan berikutnya untuk menghitung seberapa besar kontribusi setiap bobot dan bias terhadap kerugian keseluruhan. Kontribusi yang dihitung tersebut membentuk gradien untuk setiap parameter. Akhirnya, algoritma optimasi (misalnya, stochastic gradient descent atau Adam) menggunakan gradien tersebut untuk memperbarui bobot dan bias guna mengurangi kerugian pada proses selanjutnya.

4. Hyperparameter kunci meliputi pilihan arsitektur (jumlah lapisan, unit per lapisan, ukuran konvolusi/filter, dan attention head), pengaturan optimasi (tingkat pembelajaran, optimizer, ukuran batch, momentum, dan weight decay), regularisasi (tingkat dropout, penalti L_2 , dan augmentasi data), dan jadwal pelatihan (epoch, jadwal tingkat pembelajaran, dan gradient clipping). Strategi penyetelan yang efektif: mulai dari pengaturan default yang masuk akal, gunakan model yang lebih kecil untuk melakukan iterasi dengan cepat, lakukan eksplorasi kasar-ke-halus (arsitektur terlebih dahulu, kemudian optimizer/regularisasi, kemudian penyetelan yang lebih detail), lebih suka pencarian acak daripada rentang kontinu yang besar daripada pencarian grid brute-force, gunakan learning rate sweeps (LR finder), terapkan early stopping, dan gunakan metode yang lebih canggih (optimasi Bayesian, Hyperband, pelatihan berbasis populasi) untuk pencarian yang mahal.
5. MLP (fully connected layers) mempelajari pemetaan nonlinier global dan cocok untuk data tabular dan perkiraan fungsi umum tetapi mengabaikan struktur spasial/temporal. CNN menggunakan konvolusi lokal, pembagian bobot, dan pooling untuk menangkap pola lokal spasial dan invariansi translasi – paling cocok untuk gambar, bingkai video, dan data spasial lainnya. RNN mempertahankan keadaan tersembunyi berulang yang diperbarui secara berurutan dan berbagi parameter sepanjang waktu untuk memodelkan urutan; mereka bekerja untuk tugas-tugas berurutan tetapi kesulitan dengan ketergantungan jarak jauh karena gradien yang menghilang/meledak. LSTM adalah varian RNN yang diberi gerbang (gerbang input/lupa/output dan keadaan sel) yang dirancang untuk menangkap ketergantungan jangka panjang dan mengurangi gradien yang menghilang – sangat cocok untuk urutan

yang lebih panjang dalam bahasa, ucapan, dan deret waktu. Transformer menggantikan rekurensi dengan perhatian diri, menghitung interaksi berpasangan di seluruh posisi secara paralel (dengan pengkodean posisi), memungkinkan pemodelan ketergantungan jarak jauh yang efisien dalam skala besar – dominan dalam NLP dan semakin banyak digunakan dalam tugas-tugas visi dan multimodal.

6. Latihan pemrograman (dasar) – implementasikan langkah maju (forward step) dari sebuah perceptron dengan fungsi sigmoid, tanh, ReLU, dan vector softmax, serta tunjukkan efek biasanya.

```
import numpy as np
def sigmoid(z):
    return 1.0 / (1.0 + np.exp(-z))
def tanh_act(z):
    return np.tanh(z)
def relu(z):
    return np.maximum(0.0, z)
def softmax_vec(z):
    # z: (N, C) or (N,) for 2-class trick; stable softmax
    z = np.asarray(z)
    if z.ndim == 1:
        z = z.reshape(-1, 1)
    z = z - np.max(z, axis=1, keepdims=True)
    e = np.exp(z)
    return e / np.sum(e, axis=1, keepdims=True)
Linear model: z = x·w + b
w = np.array([1.0, -1.0]) # example 2D weights
def forward(X, b):
    z = X.dot(w) + b # shape (N,)
    out = {
        'z': z,
        'sigmoid': sigmoid(z),
        'tanh': tanh_act(z),
        'relu': relu(z),
        # 2-class softmax constructed as softmax([z, -z])
        to get probabilities for two classes
        'softmax_2class': softmax_vec(np.vstack([z, -z]).T)
    }
    return out
if name == 'main':
    X = np.array([[0,0],[1,0],[0,1],[1,1]], dtype=float)
    for b in [-1.0, 0.0, 1.0]:
        print("bias =", b)
        o = forward(X, b)
        print("z:", np.round(o['z'],3))
```

```
print("sigmoid:", np.round(o['sigmoid'],3))
print("tanh:    ", np.round(o['tanh'],3))
print("relu:    ", np.round(o['relu'],3))
print("softmax (2-class):")
print(np.round(o['softmax_2class'],3))
print()
```

7. Latihan pemrograman (tingkat lanjut) – MLP dua lapis minimal dengan gradien eksplisit (input → ReLU tersembunyi → sigmoid keluaran), kerugian entropi silang, pembaruan SGD

```
import numpy as np
def sigmoid(z):
    return 1.0 / (1.0 + np.exp(-z))
def relu(z):
    return np.maximum(0.0, z)
def relu_prime(z):
    return (z > 0).astype(float)
Toy dataset (X: N x d, y: N x 1)
X = np.array([[0,0],[0,1],[1,0],[1,1]], dtype=float)
y = np.array([[0],[1],[1],[0]], dtype=float) # XOR (for demonstration)
Network sizes
rng = np.random.RandomState(0)
d = X.shape[1]
h = 4
out_dim = 1
W1 = rng.normal(scale=0.5, size=(d, h))
b1 = np.zeros((1, h))
W2 = rng.normal(scale=0.5, size=(h, out_dim))
b2 = np.zeros((1, out_dim))
lr = 0.1
epochs = 5000
m = X.shape[0]
eps = 1e-9
for epoch in range(epochs):
    # Forward
    z1 = X.dot(W1) + b1 # (N,h)
    a1 = relu(z1) # (N,h)
    z2 = a1.dot(W2) + b2 # (N,1)
    a2 = sigmoid(z2) # (N,1)
    # Loss: binary cross-entropy (mean)
    loss = -np.mean(y * np.log(a2 + eps) + (1 - y) * np.log(1 - a2 + eps))
    # Backward (explicit gradients)
    dz2 = (a2 - y) / m # dL/dz2 for BCE with sigmoid
    dW2 = a1.T.dot(dz2) # (h,1)
    db2 = np.sum(dz2, axis=0, keepdims=True)
    da1 = dz2.dot(W2.T) # (N,h)
    dz1 = da1 * relu_prime(z1) # (N,h)
    dW1 = X.T.dot(dz1) # (d,h)
    db1 = np.sum(dz1, axis=0, keepdims=True)
```

```
# Parameter update (SGD)
W2 -= lr * dW2
b2 -= lr * db2
W1 -= lr * dW1
b1 -= lr * db1
if epoch % 1000 == 0:
    print(f"epoch {epoch}, loss {loss:.6f}")
```

8. B. tanh

9. B. Gradien yang menghilang/meledak pada ketergantungan jangka panjang

10. C. Pencarian grid

11.]

BAB 9: JAWABAN

1. Pembelajaran Mesin (ML) biasanya belajar dari dataset statis berlabel atau tidak berlabel untuk memetakan input ke output (supervised dan unsupervised). Pembelajaran Penguatan (RL) belajar melalui interaksi dengan lingkungan, mengoptimalkan tindakan untuk memaksimalkan imbalan kumulatif. Perbedaan utama: RL mengoptimalkan pengambilan keputusan berurutan dan imbalan tertunda, menangani trade-off eksplorasi/eksploitasi, dan belajar dari umpan balik online daripada label tetap. RL lebih disukai ketika keputusan memengaruhi keadaan atau imbalan di masa depan – misalnya, robotika (kontrol kontinu, di mana tindakan mengubah keadaan robot), permainan (AlphaGo, di mana perencanaan jangka panjang penting), mesin rekomendasi dengan metrik tertunda (keterlibatan pengguna jangka panjang), dan mengemudi otonom (kontrol loop tertutup dan keputusan berurutan yang kritis terhadap keselamatan).
2. Istilah-istilah kunci RL: Agen: pembelajar/pengambil keputusan (robot). Lingkungan: sistem eksternal yang berinteraksi dengan agen (lantai pabrik). Keadaan: representasi lingkungan pada suatu waktu (posisi robot, pembacaan sensor). Tindakan: pilihan yang tersedia bagi agen (bergerak kiri/kanan, mengambil). Hadiah: umpan balik skalar untuk suatu tindakan dalam suatu keadaan (sinyal keberhasilan tugas). Triple SAR: (s, a, r) – keadaan, tindakan yang diambil, hadiah yang diterima. Kebijakan: pemetaan dari keadaan ke probabilitas tindakan ($\pi(a | s)$). Episode: urutan dari keadaan awal ke keadaan akhir (satu tugas pengambilan oleh robot). Tujuan: tujuan yang dikodekan oleh fungsi hadiah (memaksimalkan pengambilan tanpa tabrakan). Fungsi nilai: pengembalian kumulatif yang diharapkan dari suatu keadaan di bawah suatu kebijakan ($V^\pi(s) = \mathbb{E}_\pi[\sum_t \gamma^t r_t | s]$). Contoh interaksi: agen mengamati keadaan s , memilih tindakan $a \sim \pi(a | s)$, lingkungan mengembalikan r dan keadaan berikutnya s' , membentuk SAR, agen memperbarui kebijakan/nilai untuk meningkatkan imbalan kumulatif yang diharapkan.
3. Proses Keputusan Markov (MDP) adalah formalisme (S, A, P, R, γ) untuk RL di mana S adalah ruang keadaan, A tindakan, $P(s' | s, a)$ probabilitas transisi keadaan, $R(s, a, s')$ fungsi imbalan, dan γ faktor diskon. Sifat Markov menyatakan bahwa distribusi keadaan berikutnya dan imbalan hanya bergantung pada keadaan dan tindakan saat ini, bukan

