BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian terdahulu penting dalam memahami konteks dan landasan penelitian ini. Berikut disajikan Tabel 2.1 yang merangkum penelitian-penelitian terdahulu terkait optimalisasi pencarian informasi menggunakan model *deep learning* dan *artificial intelligence* berbasis *web*. Tabel 2.1 bertujuan untuk mengidentifikasi kontribusi, metode, serta hasil dari penelitian sebelumnya yang relevan.

Tabel 2. 1 Penelitian Terdahulu

Penulis	Judul	Jurnal	Metode	Hasil
Irene Li, Jessica Pan, Jeremy Goldwasser, Neha Verma, Wai Pan Wong, Muhammed Yavuz Nuzumlalı, Benjamin Rosand, Yixin Li, Matthew Zhang, David Chang, R. Andrew Taylor, Harlan M. Krumholz [9].	*Neural Natural Language Processing for Unstructured Data in Electronic Health Records*	Computer Science Review Volume 46, November 2022, 100511	LSTM dan BERT	Penerapan NLP dalam pencarian catatan kesehatan elektronik (EHR). Metode *Natural Language Processing* (NLP) yaitu LSTM (82%) dan BERT (84%)
Arya Gaikwad, Palash Rambhia, Sarthak Pawar [10].	*An Extensive Analysis Between Different Language Models: GPT-3, BERT and MACAW*	Research Square	GPT-3, BERT, MACAW	Perbandingan antara GPT-3, MACAW, dan BERT menunjukkan bahwa GPT-3 unggul dalam menghasilkan teks dan memberikan jawaban mendalam dengan jumlah parameter terbesar, sementara MACAW lebih efisien dengan jawaban ringkas.
Rachel Chambers, Naomi Tack, Eliot Pearson, Lara J. Martin, Francis Ferraro [11].	*BERALL:Towards Generating Retrieval-augmented State-based Interactive Fiction Games*	University of Maryland	BERT-RAG	Penerapan BERALL dalam *text generation game*. Metode BERT-Tiny: 5.73%, BERT-Medium: 13.3%, BERT-Base: 14.7%
Tengyang Chen, Hongyu Li, Miho Kasamatsu, Tak	*Developing a How-to Tip Machine Comprehension*	Association for Computational	BERT	Penerapan BERT dalam tanya jawab *tip machine comprehension* dengan hasil yang baik.

Penulis	Judul	Jurnal	Metode	Hasil
ehito Utsuro, Yasuhide Kawada [12]	*Dataset and its Evaluation in Machine Comprehension by BERT*	Linguistics, Volume Proceedings of the Third Workshop on Fact Extraction and VERification (FEVER), Pages 26–35		
Alex Brandsen, Suzan Verberne, Karsten Lambers, Milco Wansleben [13].	*Can BERT Dig It? Named Entity Recognition for Information Retrieval in the Archaeology Domain*	Journal on Computing and Cultural Heritage (JOCCH), Volume 15, Issue 3 Article No.: 51, Pages 1 - 18	BERT	Penerapan model BERT pada *search engine* untuk arkeologi. Metode BERT 69%.
Juanhui Li, Yao Ma, Wei Zeng, Jiliang Tang, Dawei Yin, Suqi Cheng, Shuaiqiang Wang[14]	Graph Enhanced BERT for QueryUnderstanding	SIGIR '23: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval Pages 3315 - 3319	BERT, GE-BERT	Perbandingan model BERT dan GE-BERT dalam *search engine* informasi perilaku. Metode NLP yaitu BERT 78% dan GE-BERT 79%
Awane Widad, Ben Lahmar El Habib, El Falaki Ayoub [15].	Bert for Question Answering applied on Covid-19	Procedia Computer Science Volume 198, 2022, Pages 379-384	BERT	Penerapan model BERT untuk menjawab pertanyaan Covid 19. Hasil akurasi yang didapatkan 71.25%
Reza Khanmohammadi, Mitra Sadat Mirshafiee, Mehdi Allahyari [16]	COPER: a Query-adaptable Semantics-based Search Engine for Persian COVID-19 Articles	2021 7th International Conference on Web Research (ICWR)	BERT	Membandingkan beberapa model *text processing* untuk *search engine*, salah satunya BERT dengan akurasi 43%.
Yosi Mass, Boaz Carmeli, Haggai Roitman, David Konopnicki [17]	Unsupervised FAQ Retrieval with Question Generation and BERT	ACL, Volume Proceedings of the 58th Annual Meeting of	BERT	Membandingkan beberapa model *text processing* untuk *search engine*, salah satunya

Penulis	Judul	Jurnal	Metode	Hasil
		the Association for Computational Linguistics, Pages 807–812		BERT dengan akurasi 54%.
Vaibhav Kumar, Jamie Callan [18]	*Making Information Seeking Easier: An Improved Pipeline for Conversational Search*	Association for Computational Linguistic, Volume Findings of the Association for Computational Linguistics: EMNLP 2020, Pages 3971–3980	BERT, GPT 2	Penerapan BERT meningkatkan akurasi sebesar 14,8% dan relevansi hasil pencarian dalam konteks percakapan.
Haoyi Xiong, Senior Member, Jiang Bian, Member, Yuchen Li, Xuhong Li, Mengnan Du, Member, Shuaiqiang Wang, Dawei Yin, Senior Member, and Sumi Helal, Fellow	*When Search Engine Services meet Large Language Models: Visions and Challenges*	IEEE	LLM, GPT-4, RAG	Integrasi LLMs dan mesin pencari menjanjikan pencarian yang lebih cerdas dan adaptif, meski menghadapi tantangan teknis dan etika. Sinergi ini membuka era baru inovasi dan penelitian.
Grzegorz Chodak, Klaudia Błażyczek	*Large Language Models for Search Engine Optimization in E-commerce*	dvanced Computing. IACC 2023. Communications in Computer and Information Science, vol 2053. Springer	LLM, GPT-4	Konten yang dihasilkan oleh LLM mempercepat pembuatan, menghemat biaya, dan optimasi SEO, namun bisa kurang orisinal dan terdengar buatan, yang dapat mengurangi kepercayaan audiens. Penggunaan LLM perlu hati-hati, dan penelitian lebih lanjut tentang dampaknya pada algoritma mesin pencari dibutuhkan.

Penulis	Judul	Jurnal	Metode	Hasil
Yopi Nugraha [19]	*Information System Development With Comparison of Waterfall and Prototyping Models*	RISTEC : Research in Information Systems and Technology Vol. 1 No. 1 Tahun. 2020	*Prototyping* dan *waterfall*	*Waterfall* cocok untuk kebutuhan tetap, sementara *prototype* untuk pengembangan iteratif.
Vira Adi Kurniyanti, Deni Murdiani [20]	Perbandingan Model Waterfall Dengan Prototype Pada Pengembangan System Informasi Berbasis Website	Vol. 2 No. 08 (2022): Jurnal Fusion: Jurnal Nasional Indonesia	*Prototyping* dan *waterfall*	Waterfall cocok untuk kebutuhan tetap, sementara prototype untuk kebutuhan spesifik dengan masukan pengguna, meski memakan waktu lebih lama.

Penelitian terdahulu pada Tabel 2.1 memberikan beberapa temuan yang relevan dengan topik penelitian ini. Penerapan algoritma Natural Language Processing (NLP) dalam sistem pencarian informasi telah menunjukkan potensi besar dalam meningkatkan relevansi dan akurasi pencarian, meskipun terdapat beberapa permasalahan yang perlu diatasi. Penelitian terdahulu, seperti yang dilakukan oleh Irene Li dkk., menunjukkan bahwa algoritma seperti LSTM dan BERT berhasil meningkatkan akurasi pencarian pada data tidak terstruktur, sementara Arya Gaikwad dkk. mengungkapkan bahwa GPT-3 unggul dalam menghasilkan teks mendalam, meskipun lebih kompleks dibandingkan dengan model seperti BERT dan MACAW. Penelitian lain, seperti yang dilakukan oleh Rachel Chambers dkk. dan Alex Brandsen dkk., juga mengaplikasikan BERT dalam domain tertentu, seperti permainan interaktif dan pencarian arkeologi, dengan hasil yang bervariasi. Beberapa studi juga menunjukkan penerapan BERT dalam pencarian terkait Covid-19 dan bahasa minoritas, yang mengungkapkan tantangan dalam pemrosesan bahasa dan keterbatasan pada domain tertentu. Namun, meskipun BERT menunjukkan kemampuan luar biasa dalam memahami konteks, model-model sebelumnya masih menghadapi kendala dalam menghadapi data besar, multibahasa, dan efisiensi komputasi. Untuk itu, penelitian ini menggunakan GPT-4 karena kemampuannya yang lebih baik dalam memahami konteks, mendukung berbagai bahasa, dan mengatasi keterbatasan yang ada pada model sebelumnya. Integrasi BERT-RAG dengan GPT-4 diharapkan dapat meningkatkan kualitas pencarian informasi, memberikan hasil pencarian yang lebih relevan, akurat, dan kontekstual, serta mengatasi permasalahan yang dihadapi oleh penelitian terdahulu.

2.2 Teori tentang Topik Skripsi

2.2.1 *Database*

Database atau basis data atau gudang yang merupakan tempat menyimpan catatan data dan informasi yang dibutuhkan seperti data manusia, hewan, barang, konsep, peristiwa, dan lainnya yang dapat disimpan dalam format huruf, angka, symbol, gambar, bunyi, ataupun kombinasi dari semuanya [21]. *Database* juga merupakan sebuah sistem yang dapat digunakan untuk mengelola, menyimpan dan mengambil data dan informasi dengan mudah [22]. *Database* adalah sistem penyimpanan data yang terorganisir dengan tujuan utama untuk mengurangi duplikasi atau redundansi data [23]. Dari beberapa pengertian yang disebutkan, *database* merupakan sebuah bentuk dari suatu data dan infromasi yang dikumpulkan dan disimpan, dimana data atau informasi tersebut memiliki hubungan satu sama lain.

Database merupakan bagian yang menyimpan data yang diakses oleh pengguna yang dapat berupa manusia maupun komputer. Untuk menghubungkan *database* dan pengguna, memerlukan API. API memungkinkan pengguna untuk mengakses *database* dengan cepat tanpa mengetahui cara kerja *database* secara internal. Selain API, *database* juga akan memerlukan aplikasi untuk dapat menggunakan *database*. Aplikasi ini dapat berupa aplikasi *mobile*, aplikasi *desktop*, maupun aplikasi seluler.

Data didalam *database* akan disimpan dalam sebuah sistem komputer yang dapat memenuhi kebutuhan informasi oleh sebuah organisasi atau perusahaan. Data yang disimpan akan tersusun secara sistematis dan terstruktur yang dapat menggunakan metode yang dibutuhkan [24]. *Database* memiliki banyak kegunaan seperti, dapat membantu *user* identifikasi data, meminimalisir data yang kembar, mempermudah pengelompokan dan pengambilan data [22]. *Database* juga memiliki banyak jenis yang dapat disesuaikan dengan kebutuhan dari sebuah organisasi maupun perusahaan. Oleh karena itu, *database* merupakan gudang data yang penting bagi sebuah organisasi atau perusahaan karena selain dapat menyimpan data juga dapat membantu untuk mengolah data menjadi lebih cepat.