pada riwayat sebelumnya; ini memungkinkan pemrograman dinamis yang berprinsip. Probabilitas transisi keadaan P mengkodekan dinamika lingkungan yang digunakan dalam perencanaan atau RL berbasis model. Persamaan Bellman memberikan konsistensi rekursif untuk fungsi nilai: $V^\pi(s) = \mathbb{E}_\pi[r + \gamma V^\pi(s')]$, dan untuk optimalitas $Q^*(s, a) = \mathbb{E}[r + \gamma \max_{a'} Q^*(s', a')]$. Rekursi Bellman mendasari pemrograman dinamis, pembaruan bootstrapping (TD), dan jaminan konvergensi untuk banyak algoritma RL.

4. (C) Mereka membutuhkan tugas episodik dengan episode yang berakhir
5. (B) Perbedaan antara nilai prediksi dan pengembalian aktual.
6. Latihan Pemrograman – pengembalian yang didiskon G_t

```
def discounted_return(rewards, gamma):
    """
    Compute discounted return G0.. using list of rewards
    and discount factor gamma.
    Returns list of G_t for each time t (optional). Here
    we return G0 (full return) and list.
    """
    G = 0.0
    returns = []
    # compute from end to start
    for r in reversed(rewards):
        G = r + gamma * G
        returns.insert(0, G)
    return returns # returns[t] == G_t

# Example
rewards =
gamma = 0.9
returns = discounted_return(rewards, gamma)
print("Returns per timestep:", returns)
print("G0 (total discounted return):", returns[0])
```

7. Latihan Pemrograman – Evaluasi Kebijakan Monte Carlo Kunjungan Pertama untuk grid 4-state sederhana.

```
import random
import numpy as np
from collections import defaultdict
# Simple 4-state grid world:
# states: 0,1,2,3
# 0 -- 1
# |    |
# 2 -- 3
# Terminal states: assume 3 is terminal with reward +1, 2
is terminal with reward -1 (example)
# Starting state: 0
# Actions: 0: right, 1: left, 2: down, 3: up
# Transitions deterministic for simplicity
```

```
next_state = {
    0: {0:1, 1:0, 2:2, 3:0}, # from 0: right->1, left-
>0(stay),
down->2
    1: {0:1, 1:0, 2:3, 3:1}, # from 1: down->3
    2: {0:3, 1:2, 2:2, 3:0}, # from 2: right->3, up->0
    3: {0:3, 1:3, 2:3, 3:3} # terminal stays
}
rewards = {
    (0,0,1): 0, (0,2,2): 0, (0,1,0): 0,
    (1,2,3): 0, (1,1,0): 0,
    (2,0,3): 0, (2,3,0): 0,
    (3,0,3): 1, (3,1,3): 1, (3,2,3): 1, (3,3,3): 1
}
# For simplicity give reward +1 only when reaching state 3
at episode end; -1 for reaching state 2 as terminal with
-1:
# Let's set 2 as terminal with -1, 3 terminal +1
terminal_states = {2: -1, 3: 1}
def step(s, a):
    ns = next_state[s].get(a, s)
    if ns in terminal_states:
        return ns, terminal_states[ns], True
    else:
        return ns, 0, False
def generate_episode(policy, max_steps=20):
    episode = [] # list of (s,a,r)
    s = 0 # start
    for t in range(max_steps):
        a = policy(s)
        ns, r, done = step(s,a)
        episode.append((s,a,r))
        s = ns
        if done:
            break
    return episode
def random_policy(s):
    return random.choice([0,1,2,3])
# First-visit Monte Carlo evaluation
num_episodes = 100 # use >=10; set 100 for better
estimates
gamma = 0.9
# storage
returns_sum = defaultdict(float)
```

```
returns_count = defaultdict(int)
V = defaultdict(float)
for ep in range(num_episodes):
    ep_traj = generate_episode(random_policy)
    G = 0.0
    # compute returns from end
    # collect first-visit indices
    first_visit = {}
    for i, (s,a,r) in enumerate(ep_traj):
        if s not in first_visit:
            first_visit[s] = i
    # compute returns for each first-visit state
    for s, first_i in first_visit.items():
        G = 0.0
        for t in range(first_i, len(ep_traj)):
            G = ep_traj[t][2] + gamma * G
        returns_sum[s] += G
        returns_count[s] += 1
        V[s] = returns_sum[s] / returns_count[s]
# Print estimated values for each state (non-visited states
may be missing)
for s in range(4):
    print(f"State {s}: V_est = {V.get(s,0.0):.3f}, visits =
{returns_count.get(s,0)}")
```

BAB 10: JAWABAN

1. RL mempelajari kebijakan dari interaksi lingkungan menggunakan metode berbasis nilai atau kebijakan dan efektif ketika ruang keadaan/aksi cukup kecil untuk aproksimator fungsi tabular atau berdimensi rendah. DRL menggunakan jaringan saraf dalam untuk mengaproksimasi fungsi nilai, kebijakan, atau model, memungkinkan pembelajaran langsung dari input berdimensi tinggi (gambar dan sensor mentah) dan generalisasi di seluruh ruang keadaan yang besar. DNN menyediakan ekstraksi fitur dan aproksimasi fungsi yang kuat tetapi memperkenalkan ketidakstabilan (aproksimasi fungsi nonlinier, pelupaan katastropik) dan persyaratan sampel/komputasi yang lebih besar. Oleh karena itu, DRL sangat cocok untuk masalah kompleks berdimensi tinggi di mana fitur buatan tangan atau metode tabular tidak memungkinkan.
2. Q-learning adalah algoritma off-policy, model-free yang mempelajari fungsi nilai aksi optimal $Q^*(s, a)$ dengan menerapkan pembaruan optimalitas Bellman secara iteratif $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$, di mana α adalah laju pembelajaran dan γ adalah faktor diskon. Persamaan Bellman menyediakan kondisi konsistensi rekursif (titik tetap) yang diupayakan Q-learning, menggunakan imbalan yang diamati r dan perkiraan pengembalian di masa depan $\max_{a'} Q(s', a')$. Eksplorasi-eksploitasi sangat penting: eksplorasi (misalnya, ϵ -greedy) memastikan agen

mengambil sampel pasangan keadaan-aksi yang beragam untuk menemukan imbalan yang lebih baik, sementara eksploitasi menggunakan perkiraan Q saat ini untuk memaksimalkan pengembalian. Eksplorasi yang tidak cukup menghasilkan kebijakan yang suboptimal; eksplorasi yang berlebihan memperlambat konvergensi. Penjadwalan yang tepat (peluruhan ϵ) menyeimbangkan penemuan di awal dan eksploitasi di kemudian hari.

3. Metode gradien kebijakan membuat parameter kebijakan $\pi_\theta(a | s)$ dan secara langsung mengoptimalkan keuntungan yang diharapkan dengan kenaikan gradien pada $\mathbb{E}[\sum_t R_t]$ menggunakan estimator seperti REINFORCE: $\nabla_\theta J \approx \mathbb{E}[\nabla_\theta \log \pi_\theta(a | s) \cdot G]$. Metode ini menangani tindakan berkelanjutan dan kebijakan stokastik namun dapat memiliki varian yang tinggi. Kerangka Aktor-Kritikus menggabungkan aktor yang diparameterisasi (kebijakan π_θ) dan kritikus (penaksir nilai V_ϕ atau Q_ϕ). Kritikus memperkirakan nilai untuk menghasilkan keunggulan varians yang lebih rendah atau kesalahan TD $A_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$, yang digunakan aktor untuk menskalakan pembaruan kebijakan ($(\nabla_\theta \log \pi_\theta \cdot A_t)$). Kritikus dilatih (misalnya dengan meminimalkan kesalahan TD), dan aktor diperbarui menggunakan sinyal kritikus. Kolaborasi ini mengurangi varians, meningkatkan efisiensi sampel, dan menstabilkan pembelajaran.
4. (B) Kemungkinan mengeksplorasi tindakan acak.
5. (B) Pengalaman Buffer Putar Ulang
6. Q-learning (tabel) dengan logging dan visualisasi nilai-Q

```
# q_learning_logging.py
import gym
import numpy as np
import matplotlib.pyplot as plt
import copy
import random

env = gym.make("FrozenLake-v1", is_slippery=False) #
deterministic for clarity
seed = 0
env.reset(seed=seed)
np.random.seed(seed)
random.seed(seed)

n_states = env.observation_space.n
n_actions = env.action_space.n
# Q-learning hyperparams
alpha = 0.1
gamma = 0.99
epsilon_start = 1.0
epsilon_end = 0.05
epsilon_decay = 0.995
```

```
episodes = 5000
snapshot_interval = 100 # log Q every 100 episodes
# Initialize Q table
Q = np.zeros((n_states, n_actions))
# Choose a representative state-action to track (state 0,
action 0)
tracked_pairs = [(0, 0), (1, 0), (2, 0)]
snapshots = {pair: [] for pair in tracked_pairs}
snapshot_episodes = []
epsilon = epsilon_start
for ep in range(1, episodes + 1):
    state = env.reset()[0]
    done = False
    while not done:
        if np.random.rand() < epsilon:
            action = env.action_space.sample()
        else:
            action = np.argmax(Q[state])
        next_state, reward, terminated, truncated, _ =
env.step(action)
        done = terminated or truncated
        # Q update
        best_next = np.max(Q[next_state])
        td = reward + gamma * best_next - Q[state, action]
        Q[state, action] += alpha * td
        state = next_state
    # decay epsilon
    epsilon = max(epsilon_end, epsilon * epsilon_decay)
    if ep % snapshot_interval == 0:
        snapshot_episodes.append(ep)
        for pair in tracked_pairs:
            s, a = pair
            snapshots[pair].append(Q[s, a])
# Plot tracked Q-values
plt.figure(figsize=(8, 5))
for pair, vals in snapshots.items():
    plt.plot(snapshot_episodes, vals, label=f"Q(s={pair[0]
},a={pair[1]})")
plt.xlabel("Episode")
plt.ylabel("Q-value")
plt.title("Tracked Q-values over Training (snapshots every
100 episodes)")
plt.legend()
plt.grid(True)
```

```
plt.show()
```

7. Aktor-Kritikus (berdasarkan kebijakan) untuk CartPole (PyTorch)

```
# actor_critic_cartpole.py
import gym
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
import matplotlib.pyplot as plt
import random
# Reproducibility
seed = 42
torch.manual_seed(seed)
np.random.seed(seed)
random.seed(seed)
env = gym.make("CartPole-v1")
env.reset(seed=seed)
obs_dim = env.observation_space.shape[0]
n_actions = env.action_space.n
# Simple Actor and Critic networks
class Actor(nn.Module):
    def __init__(self, obs_dim, n_actions, hidden=128):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(obs_dim, hidden),
            nn.ReLU(),
            nn.Linear(hidden, n_actions),
            nn.Softmax(dim=-1)
        )
    def forward(self, x):
        return self.net(x)
class Critic(nn.Module):
    def __init__(self, obs_dim, hidden=128):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(obs_dim, hidden),
            nn.ReLU(),
            nn.Linear(hidden, 1)
        )
    def forward(self, x):
        return self.net(x).squeeze(-1)
device = torch.device("cuda" if torch.cuda.is_available()
else "cpu")
actor = Actor(obs_dim, n_actions).to(device)
critic = Critic(obs_dim).to(device)
actor_optim = optim.Adam(actor.parameters(), lr=1e-3)
```

```
critic_optim = optim.Adam(critic.parameters(), lr=1e-3)
# Training hyperparams
gamma = 0.99
episodes = 500
print_every = 10
episode_rewards = []
for ep in range(1, episodes + 1):
    state = env.reset()[0]
    done = False
    ep_reward = 0
    states = []
    actions = []
    rewards = []
    while not done:
        state_t = torch.tensor(state, dtype=torch.float32).
to(device)
        probs = actor(state_t)
        m = torch.distributions.Categorical(probs)
        action = m.sample().item()
        next_state, reward, terminated, truncated, _ =
env.step(action)
        done = terminated or truncated
        states.append(state_t)
        actions.append(action)
        rewards.append(reward)
        ep_reward += reward
        state = next_state
    episode_rewards.append(ep_reward)
    # Compute returns and update networks (one-episode
on-policy)
    # Using TD(1) style with returns
    R = 0
    returns = []
    for r in reversed(rewards):
        R = r + gamma * R
        returns.insert(0, R)
    returns = torch.tensor(returns, dtype=torch.float32).
to(device)
    states_tensor = torch.stack(states)
    actions_tensor = torch.tensor(actions, dtype=torch.
int64).to(device)
    values = critic(states_tensor)
    # Advantage = returns - values
    advantages = returns - values.detach()
    # Actor loss: -log_prob * advantage
    probs = actor(states_tensor)
    dist = torch.distributions.Categorical(probs)
    log_probs = dist.log_prob(actions_tensor)
```

```
actor_loss = -(log_probs * advantages).mean()
# Critic loss: MSE between returns and values
critic_loss = nn.functional.mse_loss(values, returns)
# Optimize
actor_optim.zero_grad()
actor_loss.backward()
actor_optim.step()
critic_optim.zero_grad()
critic_loss.backward()
critic_optim.step()
if ep % print_every == 0:
    avg_last=np.mean(episode_rewards[-print_every:])
    print(f"Episode {ep}\tAverage Reward (last {print_
every}): {avg_last:.2f}")
# Plot rewards
plt.figure(figsize=(8, 5))
plt.plot(episode_rewards, label="Episode reward")
# moving average
window = 20
ma = np.convolve(episode_rewards, np.ones(window)/window,
mode='valid')
plt.plot(range(window-1, episodes), ma, label=f"{window}-epi-
sode MA")
plt.xlabel("Episode")
plt.ylabel("Reward")
plt.title("Actor-Critic on CartPole-v1: Episode Rewards")
```

BAB 11: JAWABAN

1. Lima fase NLP adalah: fase Leksikal melakukan tokenisasi, normalisasi, dan analisis morfologi untuk mengidentifikasi bentuk kata dan lema; fase Sintaksis menentukan struktur gramatikal (misalnya, penandaan POS dan penguraian konstituen/ketergantungan) untuk mengungkapkan hubungan antar token; fase Semantik memetakan struktur sintaksis ke representasi makna (disambiguasi makna kata, pelabelan peran semantik, dan struktur predikat-argumen); Integrasi Wacana menyelesaikan makna di seluruh kalimat (resolusi anafora/koreferensi, relasi koherensi) sehingga ujaran bergabung menjadi interpretasi yang konsisten; Analisis Pragmatik menempatkan bahasa dalam konteks dan maksud (implikatur, tindak tutur, maksud pengguna, pengetahuan dunia) untuk memilih interpretasi dan respons dunia nyata yang sesuai. Setiap fase dibangun di atas fase sebelumnya untuk mengubah teks mentah menjadi pemahaman yang dapat ditindaklanjuti.
2. NLP berbasis aturan menggunakan aturan linguistik dan gazetteer yang dibuat secara manual; Ia menawarkan interpretasi, kontrol yang tepat dalam domain yang terdefinisi dengan baik, dan perilaku yang dapat diprediksi dengan data kecil, tetapi rapuh, mahal untuk diskalakan, dan kesulitan dengan ambiguitas dan cakupan. Pendekatan pembelajaran mesin (ML) mempelajari pola dari data, menawarkan ketahanan,

adaptabilitas, dan skalabilitas yang lebih baik (terutama dengan korpus besar dan model mendalam), tetapi membutuhkan data berlabel, dapat buram, dan mungkin kurang baik dalam generalisasi di luar distribusi pelatihan. Gunakan metode berbasis aturan ketika pengetahuan domain eksplisit, data langka, dan kebenaran sangat penting (misalnya, templat hukum); gunakan pendekatan ML/mendalam ketika data yang cukup tersedia dan cakupan/fleksibilitas serta pembelajaran pola kompleks menjadi prioritas (misalnya, IR skala besar dan agen percakapan).