2.2.2 Kecerdasan Buatan (*Artificial Intelligence*)

Kecerdasan buatan atau *artificial intelligence* (AI) merupakan suatu ilmu atau teknik dalam menciptakan mesin yang bersifat cerdaa, terutama dalam menciptakan program atau aplikasi computer cerdas. AI bertujuan untuk membuat komputer melaksanakan suatu perintah yang dapat dilakukan oleh manusia. AI memiliki sejarah

panjang dan berkembang. Pada tahun 1950-an John McCarthy memperkenalkan istilah "*Artificial Intelligence*" dan mengorganisir konferensi pertama dalam bidang ini. Sejak itu, AI telah berkembang pesat dengan penemuan-penemuan baru dalam bidang komputer, matematika, dan psikologi [25].

AI dapat diartikan sebagai kecerdasan buatan, yang mana pada prosesnya berarti membuat atau mempersiapkan mesin seperti computer agar memiliki *intelligence* atau kecerdasan berdasarkan perilaku manusia. AI bertujuan untuk membuat computer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia [26]. Terdapat lingkup utama dari AI diantaranya [27] :

1. Sistem pakar

   Komputer biasa digunakan sebagai sarana untuk menyimpan pengetahuan para pakar. Sistem pakar dapat melakukan perilaku cerdas, belajar, mendemonstrasikan, menjelaskan, dan menyarankan *user*.
2. Pengolahan Bahasa alami

   Dengan Bahasa alami, *user* dapat berkomunikasi dengan computer menggunakan bahasa sehari-hari.
3. Pengenalan ucapan

   Melalui pengenalan ucapan, computer dapat mengenali dan memahami ucapan manusia untuk melakukan tindakan tertentu.

Sehingga, AI merupakan suatu metode untuk membuat computer dapat memiliki kecerdasan dan dapat berpikir layaknya manusia dalam mencari jalan keluar suatu permasalahan. AI memiliki banyak bidang terapan sehingga dapat membantu meningkatkan efisiensi dan keandalan dalam berbagai bidang aplikasi.

2.2.3 Basis data terdistribusi

Basis data terdistribusi atau *distributed database* merupakan suatu sistem yang menyimpan data di beberapa lokasi yang berbeda dan masing masing diatur oleh suatu sistem manajemen basis data (DBMS) yang mandiri [28]. *Distributed database* memungkinkan *user* untuk mengakses dan memanipulasi data dari berbagai lokasi secara transparan dan efisien. Terdapat beberapa keuntungan menggunakan basis data terdistribusi:

1. Manajemen data terdistribusi dengan tingkat transparansi yang berbeda

   User tidak perlu mengetahui dimana data berada, sehingga memudahkan penggunaan dan integrasi dengan aplikasi.

2. Keandalan dan ketersediaan

   Sistem terus beroperasi selama paling tidak satu *site* tetap beroperasi, sehingga meningkatkan keandalan dan ketersediaan data.

3. Peningkatan peforma

   Dengan membagi beban pengolahan data ke beberapa lokasi, peforma sistem dapat ditingkatkan.

4. Ekspansi yang lebih mudah

   Sistem dapat dengan mudah menambah atau mengurangi kapasitas penyimpanan data tanpa gangguan operasional.

2.2.4 Natural Language Processing (NLP)

Pemrosesan Bahasa Alami *Natural Language Processing* (NLP) merupakan cabang dari kecerdasan buatan (AI) yang berfokus pada interaksi antara komputer dan bahasa manusia [29]. NLP memungkinkan komputer untuk memahami, menganalisis, dan memproduksi bahasa manusia dengan cara yang lebih alami. Dengan mengombinasikan teknik komputasi linguistik, model statistik, dan algoritma pembelajaran mesin, NLP dapat memproses data suara dan teks untuk memahami makna yang terkandung di dalamnya [30] .

Proses kerja NLP dimulai dengan tokenisasi, yaitu memecah teks menjadi unit-unit kecil seperti kata atau frasa. Selanjutnya, dilakukan analisis gramatikal untuk memahami struktur kalimat dan hubungan antar kata [31]. Teknik lain yang digunakan termasuk analisis sentimen, yang mengidentifikasi apakah teks memiliki nada positif, negatif, atau netral, serta penerjemahan mesin yang memungkinkan terjemahan otomatis dari satu bahasa ke bahasa lain. Contoh aplikasi NLP yang umum ditemukan adalah asisten virtual seperti Siri dan, serta alat penerjemahan seperti Google translate [32] .

NLP sangat penting karena membantu komputer berkomunikasi dengan manusia dalam bahasa yang mereka gunakan sehari-hari. Ini tidak hanya meningkatkan efisiensi dalam pemrosesan informasi tetapi juga memungkinkan interaksi yang lebih intuitif antara manusia dan mesin. Namun, NLP juga menghadapi tantangan seperti ambiguitas bahasa dan variasi dialek yang dapat menyulitkan pemahaman [30].

2.2.5 *Machine Learning*

Machine learning adalah cabang dari kecerdasan buatan (AI) yang menggunakan algoritma komputer dan teknik statistik untuk memungkinkan sistem belajar dari data dan membuat prediksi atau keputusan tanpa diprogram secara eksplisit. Pada dasarnya, *machine learning* memproses data historis untuk menemukan pola atau hubungan di dalamnya, lalu menggunakan pola tersebut untuk memprediksi hasil di masa depan. Ada tiga jenis utama pembelajaran dalam *machine learning*, yaitu *supervised learning*, *unsupervised learning*, dan *reinforcement learning*, yang masing-masing memiliki metode dan aplikasi yang berbeda [33].

Supervised learning adalah tipe pembelajaran di mana model dilatih menggunakan dataset yang sudah diberi label. Setiap input memiliki output yang diketahui, sehingga algoritma dilatih untuk memetakan input ke output yang benar. Proses ini melibatkan pembelajaran dari data berlabel untuk memprediksi hasil pada data baru yang belum pernah dilihat sebelumnya. Contoh dari *supervised learning* termasuk klasifikasi email sebagai spam atau tidak, serta prediksi harga rumah berdasarkan fitur-fitur tertentu seperti luas rumah dan lokasi [34].