3. NLP klasik menekankan rekayasa fitur dan model probabilistik/statistik: contohnya termasuk model bahasa n-gram, Model Markov Tersembunyi untuk penandaan POS, bidang acak bersyarat untuk pelabelan urutan, dan parser berbasis aturan. Deep Learning NLP menggantikan fitur manual dengan representasi terdistribusi yang dipelajari dan model end-to-end: contohnya termasuk word embedding (Word2Vec, GloVe), RNN/LSTM untuk pemodelan urutan, arsitektur berbasis Transformer (BERT dan GPT) untuk representasi kontekstual dan transfer learning. Metode klasik ringan dan mudah diinterpretasikan; metode deep learning biasanya mencapai kinerja superior pada tugas data besar dan memungkinkan transfer learning tetapi membutuhkan lebih banyak komputasi dan data.
4. (b) Mengidentifikasi dan mengklasifikasikan kata benda khusus seperti nama, organisasi, dan lokasi.
5. (c) Ukuran seberapa baik model memprediksi sampel.
6. Latihan pemrograman dasar – tokenisasi dan penghapusan stopwords menggunakan NLTK

```
# Requires: pip install nltk
import nltk
nltk.download('punkt', quiet=True)
nltk.download('stopwords', quiet=True)
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import string

def clean_tokens(paragraph):
    tokens = word_tokenize(paragraph)
    tokens = [t.lower() for t in tokens if t.isalpha()] #
remove punctuation/numbers
    stops = set(stopwords.words('english'))
    cleaned = [t for t in tokens if t not in stops]
    return cleaned

# Example
text = "This is an example paragraph. It demonstrates
tokenization and stopwords removal!"
print(clean_tokens(text))
```

7. Latihan pemrograman tingkat lanjut – model bigram dengan MLE dan penghalusan Laplace

```
# Minimal bigram LM with Laplace smoothing
from collections import defaultdict, Counter
import math

class BigramLanguageModel:
    def __init__(self, corpus_sentences):
        # corpus_sentences: list of token lists (already
        # tokenized, with <s> and </s> if desired)
        self.unigram_counts = Counter()
        self.bigram_counts = defaultdict(Counter)
        for sent in corpus_sentences:
            tokens = ['<s>'] + sent + ['</s>']
            for i in range(len(tokens)):
                self.unigram_counts[tokens[i]] += 1
                if i < len(tokens)-1:
                    self.bigram_counts[tokens[i]][tokens[i+1]]
                    += 1

            self.vocab = set(self.unigram_counts.keys())
            self.V = len(self.vocab)

        def bigram_prob(self, w_prev, w_curr, alpha=1.0):
            # Laplace smoothing: (count(w_prev,w_curr)+alpha)
            / (count(w_prev)+alpha*V)
            num = self.bigram_counts[w_prev][w_curr] + alpha
            den = self.unigram_counts[w_prev] + alpha *
            self.V
            return num / den

        def sentence_log_prob(self, sentence_tokens,
            alpha=1.0):
            tokens = ['<s>'] + sentence_tokens + ['</s>']
            logp = 0.0
            for i in range(len(tokens)-1):
                prob = self.bigram_prob(tokens[i], tokens[i+1],
                    alpha)
                logp += math.log(prob)
            return logp

        def sentence_prob(self, sentence_tokens, alpha=1.0):
            return math.exp(self.sentence_log_prob(sentence_
            tokens, alpha))

# Example usage
corpus = [
    ["the", "quick", "brown", "fox"],
    ["the", "lazy", "dog"],
    ["the", "fox", "jumps"]
```

```
]
model = BigramLanguageModel(corpus)
test = ["the", "fox", "jumps"]
print("Log-prob:", model.sentence_log_prob(test,
alpha=1.0))
print("Prob:", model.sentence_prob(test, alpha=1.0))
```

BAB 12: JAWABAN

1. Penerjemahan Mesin Statistik (SMT) dan Penerjemahan Mesin Neural (NMT) mewakili paradigma yang berbeda dalam penerjemahan. SMT bergantung pada model statistik yang berasal dari korpus paralel besar, biasanya melibatkan analisis linguistik kompleks, rekayasa fitur, dan komponen modular seperti model penerjemahan dan bahasa. SMT seringkali kesulitan dengan ketergantungan jarak jauh dan membutuhkan upaya manusia yang signifikan untuk aturan spesifik bahasa. Sebaliknya, NMT menggunakan arsitektur jaringan neural ujung-ke-ujung, mempelajari pemetaan langsung dari urutan input ke output. Model NMT awal, berdasarkan Jaringan Neural Berulang (RNN) dan Memori Jangka Pendek Panjang (LSTM), dapat mempelajari representasi dan ketergantungan yang lebih abstrak tetapi menghadapi tantangan dengan hambatan informasi ketika mengkodekan kalimat panjang ke dalam vektor konteks berukuran tetap. Pengenalan mekanisme perhatian secara fundamental merevolusi NMT dengan memungkinkan model untuk secara dinamis menimbang pentingnya berbagai bagian kalimat sumber ketika menghasilkan setiap kata dari kalimat target. Terobosan ini secara signifikan meningkatkan kualitas terjemahan, khususnya untuk kalimat panjang, dengan mengatasi hambatan konteks tetap dan memungkinkan model untuk mempelajari keselarasan lunak antara kata sumber dan kata target, yang mengarah pada terjemahan yang lebih lancar dan akurat.
2. Arsitektur Word2Vec (CBOW dan Skip-gram): Word2Vec menawarkan dua arsitektur utama untuk mempelajari embedding kata: Continuous Bag-of-Words (CBOW) dan Skip-gram. CBOW bertujuan untuk memprediksi kata saat ini berdasarkan kata-kata konteks di sekitarnya. Tujuan pelatihannya adalah untuk memaksimalkan probabilitas kata target berdasarkan jumlah atau penggabungan vektor kata konteksnya. Secara komputasi, CBOW umumnya lebih cepat dilatih untuk dataset besar dan kata-kata yang sering muncul, sehingga cocok untuk embedding kata tujuan umum di mana kecepatan menjadi prioritas. Sebaliknya, Skip-gram memprediksi kata-kata konteks di sekitarnya berdasarkan satu kata saat ini (pusat). Tujuan pelatihannya adalah untuk memaksimalkan probabilitas kata-kata konteks yang muncul di sekitar kata target. Skip-gram biasanya lebih lambat untuk dilatih tetapi berkinerja lebih baik dengan dataset pelatihan yang lebih kecil dan lebih efektif dalam mempelajari embedding untuk kata-kata langka, sering kali menangkap hubungan semantik yang lebih bernuansa.
3. GloVe (Global Vectors for Word Representation) membedakan dirinya dari model prediktif murni seperti Word2Vec dengan memanfaatkan statistik kemunculan

bersama kata-kata global dari seluruh korpus. Tujuan optimasinya adalah untuk mempelajari vektor kata sedemikian rupa sehingga hasil perkalian titiknya sebanding dengan logaritma probabilitas kemunculan bersama mereka. Secara khusus, ia meminimalkan fungsi kerugian yang berupaya membuat hasil perkalian titik dari vektor dua kata apa pun sama dengan logaritma dari jumlah kali mereka muncul bersama, secara opsional diberi bobot oleh fungsi yang memberikan bobot lebih rendah pada kemunculan bersama yang sangat sering. Perspektif global ini memungkinkan GloVe untuk menangkap substruktur linier yang lebih canggih dalam ruang embedding kata, yang bermanifestasi sebagai struktur analogis. Misalnya, hubungan vektor antara “raja” dan “pria” sering kali mencerminkan hubungan antara “ratu” dan “wanita”. Hal ini muncul karena perbedaan proporsional dalam jumlah kemunculan bersama (misalnya, “raja” muncul bersama dengan “kerajaan” lebih sering daripada “pria”, dan “ratu” muncul bersama dengan “kerajaan” lebih sering daripada “wanita”) secara inheren dikodekan ke dalam hubungan vektor yang dipelajari.

4. Model Transformer dibangun di atas struktur encoder-decoder. Encoder memproses urutan input (misalnya, kalimat bahasa sumber), mengubahnya menjadi representasi kontinu. Ini terdiri dari tumpukan lapisan identik, masing-masing berisi mekanisme self-attention multi-head dan jaringan feed-forward position-wise. Decoder menghasilkan urutan output (misalnya, kalimat bahasa target) satu token pada satu waktu. Algoritma ini juga terdiri dari tumpukan lapisan identik, masing-masing memiliki mekanisme self-attention multi-head bertopeng, mekanisme attention encoder-decoder (silang), dan jaringan feed-forward posisi-per-posisi. Self-attention sangat penting, memungkinkan setiap kata dalam suatu urutan untuk mempertimbangkan pentingnya semua kata lain dalam urutan yang sama ketika menghitung representasinya sendiri. Hal ini secara efektif menangkap ketergantungan jarak jauh tanpa bergantung pada pemrosesan sekuensial. Multi-head attention memperluas self-attention dengan menjalankan beberapa mekanisme attention secara paralel (masing-masing dengan proyeksi linier yang berbeda dari matriks query, key, dan value).

Output dari "kepala" ini kemudian digabungkan dan ditransformasikan secara linier, memungkinkan model untuk secara bersamaan memperhatikan informasi dari subruang representasi yang berbeda pada posisi yang berbeda. Terakhir, embedding posisional ditambahkan ke embedding kata (yang mengubah token diskrit menjadi representasi vektor padat). Karena arsitektur Transformer secara inheren invarian terhadap permutasi karena pemrosesan paralelnya, embedding posisional memberikan informasi penting tentang posisi relatif atau absolut token dalam urutan, memungkinkan model untuk memahami urutan kata.