Unsupervised learning, sebaliknya, bekerja dengan dataset yang tidak diberi label. Dalam tipe pembelajaran ini, model mencoba menemukan struktur atau pola tersembunyi dalam data. Algoritma *unsupervised learning* berfungsi untuk mengelompokkan atau mengelompokan data tanpa mengetahui output sebelumnya. Salah satu aplikasi populer dari *unsupervised learning* adalah *clustering*, seperti mengelompokkan pelanggan berdasarkan perilaku belanja mereka, atau segmentasi gambar dalam pengenalan pola[35].

Reinforcement Learning adalah metode pembelajaran yang berbeda, di mana agen (sistem atau model) belajar melalui interaksi berulang dengan lingkungan. Agen tersebut tidak membutuhkan pengetahuan awal, melainkan berupaya meningkatkan performanya dengan mencoba berbagai tindakan, mengamati hasil dari tindakan tersebut, dan belajar dari umpan balik yang diterima. Proses ini berlangsung melalui siklus coba-coba, di mana agen menerima penghargaan atau hukuman berdasarkan hasil tindakannya. Contoh aplikasi *reinforcement learning* meliputi pengendalian robot, permainan video, dan sistem rekomendasi [36].

Ketiga jenis pembelajaran ini memberikan solusi yang berbeda tergantung pada masalah dan jenis data yang tersedia. *Machine learning* telah membawa revolusi di berbagai industri, seperti kesehatan, keuangan, pemasaran, hingga

transportasi, karena kemampuannya untuk menganalisis data besar, menemukan pola tersembunyi, dan menghasilkan keputusan atau prediksi yang akurat.

2.2.6 *Deep Learning*

Deep learning merupakan cabang dari *machine learning* yang menggunakan jaringan atau *neural networks* dengan banyak lapisan. Setiap lapisan dalam deep learning terdiri dari sejumlah node yang berfungsi sebagai tempat untuk melakukan perhitungan [37]. Secara umum, lapisan-lapisan dalam *deep learning* dibagi menjadi tiga bagian: *input layer, hidden layer*, dan *output layer*. *Input layer* adalah tempat data dimasukkan ke dalam jaringan saraf. *Hidden layer* berada di antara *input* dan *output*, berfungsi untuk mengolah dan mentransformasikan data. Sementara itu, *output layer* adalah lapisan yang menghasilkan hasil akhir dari transformasi yang dilakukan oleh *hidden layer* [38].

2.2.7 *Software Development Life Cycle*

Sofware Development Life Cycle (SDLC) merupakan metode yang digunakan untuk pengembangan sistem informasi, yang bertujuan untuk memberikan struktur dan proses yang jelas dalam setiap fase pengembagan perangkat lunak [39]. SDLC terdiri dari beberapa tahapan, yaitu, perencanaan, analisis, desain, implementasi, pengujian. Integrasi, dan pemeliharaan. Setiap tahap memiliki tujuan spesifik dan saling terkait untuk memasikan bahwa produk akhir memenuhi kebutuhan pengguna dan standar kualitas yang diharapkan [40]. Terdapat berbagai model SDLC yang umum digunakan, seperti *waterfall*, *prototyping*, dan *agile*. Model *waterfall* mengikuti pendekatan linear yang cocok untuk proyek dengan persyaratan yang stabil, sementara *prototyping* memungkinkan pengembangan prototipe untuk mendapatkan umpan balik dari pengguna. Selain itu, model agile menekankan fleksibilitas dan kolaborasi tim dengan membagi proyek menjadi iterasi kecil. Penerapan SDLC yang efektif tidak hanya meningkatkan kualitas perangkat lunak tetapi juga membantu dalam manajemen risiko dan meningkatkan kepuasan pelanggan, sehingga organisasi dapat mencapai tujuan startegis mereka dengan lebih baik [41].

2.2.8 *User Acceptance Testing*

User Acceptance Testing (UAT) adalah tahap akhir dalam proses pengujian perangkat lunak yang bertujuan untuk memastikan bahwa sistem yang dikembangkan memenuhi kebutuhan dan harapan pengguna akhir [41]. UAT dilakukan oleh klien

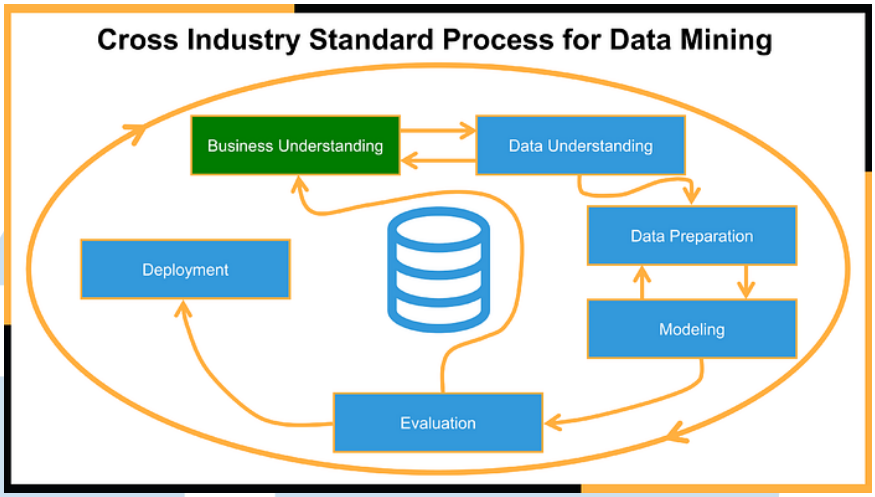
atau pengguna akhir. UAT dilakukan oleh klien atau pengguna akhir untuk memverifikasi bahwa solusi yang diimplementasikan dapat berfungsi dengan baik dalam konteks nyata dan sesuai dengan spesifikasi yang telah ditentukan sebelumnya [42]. Berbeda dengan pengujian sistem yang lebih fokus pada stabilitas dan kesesuaian teknis, UAT menilai apakah aplikasi tersebut siap untuk digunakan secara praktis oleh pengguna [43]. Proses UAT melibatkan beberapa langkah penting termasuk perencanaan, pelaksanaan, dan evaluasi. Pada tahap perencanaan, tim pengembang dan pengguna harus bekerja sama untuk menentukan kriteria keberhasilan dan skenario pengujian yang relevan. Selama pelaksanaan, pengguna akan melakukan serangkaian tes untuk mengevaluasi fungsi dan kegunaan sistem. Akhirnya, hasil dari pengujian ini akan dievaluasi untuk menentukan apakah sistem dapat diterima atau perlu perbaikan lebih lanjut.