5. Evolusi LLM dan Kekuatan/Tantangannya: Evolusi Model Bahasa Besar (LLM) ditandai dengan peningkatan skala, kecanggihan arsitektur, dan kemampuan yang muncul. Inovasi awal seperti ELMo (Embeddings from Language Models) memperkenalkan embedding kata kontekstual, melampaui representasi statis. Munculnya arsitektur

Transformer menghasilkan model seperti BERT (Bidirectional Encoder Representations from Transformers), yang unggul dalam tugas pemahaman bahasa melalui pra-pelatihan dua arah (misalnya, pemodelan bahasa bertopeng), dan GPT (Generative Pre-trained Transformer), yang mempelopori pra-pelatihan generatif skala besar untuk pembangkitan teks. Versi selanjutnya seperti GPT-2 dan GPT-3 secara dramatis meningkatkan ukuran model dan data pelatihan, menunjukkan kemampuan pembelajaran few-shot dan zero-shot yang mengesankan serta menghasilkan teks yang sangat koheren dan beragam. Generasi terbaru, yang dicontohkan oleh GPT-4 dan ChatGPT, telah disempurnakan lebih lanjut melalui teknik seperti penyesuaian instruksi dan Pembelajaran Penguatan dari Umpan Balik Manusia (RLHF), yang memungkinkan mereka untuk mengikuti instruksi yang kompleks, terlibat dalam interaksi percakapan yang bernuansa, dan menunjukkan peningkatan keamanan dan keselarasan dengan preferensi manusia. Dua kekuatan utama dari LLM canggih ini adalah:

1. **Fleksibilitas dan Generalisasi:** Mereka dapat melakukan berbagai tugas bahasa alami (misalnya, penerjemahan, peringkasan, menjawab pertanyaan, pembuatan kode, penulisan kreatif) dengan kemahiran yang luar biasa, seringkali tanpa penyesuaian khusus tugas secara eksplisit.
 2. **Kemampuan yang Muncul:** Skala mereka yang sangat besar telah menghasilkan kemampuan yang tidak terduga, seperti penalaran dasar, mengikuti instruksi yang kompleks, dan pemecahan masalah yang canggih, yang tidak diprogram secara eksplisit tetapi muncul dari pola luas yang dipelajari selama pelatihan awal.
1. Dua tantangan signifikan meliputi:
1. **Halusinasi dan Ketidakakuratan Faktual:** LLM dapat menghasilkan informasi yang sangat masuk akal tetapi tidak akurat secara faktual atau sepenuhnya dibuat-buat, yang menimbulkan risiko dalam aplikasi kritis.
 2. **Bias dan Keadilan:** Dilatih pada sejumlah besar data internet, LLM pasti menginternalisasi dan dapat melanggengkan atau bahkan memperkuat bias sosial yang ada dalam data tersebut, yang menyebabkan keluaran yang tidak adil, diskriminatif, atau berbahaya, terutama menyangkut kelompok yang dilindungi. Tantangan lainnya termasuk biaya komputasi yang tinggi dan kurangnya pemahaman yang sebenarnya atau penalaran akal sehat.

6. [Latihan Pemrograman – Dasar] Implementasi Perhatian Produk Titik Skala:

```
import torch
import math
def scaled_dot_product_attention(Q, K, V):
    """
    Computes scaled dot-product attention for a single head.
    Args:
    Q (torch.Tensor): Query tensor (batch_size, seq_len_q,
    d_k)
    K (torch.Tensor): Key tensor (batch_size, seq_len_kv,
    d_k)
```

```
V (torch.Tensor): Value tensor (batch_size, seq_len_kv,
d_v)
Returns:
torch.Tensor: Output tensor (batch_size, seq_len_q,
d_v)
torch.Tensor: Attention weights (batch_size, seq_len_q,
seq_len_kv)
"""
d_k = Q.size(-1)
# 1. Compute dot product of Q and K^T
scores = torch.matmul(Q, K.transpose(-2, -1)) # (batch_
size, seq_len_q, seq_len_kv)
# 2. Scale by 1/sqrt(d_k)
scaled_scores = scores / math.sqrt(d_k)
# 3. Apply softmax to get attention weights
attention_weights = torch.softmax(scaled_scores, dim=-1)
# 4. Multiply weights by V
output = torch.matmul(attention_weights, V) # (batch_
size, seq_len_q, d_v)
return output, attention_weights
# Example usage:
batch_size = 2
seq_len_q = 3
seq_len_kv = 4
d_k = 64 # Dimension of queries and keys
d_v = 32 # Dimension of values
Q = torch.randn(batch_size, seq_len_q, d_k)
K = torch.randn(batch_size, seq_len_kv, d_k)
V = torch.randn(batch_size, seq_len_kv, d_v)
output, weights = scaled_dot_product_attention(Q, K, V)
print("Output shape:", output.shape)
print("Attention weights shape:", weights.shape)
# Explanation of scaling:
# The scaling factor of 1/sqrt(d_k) stabilizes the soft-
max operation.
# Without scaling, as the dimension d_k increases, the
dot products (Q * K^T) can become very large in magnitude
(either positive or negative).
# When these large values are passed into the softmax
function, the gradients can become extremely small (satu-
rating the softmax output to 0 or 1),
# leading to vanishing gradients and hindering effective
learning.
# By dividing by the square root of d_k, the variance of
the dot products is reduced,
# keeping the inputs to the softmax in a more stable
range,
# which allows for more effective gradient flow during
training.
```

7. [Latihan Pemrograman – Tingkat Lanjut] Loop Pelatihan Skip-gram Minimal dengan Pengambilan Sampel Negatif:

8. B. Dekoder menambahkan perhatian encoder-decoder untuk memperhatikan output encoder.
9. C. Bagi dengan $\sqrt{d_k}$ untuk menstabilkan gradien/softmax.
10. B. GloVe memfaktorkan struktur ko-okurensi global untuk mempelajari embedding.

BAB 13: JAWABAN

1. Pola desain, dalam konteks sistem AI terdistribusi, adalah templat yang diformalkan dan dapat digunakan kembali yang menawarkan solusi terbukti untuk masalah arsitektur dan pemrograman umum. Penerapannya secara signifikan meningkatkan efisiensi pengembangan dan kualitas sistem. Pola-pola ini meningkatkan:
 - **Kemampuan penggunaan kembali:** Dengan menyediakan solusi standar yang dapat diterapkan di berbagai konteks dan proyek, mengurangi upaya yang berlebihan.
 - **Kemudahan pemeliharaan:** Melalui adopsi struktur yang mapan, yang menyederhanakan debugging, pembaruan, dan modifikasi di masa mendatang.
 - **Skalabilitas:** Dengan memfasilitasi distribusi sumber daya dan modularitas yang lebih baik, memungkinkan sistem untuk menangani beban kerja yang meningkat secara lebih efektif.
2. Pipeline Big Data OCI sering kali menghadapi tantangan fungsional dan non-fungsional yang menghambat operasinya yang optimal:
 - **Tantangan Fungsional:**
 - **Keterbatasan Sumber Daya:** Batasan ukuran memori dapat mempersulit strategi partisi data, yang berdampak pada kemampuan untuk memproses dataset besar secara efisien.
 - **Latensi:** Ketergantungan pada API HTTP untuk komunikasi antar komponen dapat menimbulkan penundaan yang signifikan, memperlambat pemrosesan data secara keseluruhan dan analitik waktu nyata.
 - **Tantangan Non-Fungsional:**
 - **Inkonsistensi Data (NF11):** Eksekusi tugas dapat gagal karena karakter khusus atau data yang salah format dalam kolom tabel sumber, yang membahayakan integritas data dan keandalan pipeline.
 - **Stabilitas Aplikasi (NF22):** Kesalahan dalam Spark UI yang terjadi saat aplikasi Spark dihentikan menunjukkan masalah stabilitas yang mendasar, yang dapat menghambat upaya pemantauan dan pemecahan masalah. Tantangan-tantangan ini secara kolektif membahayakan skalabilitas, keandalan, dan efisiensi operasional pipeline.
3. **Menjelaskan Bahasa Pola untuk Sistem AI Terdistribusi** Bahasa pola melibatkan penggabungan beberapa pola desain yang saling terkait untuk mengatasi tantangan arsitektur sistem yang kompleks secara holistik. Pendekatan ini memungkinkan terciptanya solusi yang tangguh yang mengintegrasikan berbagai aspek sistem AI terdistribusi. Misalnya, penggabungan pola seperti Multisource Extractor (untuk

- penyerapan data), Lightweight Stateless (untuk pemrosesan yang efisien), dan Workflow Pipeline (untuk orkestrasi) dapat secara kolaboratif meningkatkan penyerapan data, pemrosesan, dan alur kerja sistem secara keseluruhan. Adopsi pola terintegrasi tersebut memastikan solusi komprehensif yang menyelaraskan kemampuan teknis dengan persyaratan bisnis tertentu, sehingga meningkatkan ketangguhan dan skalabilitas.
4. (c) Menawarkan akses data independen platform melalui layanan web tanpa menyimpan status
 5. (c) Penyeimbangan Beban Kerja

BAB 14: JAWABAN

1. Ancaman, Kerentanan, Serangan, dan Pertahanan

- **Ancaman:** Penyebab potensial suatu insiden yang dapat mengakibatkan kerugian bagi sistem atau organisasi (misalnya, phishing, ancaman dari dalam, atau aktor yang bermusuhan).
- **Kerentanan:** Kelemahan atau kekurangan dalam desain, implementasi, atau operasi sistem yang dapat dieksploitasi (misalnya, perangkat lunak yang belum ditambal, kurangnya sanitasi input, atau ketergantungan yang berlebihan pada data pelatihan yang tidak terverifikasi).
- **Serangan:** Eksploitasi kerentanan secara sengaja untuk mewujudkan ancaman (misalnya, DDoS, peracunan data, atau inversi model).
- **Pertahanan:** Serangkaian tindakan pencegahan, strategi, dan teknologi yang digunakan untuk mencegah, mendeteksi, dan menanggapi serangan (misalnya, kontrol akses, pelatihan musuh, dan pemantauan).