2.3 *Framework* dan Algoritma

Framework yang digunakan dalam penelitian mencakup konsep, fungsi, dan keunggulan framework tersebut dalam mendukung pengembangan sistem. Penjelasan ini bertujuan untuk memberikan pemahaman mendalam tentang alasan pemilihan framework serta relevansinya dalam implementasi sistem yang dirancang.

2.3.1 *Cross-Industry Standard Process for Data Mining* (CRISP-DM)

Cross-Industry Standard Process for Data Mining (CRISP-DM) merupakan metodologi yang sering digunakan dalam proses *data mining*, yang terdiri dari enam tahapan yang saling terhubung dan terintegrasi dengan baik [44]. Tahapan tersebut meliputi pemahaman bisnis, yang menetapkan tujuan proyek dan kebutuhan bisnis; pemahaman data, yang mengumpulkan dan mengevaluasi data untuk memahami karakteristiknya; persiapan data, yang membersihkan dan mempersiapkan data untuk analisis lebih lanjut; pemodelan, di mana model analisis dibangun berdasarkan data yang telah disiapkan; Evaluasi, yang menilai model untuk memastikan bahwa ia memenuhi tujuan bisnis yang ditetapkan; dan penyebaran, yang mengimplementasikan model dalam lingkungan nyata untuk digunakan oleh pengguna akhir [44]. Metodologi ini dirancang untuk membantu organisasi dalam mengelola dan menganalisis data secara efektif, serta menghasilkan wawasan yang mendukung pengambilan keputusan yang lebih baik. CRISP-DM memberikan kerangka kerja terstruktur untuk melakukan proyek data mining secara efektif dan efisien. Metode ini bersifat iteratif dan dapat diterapkan pada berbagai industri dan jenis data. CRISP-DM membantu tim data mining untuk tetap fokus pada tujuan bisnis sambil mengeksplorasi dan menganalisis data secara mendalam.



Gambar 2. 1 Tahapan CRISP-DM [45]

Metode CRISP-DM ini telah banyak diterapkan dalam berbagai industri, termasuk dalam pengelolaan sistem informasi dan data yang kompleks, seperti pada proyek penelitian ini. Setiap langkah memiliki peran penting dalam memastikan bahwa analisis data dilakukan secara efektif dan menghasilkan wawasan yang dapat mendukung pengambilan keputusan yang lebih baik. Pada Gambar 2.1 diperlihatkan tahapan dalam metode CRISP-DM yang terdiri dari enam langkah utama yang membentuk suatu proses iteratif dalam data mining diantaranya [45]:

1. *Bussiness Understanding*

   Tahap pertama merupakan pemahaman bisnis, yang bertujuan untuk memahami tujuan yang ingin dicapai melalui data mining. Pada tahap ini, masalah bisnis didefinisikan dengan jelas, tujuan yang ingin dicapai diidentifikasi, dan informasi yang relevan dikumpulkan untuk mendukung analisis. Pemahaman yang baik terhadap konteks bisnis sangat penting agar hasil data mining dapat memberikan nilai tambah yang signifikan bagi organisasi.

2. *Data Understanding*

   Tahap kedua merupakan pemahaman data, yang bertujuan untuk memahami data yang tersedia untuk proyek. Langkah-langkah pada tahap ini meliputi pengumpulan data awal, deskripsi data untuk memahami karakteristiknya, eksplorasi data untuk mencari pola atau hubungan yang ada, serta verifikasi kualitas data untuk memastikan data yang digunakan dapat diandalkan.

Proses ini penting untuk memastikan bahwa data yang digunakan dalam analisis sudah siap dan sesuai dengan tujuan proyek.

3. *Data Preparation*

   Tahap ketiga merupakan persiapan data, yang bertujuan untuk menyiapkan data agar siap untuk dimodelkan. Langkah-langkah pada tahap ini mencakup pembersihan data untuk menghilangkan kesalahan atau ketidaksesuaian, integrasi data dari berbagai sumber, serta pemilihan data yang relevan untuk analisis. Selain itu, data juga diubah ke dalam format yang sesuai agar dapat digunakan dengan efektif dalam tahap pemodelan selanjutnya.

4. *Data Modelling*

   Tahap keempat merupakan pemodelan, yang bertujuan untuk membangun model data mining yang sesuai dengan tujuan bisnis. Langkah-langkah pada tahap ini meliputi pemilihan teknik data mining yang tepat berdasarkan karakteristik data dan tujuan analisis, serta pembangunan model menggunakan teknik yang dipilih. Setelah model dibangun, langkah selanjutnya adalah menguji model untuk mengevaluasi kinerjanya dan memastikan bahwa model tersebut dapat memberikan hasil yang akurat dan relevan.

5. *Evaluation*

   Tahap kelima merupakan evaluasi, yang bertujuan untuk mengevaluasi model yang telah dibangun. Langkah-langkah pada tahap ini mencakup evaluasi kinerja model untuk memastikan bahwa model bekerja sesuai harapan, serta membandingkan model dengan alternatif lain untuk memilih yang terbaik. Selain itu, hasil model juga dievaluasi terhadap tujuan bisnis yang telah ditetapkan sebelumnya, untuk memastikan bahwa model memberikan solusi yang relevan dan bernilai bagi bisnis.