Konsep-konsep ini membentuk siklus keamanan: ancaman mengeksploitasi kerentanan untuk meluncurkan serangan, sementara pertahanan diimplementasikan untuk mengurangi kerentanan dan memblokir atau meminimalkan dampak serangan.

2. Pemodelan Ancaman AI vs. Pemodelan Perangkat Lunak: Pemodelan ancaman AI berbeda dari pemodelan perangkat lunak tradisional karena sistem AI bersifat probabilistik, bergantung pada data, dan sering beroperasi sebagai "kotak hitam." Tidak seperti perangkat lunak deterministik, AI sangat rentan terhadap peracunan data, masukan musuh, dan pergeseran model. Untuk mengadaptasi model **STRIDE** untuk AI:

- **Spoofing:** Menilai apakah penyerang dapat meniru sumber data.
- **Perusakan:** Mengevaluasi risiko terhadap integritas data pelatihan atau bobot model.
- **Penolakan:** Memastikan tindakan yang diambil oleh AI atau kontributor data dapat diaudit.
- **Pengungkapan Informasi:** Melindungi dari ekstraksi model atau rekonstruksi data pelatihan.

- **Penolakan Layanan:** Mencegah kehabisan sumber daya melalui kueri inferensi yang kompleks.
 - **Peningkatan Hak Akses:** Memastikan model tidak dapat dimanipulasi untuk melewati kontrol akses. Adaptasi ini memerlukan perlakuan data, model, dan pipeline ML sebagai aset penting dan melibatkan tim yang beragam untuk memetakan permukaan serangan yang unik.
3. **Jenis Serangan pada sistem AI:** Serangan diklasifikasikan berdasarkan tingkat pengetahuan penyerang:
- **Kotak Putih:** Penyerang memiliki akses penuh ke arsitektur model, parameter, dan data pelatihan.
 - **Kotak Abu-abu:** Penyerang memiliki pengetahuan parsial tentang sistem.
 - **Kotak Hitam:** Penyerang tidak memiliki pengetahuan internal dan hanya berinteraksi melalui input dan output (API).
 - **Jenis serangan utama berdasarkan fase siklus hidup:**
 - **Fase Pelatihan:** Peracunan (merusak data pelatihan) dan serangan Pintu Belakang (menyuntikkan pemicu tersembunyi).
 - **Fase Inferensi:** Penghindaran (contoh-contoh yang merugikan yang menyebabkan kesalahan klasifikasi), Inferensi Keanggotaan (menentukan apakah data spesifik digunakan dalam pelatihan), Inversi Model (merekonstruksi data pelatihan), dan Ekstraksi Model (mencuri fungsionalitas model).
4. **Pertahanan Terhadap Penghindaran dan Ekstraksi Informasi**
- **Kontrol Proaktif:**
 - **Pelatihan Adversarial:** Menggabungkan contoh adversarial ke dalam set pelatihan untuk meningkatkan ketahanan.
 - **Distilasi Defensif:** Melatih model untuk memprediksi probabilitas daripada label keras untuk menghaluskan gradien.
 - **Privasi Diferensial:** Menambahkan noise ke data pelatihan atau gradien untuk mencegah ekstraksi informasi (menjaga privasi).
 - **Regularisasi:** Menggunakan teknik seperti Dropout atau regularisasi L_1/L_2 untuk mengurangi overfitting.
 - **Kontrol Reaktif:**
 - **Validasi Input:** Mendeteksi dan menolak input Out-of-Distribution (OOD).
 - **Pemantauan Stateful:** Menganalisis pola kueri untuk mendeteksi upaya ekstraksi (misalnya, kueri frekuensi tinggi dan serupa).
 - **Metode Penjelasan:** Menggunakan alat seperti SHAP atau LIME untuk mengaudit dan mendeteksi batas keputusan anomali.
4. **Praktik Terbaik untuk Mengurangi Risiko Peracunan dan Backdoor:**
- **Asal Usul Data:** Pastikan rantai pengawasan yang terkontrol ketat untuk semua data pelatihan.
 - **Sanitasi:** Saring dan validasi data secara ketat untuk menghilangkan outlier atau sampel yang mencurigakan.

- **Lingkungan Terkendali:** Latih model di lingkungan yang aman dan terisolasi untuk mencegah akses yang tidak sah.
 - **Augmentasi Data:** Diversifikasi data pelatihan untuk mengurangi dampak sampel yang diracuni.
 - **Metode Evaluasi:**
 - **Pembersihan Neural:** Teknik untuk merekayasa balik pemicu potensial.
 - **Analisis Spektral:** Menganalisis representasi internal data untuk menemukan anomali yang khas dari sampel yang diracuni.
 - **Evaluasi Adversarial:** Menggunakan rangkaian pengujian (misalnya, AutoAttack) untuk menguji model terhadap vektor serangan yang diketahui.
5. B. Privasi Diferensial
 6. B. Inferensi Keanggotaan
 7. B. Pelatihan; Kesimpulan

BAB 15: JAWABAN

1. Keamanan sistem AI mencakup kumpulan protokol, pedoman, dan metodologi yang dirancang untuk memastikan bahwa kecerdasan buatan dikembangkan dan diterapkan dengan cara yang dapat dipercaya, bermanfaat, dan konsisten dengan nilai-nilai manusia. Tujuan utamanya adalah untuk meminimalkan kerugian yang diakibatkan oleh bias, penyalahgunaan, perilaku yang muncul, dan kerentanan keamanan. Di domain yang berdampak tinggi seperti perawatan kesehatan, transportasi, dan keuangan, langkah-langkah keamanan yang ketat – termasuk pengujian menyeluruh, transparansi, dan tata kelola etis – sangat penting. Karena AI beroperasi dengan kecepatan dan skala tinggi, setiap ketidaksesuaian atau cacat teknis di sektor-sektor ini dapat memperburuk efek buruk, yang menyebabkan kerugian finansial yang signifikan atau bahaya fisik.
2. **Aspek Utama Keamanan Sistem AI:** Bab ini menyoroti lima aspek utama keamanan AI:
 1. **Keselarasan dan Etika:** Memastikan tujuan dan perilaku sistem tetap konsisten dengan nilai-nilai manusia dan standar etika.
 2. **Ketangguhan dan Keandalan:** Menjamin bahwa sistem berfungsi sebagaimana mestinya, bahkan ketika menghadapi kondisi yang merugikan, kasus ekstrem, atau pergeseran data.
 3. **Transparansi dan Kemudahan Penjelasan:** Merancang sistem di mana proses pengambilan keputusan dapat dipahami, dilacak, dan diaudit oleh manusia.
 4. **Keamanan:** Menerapkan pertahanan terhadap serangan berbahaya, akses tidak sah, dan pelanggaran data.
 5. **Tata Kelola dan Praktik Terbaik:** Menetapkan kerangka kerja untuk pengembangan yang bertanggung jawab melalui peraturan, kebijakan, dan kontrol siklus hidup.
3. **Empat Prinsip Keamanan AI:** Empat prinsip inti berinteraksi untuk menciptakan kerangka kerja keamanan yang kohesif:

- **Keselarasan:** Berfokus pada memastikan tujuan AI berkembang dari waktu ke waktu agar tetap konsisten dengan niat dan nilai-nilai manusia.
 - **Ketangguhan:** Memastikan kinerja yang konsisten di bawah tekanan, serangan, atau perubahan lingkungan melalui pengujian yang ketat dan toleransi kesalahan.
 - **Transparansi:** Meningkatkan kepercayaan pengguna dan kemampuan audit dengan membuat perilaku sistem yang kompleks mudah dipahami, meskipun beberapa model bersifat "kotak hitam".
 - **Akuntabilitas:** Menetapkan standar, mekanisme pengawasan, dan prosedur untuk menetapkan tanggung jawab atas hasil sistem.
 - **Interaksi:** Bersama-sama, prinsip-prinsip ini menciptakan lingkaran umpan balik positif. Transparansi memungkinkan pengamatan Keselarasan dan Ketahanan, sementara Akuntabilitas memastikan bahwa penyimpangan dari prinsip-prinsip ini dikoreksi dan ditegakkan.
4. **Tantangan dan Strategi Mitigasi:** Bab ini mengidentifikasi beberapa tantangan utama dan mitigasi yang sesuai:
- **Ketidakselarasan dan Pergeseran Tujuan:** *Mitigasi:* Terapkan pemantauan berkelanjutan dan teknik seperti Pembelajaran Penguatan Terbalik (Inverse Reinforcement Learning/IRL) untuk menjaga tujuan tetap selaras.
 - **Masalah Kualitas Data:** *Mitigasi:* Terapkan tata kelola data yang ketat, protokol validasi, dan audit reguler.
 - **Bias Kumpulan Data:** *Mitigasi:* Gunakan metode pengumpulan data yang representatif, teknik penghilangan bias, dan pemodelan yang memperhatikan keadilan.
 - **Risiko Privasi:** *Mitigasi:* Manfaatkan enkripsi, anonimisasi data, kontrol akses yang ketat, dan pastikan kepatuhan terhadap peraturan seperti GDPR.
5. **Perbandingan Pendekatan Regulasi**
- **Undang-Undang AI Uni Eropa:** Regulasi yang mengikat secara hukum yang melindungi hak-hak fundamental melalui pendekatan berbasis risiko, memberlakukan kewajiban ketat terkait penilaian risiko, tata kelola data, dan transparansi pada sistem AI berisiko tinggi.
 - **Pedoman AI Tepercaya Uni Eropa:** Serangkaian prinsip etika yang menekankan otonomi manusia, ketahanan, privasi, dan transparansi, berfungsi sebagai kerangka kerja yang tidak mengikat untuk pengembangan etika.
 - **Undang-Undang Inisiatif AI Nasional AS:** Berfokus pada percepatan riset, pengembangan standar, dan mendorong kerja sama internasional untuk mempromosikan inovasi bersamaan dengan etika.
 - **Undang-Undang Akuntabilitas Algoritma:** Usulan legislatif yang mewajibkan penilaian dampak untuk sistem pengambilan keputusan otomatis, mewajibkan transparansi terkait data dan logika, dan mewajibkan mitigasi bias dan risiko privasi.
6. D. Akuntabilitas.