6. *Deployment*

   Tahap terakhir merupakan deployment, yang bertujuan untuk menggunakan model yang telah dibangun untuk membuat keputusan bisnis. Langkah-langkah pada tahap ini meliputi penerapan model dalam lingkungan produksi, pembuatan laporan untuk menyampaikan hasil model kepada pemangku kepentingan, dan pemantauan kinerja model secara terus-menerus untuk memastikan model tetap efektif dan relevan seiring

berjalannya waktu. Implementasi yang baik pada tahap ini akan memastikan bahwa model dapat memberikan dampak positif yang berkelanjutan bagi bisnis.

2.4.1 *Consine Similarity*

Cosine similarity adalah ukuran yang digunakan untuk mengukur kesamaan antara dua vektor dalam ruang multi-dimensi dengan menghitung kosinus dari sudut di antara keduanya. Ukuran ini sangat berguna dalam analisis teks, di mana dokumen atau kalimat dapat direpresentasikan sebagai vektor berdasarkan frekuensi kata. *Cosine similarity* dihitung dengan menggunakan rumus:

$$C_s = \frac{x.y}{||x||||y||}$$ Persamaan 2. 1 [46]

x dan *y* adalah dua vektor, *x.y* adalah produk dot dari kedua vektor, dan ||*x*|| serta ||*y*|| adalah magnitudo dari masing-masing vektor. Nilai *cosine similarity* berkisar antara -1 hingga 1; nilai 1 menunjukkan bahwa kedua vektor memiliki arah yang sama, nilai 0 menunjukkan bahwa mereka ortogonal (tidak ada kesamaan), dan nilai -1 menunjukkan bahwa mereka berlawanan arah. Keunggulan utama dari cosine similarity adalah kemampuannya untuk mengukur kesamaan berdasarkan orientasi vektor, bukan magnitudonya, sehingga sangat efektif dalam aplikasi seperti pencarian dokumen, analisis sentimen, dan sistem rekomendasi [46].

2.4.2 *Longest Common Subsequence*

Longest Common Subsequence (LCS) adalah konsep fundamental dalam teori algoritma yang digunakan untuk menentukan subsekuens terpanjang yang muncul dalam dua urutan atau lebih tanpa mengubah urutan relatif dari elemen-elemen tersebut. Dalam konteks ini, subsekuens adalah urutan yang dapat diperoleh dengan menghapus beberapa elemen dari urutan asli tanpa mengubah urutan sisa. Contohnya, jika kita memiliki dua string, S1 dan S2, LCS akan menemukan elemen-elemen yang muncul di kedua string dalam urutan yang sama, meskipun tidak harus berurutan secara berdekatan.

LCS dapat dihitung menggunakan pendekatan pemrograman dinamis, yang merupakan metode efisien untuk menyelesaikan masalah yang dapat dibagi menjadi sub-masalah yang lebih kecil. Proses ini melibatkan pembuatan tabel dua dimensi L dengan ukuran (m+1) \times (n+1), di mana m dan n adalah panjang dari string S1 dan S2. Tabel ini digunakan untuk menyimpan panjang LCS dari substring yang berbeda. Rumus dasar untuk menghitung panjang LCS adalah sebagai berikut:

1. Jika karakter pada posisi ke- i dari string S1 sama dengan karakter pada posisi ke- j dari string S2, maka:

$$L[i][j] = L[i-1][j-1] + 1$$ PER*Persamaan 2. 2*

Ini berarti bahwa kita menemukan satu karakter tambahan dalam LCS.

2. Jika karakter tersebut tidak sama, maka kita mengambil nilai maksimum antara dua kemungkinan: panjang LCS tanpa karakter terakhir dari salah satu string:

$$L[i][j] = \backslash max(L[i-1][j], L[i][j-1])$$ *Persamaan 2. 3*

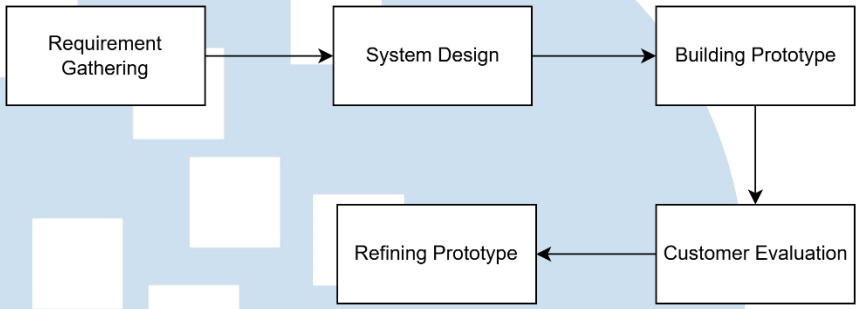
Dengan menggunakan rumus ini, kita dapat mengisi tabel secara iteratif hingga mencapai sel terakhir, yaitu L[m][n] yang memberikan panjang dari LCS antara kedua string. Selain itu, algoritma ini juga memungkinkan kita untuk membangun kembali subsekuens itu sendiri dengan melacak langkah-langkah yang diambil selama pengisian tabel.

LCS memiliki banyak aplikasi praktis, termasuk dalam pengolahan teks untuk membandingkan file atau dokumen, dalam analisis sekuens biologis untuk membandingkan gen atau protein, serta dalam sistem versi kontrol untuk mendeteksi perubahan antara versi dokumen. Dengan demikian, pemahaman tentang LCS sangat penting bagi para peneliti dan praktisi di berbagai bidang ilmu komputer dan bioinformatika.

2.3.2 *Prototype*

Prototype merupakan pendekatan dalam pengembangan sistem yang berlandasakan pada konsep evolusi, di mana proses ini memiliki dampak signifikan terhadap keseluruhan pengembangan. Dalam metode ini, sebuah prototipe dari sistem aplikasi dibuat untuk memungkinkan pengguna melakukan evaluasi dan pengujian terhadap ide ide yang diajukan oleh pengembang sebelum implementasi dilakukan. *Prototype* berfungsi sebagai alat komunikasi yang efektif, membantu penguna untuk memahami dan menguji fungsionalitas sistem, serta memberikan umpan balik yang berharga mengenai kebutuhan spesifik mereka yang mungkin tidak dipertimbangkan selama fase desain awal. Pentingya *prototyping* terletak pada kemampuannya untuk memperjelas dan mendefisikan persyaratan pengguna dengan lebih baik [47]. Dengan melibatkan pengguna dalam proses pengembangan, pengembang dapat mengidentifikasi dan mengatasi potensi masalah sejak dini, sehingga meningkatkan kemungkinan bahwa sistem akhir aka memenuhi harapan pengguna. *Prototyping* juga memfasilitasi interaksi yang lebih baik antara pengembang dan pengguna, memungkinkan mereka untuk berkolaborasi secara lebih efektif dalam menciptakan

solusi yang sesuai dengan kebutuhan nyata. Model prototipe cocok digunakan untuk proyek-proyek yang memiliki tingkat ketidakpastian yang tinggi, kebutuhan pengguna yang sulit didefinisikan secara lengkap di awal proyek, atau proyek yang memerlukan interaksi yang intensif antara pengembang dan pengguna.



Gambar 2. 2 Tahapan Metode *Prototype*

Pendekatan ini memungkinkan tim pengembang untuk mendapatkan umpan balik langsung dari pengguna, yang dapat digunakan untuk menyempurnakan sistem secara bertahap. Proses ini terdiri dari beberapa tahapan penting, mulai dari pengumpulan kebutuhan pengguna hingga penyempurnaan prototipe berdasarkan evaluasi. Dengan pendekatan ini, diharapkan dapat tercipta sistem yang lebih sesuai dengan kebutuhan pengguna dan lebih efisien dalam pengembangan. Pada Gambar 2.2 diperlihatkan tahapan dalam metode *prototype* diantaranya:

1. *Requirement Gathering*

   Tahap pertama dalam model prototipe adalah pengumpulan kebutuhan, yang bertujuan untuk memahami kebutuhan dasar pengguna dan sistem yang akan dibangun. Pada tahap ini, dilakukan wawancara, survei, atau analisis dokumen untuk mengidentifikasi fitur-fitur yang diperlukan dalam sistem. Informasi yang diperoleh dari tahap ini sangat penting untuk menentukan arah pengembangan prototipe agar sesuai dengan harapan pengguna.

2. *System Design*

   Setelah kebutuhan dikumpulkan, tahap berikutnya adalah desain cepat, yang bertujuan untuk membuat desain awal dari prototipe yang sederhana dan fungsional. Desain ini mencakup pembuatan sketsa antarmuka pengguna, alur kerja, dan fitur-fitur utama yang akan diimplementasikan. Tujuan dari desain cepat adalah untuk memberikan gambaran visual yang jelas tentang sistem yang akan dibangun, meskipun dalam bentuk yang masih kasar dan dapat dengan mudah diubah.

3. *Building Prototype*

Pada tahap ini, prototipe dibangun berdasarkan desain yang telah dibuat sebelumnya. Prototipe yang dibangun berfungsi untuk mengimplementasikan fitur-fitur yang telah diidentifikasi dalam bentuk yang dapat digunakan oleh pengguna. Meskipun belum sepenuhnya final, prototipe ini memungkinkan pengguna untuk berinteraksi dengan sistem dan memberikan gambaran yang lebih jelas tentang bagaimana sistem akan bekerja.

4. *Customer Evaluation*

   Setelah prototipe dibangun, tahap selanjutnya adalah evaluasi pelanggan, yang bertujuan untuk mendapatkan umpan balik dari pengguna mengenai prototipe yang telah dibangun. Pada tahap ini, prototipe ditunjukkan kepada pengguna dan mereka diminta untuk memberikan masukan mengenai kelebihan dan kekurangan sistem. Umpan balik yang diperoleh pada tahap ini sangat penting untuk memastikan bahwa prototipe tersebut sesuai dengan kebutuhan pengguna dan dapat diperbaiki lebih lanjut.

5. *Refining Prototype*

   Berdasarkan umpan balik yang diperoleh dari evaluasi pelanggan, tahap terakhir adalah penyempurnaan prototipe. Pada tahap ini, perubahan dilakukan pada prototipe sesuai dengan masukan yang diterima dari pengguna. Penyempurnaan ini bertujuan untuk memperbaiki dan meningkatkan kualitas prototipe agar lebih memenuhi kebutuhan pengguna dan lebih siap untuk diterapkan dalam lingkungan produksi.

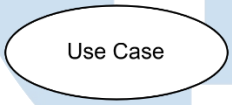
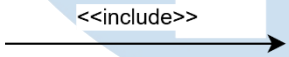
2.3.3 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) merupakan sebuah bahasa atau metode yang umum digunakan dalam penegambangan sistem untuk menggambarkan alur, fungsi, tujuan, dan mekanisme sistem tersebut. UML mempunyai dua kategori utama, yaitu *Structural Modeling Diagrams* dan *Behavior Modeling Diagrams*. Dalam penelitian ini, fokus akan diberikan pada diagram dari kategori *Behavior Modeling Diagrams*, yaitu *use case diagram*, *class diagram*, dan *activity diagram*. *Diagram use case* digunakan untuk menggambarkan fungsionalitas sistem serta interaksi antara pengguna dan sistem. Sementara itu, *class diagram* menampilkan struktur statis dari sistem dengan menunjukan kelas-kelas, atribut, dan hubungan antar kelas. Sedangkan *activity diagram* menggambarkan alur kerja dan langkah-langkah yang terlibat dalam proses tertentu dalam sistem, memvisualisasikan bagaimana kontrol mengalir dari satu aktivitas ke aktivitas lainnya [48].