7. B. Kerangka Kerja Manajemen Risiko AI NIST.
8. C. Keselarasan & keadilan; Mengumpulkan data yang beragam dan menerapkan pemodelan yang mempertimbangkan bias/keadilan.

BAB 16: JAWABAN

1. Pengujian AI berbeda secara fundamental dari pengujian perangkat lunak tradisional karena harus memperhitungkan ketergantungan data, perilaku probabilistik (non-deterministik), dan evolusi model yang berkelanjutan. Sementara pengujian tradisional berfokus pada logika deterministik, strategi pengujian AI harus mencakup area fokus utama berikut:
 - **Validasi Data:** Memastikan kualitas, keterwakilan, dan bebas dari bias baik dalam dataset pelatihan maupun pengujian.
 - **Evaluasi Kinerja Model:** Menilai akurasi dan keandalan model di berbagai skenario dan kasus ekstrem.
 - **Pengujian Bias dan Keadilan:** Mendeteksi dan mengurangi hasil diskriminatif untuk memastikan hasil yang adil di berbagai kelompok pengguna.
 - **Pengujian Ketahanan dan Adversarial:** Memverifikasi ketahanan sistem terhadap input berbahaya, gangguan, atau kasus ekstrem.
 - **Pemantauan Berkelanjutan:** Melacak kinerja pasca-penyebaran untuk mengidentifikasi pergeseran konsep atau degradasi dari waktu ke waktu. Secara kolektif, area-area ini memastikan keandalan, keadilan, dan ketahanan operasional di seluruh siklus hidup AI.
2. **Pengujian Pakar Domain (DET) vs. Pengujian Penerimaan Pengguna (UAT)** Meskipun kedua strategi tersebut memvalidasi sistem dari perspektif manusia, keduanya memiliki tujuan yang berbeda:
 - **Pengujian Pakar Domain (DET):** Memanfaatkan pakar bidang untuk memvalidasi kasus penggunaan spesifik, kondisi ekstrem yang kompleks, dan kepatuhan terhadap batasan domain. Idealnya, ini menegaskan bahwa logika AI selaras dengan standar profesional.
 - *Contoh:* Seorang ahli radiologi meninjau alat triase AI untuk memastikan alat tersebut secara akurat menandai kelainan klinis yang halus namun kritis.
 - **Pengujian Penerimaan Pengguna (UAT):** Berfokus pada tujuan bisnis dan pengalaman pengguna akhir. Ini memvalidasi bahwa sistem memenuhi persyaratan operasional, terintegrasi dengan lancar ke dalam alur kerja, dan memenuhi standar kegunaan.
 - *Contoh:* Staf administrasi rumah sakit memastikan bahwa alat triase yang sama terintegrasi dengan lancar dengan perangkat lunak penjadwalan mereka dan bahwa waktu respons memenuhi kebutuhan operasional. Bersama-sama, DET memastikan kebenaran domain, sementara UAT memastikan keselarasan bisnis dan kegunaan praktis.

3. Pengujian non-fungsional memastikan sistem beroperasi dengan andal dalam berbagai kondisi. Aspek-aspek kunci meliputi:
 - **Kinerja:** Berfokus pada metrik standar seperti latensi, throughput, skalabilitas, dan pemanfaatan sumber daya untuk memastikan sistem memenuhi persyaratan operasional dasar.
 - **Pengujian Beban:** Mengevaluasi perilaku sistem di bawah lalu lintas yang padat namun diharapkan, mengukur konkurensi, throughput, dan tingkat kesalahan untuk memastikan stabilitas selama penggunaan puncak.
 - **Pengujian Stres:** Mendorong sistem melampaui kapasitas normalnya untuk mengidentifikasi titik kritis, hambatan, dan kemampuan untuk pulih dari kegagalan.
 - **Keamanan:** Menangani privasi data, perlindungan model (terhadap pencurian atau inversi), kontrol akses, deteksi ancaman, dan kepatuhan terhadap peraturan seperti GDPR atau HIPAA.
4. Pengujian fungsional dalam AI memprioritaskan validasi komponen individual, termasuk pra-pemrosesan data, rekayasa fitur, pipeline pelatihan, dan logika inferensi. Setiap komponen harus diuji unit secara terpisah untuk memastikan kebenarannya.

Teknik interpretasi mendukung prioritas ini dengan membuat penalaran model menjadi transparan. Metode seperti analisis kepentingan fitur, Plot Ketergantungan Parsial (PDP), Ekspektasi Bersyarat Individual (ICE), LIME, dan SHAP memungkinkan pengembang untuk:

 - Membedakan antara perilaku model lokal (tingkat instance) dan global.
 - Menganalisis sensitivitas terhadap fitur tertentu.
 - Melakukan debugging kasus kegagalan dengan lebih efektif. Transparansi ini membangun kepercayaan, memfasilitasi kepatuhan di domain yang diatur, dan mempercepat lokalisasi cacat.
5. **Metrik Keadilan: Definisi dan Penggunaan**
 - **Kesetaraan Demografis:** Membutuhkan tingkat prediksi positif yang sama di berbagai kelompok. Ini lebih disukai ketika tingkat hasil harus tetap sama terlepas dari keanggotaan kelompok (misalnya, jangkauan pemasaran).
 - **Kesempatan yang Sama:** Membutuhkan Tingkat Positif Sejati (TPR) yang sama di seluruh kelompok. Ini sangat penting ketika tujuannya adalah untuk memastikan bahwa individu yang memenuhi syarat di semua kelompok memiliki kemungkinan yang sama untuk dipilih (misalnya, persetujuan pinjaman untuk pelamar yang layak kredit).
 - **Peluang yang Sama:** Membutuhkan TPR dan Tingkat Positif Palsu (FPR) yang sama di seluruh kelompok. Ini digunakan ketika kedua jenis kesalahan (alarm palsu dan deteksi yang terlewatkan) membawa konsekuensi yang signifikan.
 - **Rasio Dampak yang Berbeda:** Mengukur rasio tingkat seleksi positif antara kelompok yang dilindungi dan kelompok referensi. Metrik ini sering digunakan untuk mengukur keadilan seleksi dalam konteks hukum atau SDM. Pilihan metrik

sangat bergantung pada pertimbangan spesifik domain, seperti keseimbangan antara kesetaraan perekrutan dan manajemen risiko.

6. C. Throughput
7. B. Dorong melampaui kapasitas normal untuk menemukan titik kritis dan memverifikasi pemulihan/stabilitas.
8. C. LIME (Penjelasan Model-agnostik yang Dapat Diinterpretasikan Secara Lokal)

DAFTAR PUSTAKA

- Ahir, S., Telavane, D., & Thomas, R. (2020, September). The impact of artificial intelligence, blockchain, big data and evolving technologies in coronavirus disease-2019 (COVID-19) curtailment. In *2020 international conference on smart electronics and communication (ICOSEC)* (pp. 113-120). IEEE.
- Ahmed, I., Jeon, G., & Piccialli, F. (2022). From artificial intelligence to explainable artificial intelligence in industry 4.0: a survey on what, how, and where. *IEEE transactions on industrial informatics*, 18(8), 5031-5042.
- Auger, S. D., Jacobs, B. M., Dobson, R., Marshall, C. R., & Noyce, A. J. (2021). Big data, machine learning and artificial intelligence: a neurologist's guide. *Practical Neurology*, 21(1), 4-11.
- Bhandari, K., Kumar, K., & Sangal, A. L. (2023, May). Artificial intelligence in software engineering: Perspectives and challenges. In *2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC)* (pp. 133-137). IEEE.
- Bickley, S. J., Macintyre, A., & Torgler, B. (2025). Artificial intelligence and big data in sustainable entrepreneurship. *Journal of Economic Surveys*, 39(1), 103-145.
- Busuioc, M. (2021). Accountable artificial intelligence: Holding algorithms to account. *Public administration review*, 81(5), 825-836.
- Chinta, P. C. R. (2020). A Deep Learning Architectures for Enhancing Cyber Security Protocols in Big Data Integrated ERP Systems. *Journal of Artificial Intelligence and Big Data*, 1(1), 10-31586.
- Darvazeh, S. S., Vanani, I. R., & Musolu, F. M. (2020). Big data analytics and its applications in supply chain management. *New trends in the use of artificial intelligence for the industry*, 4(10), 175.
- Dorneanu, B., Zhang, S., Ruan, H., Heshmat, M., Chen, R., Vassiliadis, V. S., & Arellano-Garcia, H. (2022). Big data and machine learning: A roadmap towards smart plants. *Frontiers of Engineering Management*, 9(4), 623-639.
- Dounia, G., Chaimae, K., Yassine, H., & Houda, B. (2025). Artificial Intelligence And Big Data In Management Control Of Moroccan Companies: Case Of The Rabat-Sale-Kenitra Region. *Proceedings on Engineering*, 7(2), 925-938.

- Gourlet, P., Ricci, D., & Crépel, M. (2024). Reclaiming artificial intelligence accounts: A plea for a participatory turn in artificial intelligence inquiries. *Big Data & Society*, 11(2), 20539517241248093.
- Goyal, D., Goyal, R., Rekha, G., Malik, S., & Tyagi, A. K. (2020, February). Emerging trends and challenges in data science and big data analytics. In *2020 International conference on emerging trends in information technology and engineering (ic-ETITE)* (pp. 1-8). IEEE.
- Holton, R., & Boyd, R. (2021). 'Where are the people? What are they doing? Why are they doing it?' (Mindell) Situating artificial intelligence within a socio-technical framework. *Journal of Sociology*, 57(2), 179-195.
- Hopgood, A. A. (2021). *Intelligent systems for engineers and scientists: a practical guide to artificial intelligence*. CRC press.
- Johnson, M., Jain, R., Brennan-Tonetta, P., Swartz, E., Silver, D., Paolini, J., ... & Hill, C. (2021). Impact of big data and artificial intelligence on industry: developing a workforce roadmap for a data driven economy. *Global Journal of Flexible Systems Management*, 22(3), 197-217.
- Mahmoud, A. B., Kumar, V., & Spyropoulou, S. (2025). Identifying the public's beliefs about generative artificial intelligence: A big data approach. *IEEE Transactions on Engineering Management*, 72, 827-841.
- Mathew, D., Brintha, N. C., & Jappes, J. W. (2023). Artificial intelligence powered automation for industry 4.0. In *New horizons for Industry 4.0 in modern business* (pp. 1-28). Cham: Springer International Publishing.
- Misra, N. N., Dixit, Y., Al-Mallahi, A., Bhullar, M. S., Upadhyay, R., & Martynenko, A. (2020). IoT, big data, and artificial intelligence in agriculture and food industry. *IEEE Internet of things Journal*, 9(9), 6305-6324.
- Mohapatra, P. S. (2025). Artificial intelligence and machine learning for test engineers: concepts in software quality assurance. *Intelligent Assurance: Artificial Intelligence-Powered Software Testing in the Modern Development Lifecycle*, 4, 17.
- Nadikattu, R. R. (2020). Research on data science, data analytics and big data. *International Journal of Engineering, Science and*, 9(5), 99-105.
- Nesterov, V. (2023). Integration of artificial intelligence technologies in data engineering: Challenges and prospects in the modern information environment. *Вісник Черкаського державного технологічного університету. Технічні науки*, 28(4), 82-90.

- Raju, B., Jumah, F., Ashraf, O., Narayan, V., Gupta, G., Sun, H., ... & Nanda, A. (2020). Big data, machine learning, and artificial intelligence: a field guide for neurosurgeons. *Journal of neurosurgery*, 135(2), 373-383.
- Reis, T., Bornschlegl, M. X., & Hemmje, M. L. (2021). Toward a reference model for artificial intelligence supporting big data analysis. In *Advances in Data Science and Information Engineering: Proceedings from ICDATA 2020 and IKE 2020* (pp. 561-572). Cham: Springer International Publishing.
- Srinivasan, S. M., & Sangwan, R. S. (2026). *What Every Engineer Should Know about Artificial Intelligence and Big Data*. CRC Press.
- Sterne, J., & Razlogova, E. (2021). Tuning sound for infrastructures: artificial intelligence, automation, and the cultural politics of audio mastering. *Cultural Studies*, 35(4-5), 750-770.
- Thayyib, P. V., Mamilla, R., Khan, M., Fatima, H., Asim, M., Anwar, I., ... & Khan, M. A. (2023). State-of-the-art of artificial intelligence and big data analytics reviews in five different domains: a bibliometric summary. *Sustainability*, 15(5), 4026.
- Thontirawong, P., & Chinchanchokchai, S. (2021). Teaching artificial intelligence and machine learning in marketing. *Marketing Education Review*, 31(2), 58-63.
- Torgler, B. (2023). Big data, artificial intelligence, and quantum computing in sports. In *21st century sports: How technologies will change sports in the digital age* (pp. 169-189). Cham: Springer International Publishing.
- Widder, D. G., & Nafus, D. (2023). Dislocated accountabilities in the “AI supply chain”: Modularity and developers’ notions of responsibility. *Big data & society*, 10(1), 20539517231177620.
- Yao, Y. (2022). A review of the comprehensive application of big data, artificial intelligence, and internet of things technologies in smart cities. *Journal of computational methods in engineering applications*, 1-10.
- Zabala-Vargas, S., Jaimes-Quintanilla, M., & Jimenez-Barrera, M. H. (2023). Big data, data science, and artificial intelligence for project management in the architecture, engineering, and construction industry: a systematic review. *Buildings*, 13(12), 2944.
- Zhai, X., Chu, X., Chai, C. S., Jong, M. S. Y., Istenic, A., Spector, M., ... & Li, Y. (2021). A Review of Artificial Intelligence (AI) in Education from 2010 to 2020. *Complexity*, 2021(1), 8812542.

Sistem Kecerdasan Buatan (AI) dan Big Data

Dr. Joseph Teguh Santoso, S.Kom, M.Kom.

BIODATA PENULIS



Dr. Joseph Teguh Santoso, M.Kom memiliki Jabatan Akademik Lektor Kepala dan praktisi industri yang berpengalaman. Saat ini menjabat sebagai Rektor Universitas Sains dan Teknologi Komputer (Universitas STEKOM), salah satu universitas terkemuka di Jawa Tengah, Indonesia. Dengan pengalaman lebih dari 13 tahun di dunia bisnis dan praktisi industri di China, beliau membawa perspektif global dan inovasi yang signifikan ke dalam dunia akademis. Sebagai seorang entrepreneur, penulis adalah pencipta TopLoker.com, sebuah platform inovatif yang merevolusi cara mencari dan menawarkan pekerjaan. TopLoker.com adalah portal lowongan bursa kerja terbesar di Indonesia, khusus untuk pendidikan SMA/SMK sederajat.

TopLoker.com telah mendapatkan penghargaan sebagai juara 1 Startup4Industry 2022 oleh Kementerian Perindustrian Republik Indonesia. Kontribusi Dr. Joseph dalam menyediakan akses pekerjaan yang luas bagi lulusan SMA/SMK telah membantu banyak individu menemukan peluang kerja yang sesuai dengan keahlian mereka. Selain itu, Dr. Joseph Teguh Santoso, M.Kom adalah pendiri dari dua organisasi yaitu (1) organisasi guru/pendidik PTIC (Perkumpulan Teacherpreneur Indonesia Cerdas) yang bertujuan untuk meningkatkan kualitas pendidikan dan kesejahteraan guru/pendidik dengan wawasan entrepreneurship, serta (2) organisasi industri PERKIVI (Perkumpulan Komunitas Industri dan Vokasi Indonesia) yang berfokus pada pengembangan link and match antara industri dan dunia pendidikan. Sebagai Rektor, Dr. Joseph Teguh Santoso, M.Kom memiliki kepemimpinan yang berorientasi pada hasil, dan berkomitmen untuk mendorong kemajuan Universitas Sains dan Teknologi Komputer (Universitas STEKOM). Saat ini Universitas STEKOM telah mengalami transformasi positif dalam peningkatan kualitas pendidikan, perluasan fasilitas, serta penguatan kemitraan Perguruan Tinggi Nasional dan Internasional. Beliau memprioritaskan pengembangan sumber daya manusia dan penelitian, serta memastikan bahwa universitas berada di garis depan dalam inovasi dan teknologi untuk mencapai tujuan akhir, yaitu lulusan yang mampu bekerja dan sukses setelah lulus. Dr. Joseph Teguh Santoso, M.Kom sering diundang sebagai pembicara di berbagai konferensi nasional maupun internasional dan telah menerima berbagai penghargaan atas dedikasinya dalam bidang pendidikan, industri, dan kewirausahaan.



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-634-7695-46-8 (PDF)



9

786347

695468