1. *Use Case Diagram*

Use case diagram berfungsi untuk menggambarkan interaksi antara pengguna, yang disebut sebagai aktor, dan sistem yang sedang dikembangkan. *Diagram* ini menunjukan tindakan atau fungsi yang dapat dilakukan oleh aktor dalam konteks sistem tersebut . Dalam use case *diagram*, terdapat berbagai jenis simbol yang digunakan untuk menggambarkan fungsi dan peran dalam sistem, yang dijelaskan secara rinci dalam tabel yang mencakup bentuk simbol, notasi, dan fungsi masing-masing dan akan di jelaskan pada Tabel 2.2.

Tabel 2. 2 Simbol Use Case *Diagram*

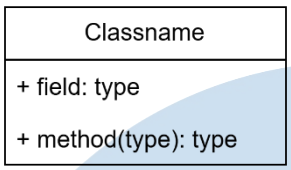
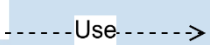
Simbol	Notasi	Fungsi
	*Actor*	Mempresentasikan pengguna yang berinteraksi dengan sistem.
	*Use Case*	Menggambarkan dekripsi aksi sesuai urutan yang terjadi
	*Include*	Tambahan dari *use case* yang diperlukan agar *use case* lain dapat berjalan
	Association	Menunjukan hubungan antara aktor dan *use case*
	*Extension*	Tambahan dari *use case* yang biasa berdiri sendiri tanpa ketergantungan
	*Generalization*	Menunjukan hubungan hirarki antara objek *parent* dan *child*

2. *Class Diagram*

Class diagram merupakan alat penting dalam pemodelan sistem yang digunakan untuk menggambarkan struktur statis dari sebuah sistem, terutama dalam konteks pengembangan perangkat lunak bebasis objek. Tabel 2.3 merupakan penjelasan mendetail mengenai *class diagram* termasuk simbol, notasi, dan fungsinya.

Tabel 2. 3 Simbol *Class Diagram*

Simbol	Notasi	Fungsi

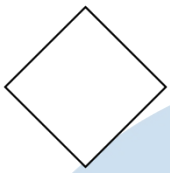
	*Class*	Elemen utama dalam *diagram* kelas, terbagi menjadi tiga bagian utama.
	Association	Menunjukan hubungan antara dua kelas.
	Generalization	Menggambarkan hubungan antara kelas induk dan kelas anak
	*Dependency*	Menyatakan bahwa suatu kelas memiliki ketergantungan terhadap kelas lainna.
	Aggregation	Menunjukan bahwa sebuah kelas merupakan bagian dari kelas lainnya.

3. *Activity Diagram*

Activity diagram merupakan model yang menggambarkan dinamika sistem setelah pembuatan *use case diagram. Diagram* ini menampilkan alur kerja secara terstruktur berdasarkan *use case* yang telah ada, dengan setiap aktivitas yang dilakukan durepresentasikan menggunakan notasi yang sesuai dengan fungsinya. Terdapat berbagai simbol yang digunakan dalam *activity diagram* yang ditampilkan pada Tabel 2.4.

Tabel 2. 4 Simbol *Activity Diagram*

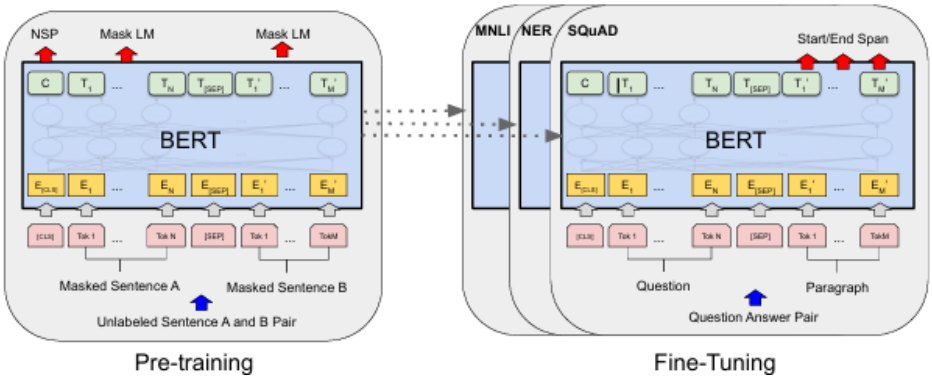
Simbol	Notasi	Funsi
	Activity	Memuat semua interaksi dalam *activity diagram*
	Start Point	Menandai permulaan dari suatu aktivitas
	Action	Menunjukan proses yang berlansung dalam *activiy diagram.*

	Decision	Digunakan saat ada kondisi yang membutuhkan keputusan
	Line Connector	Berungsi untuk menunjukan hubungan antar objek

2.3.4 *Birectional Encoder Representations from Transformers* (BERT)

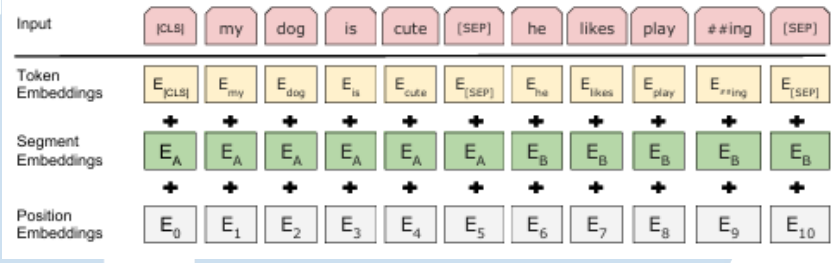
Birectional Encoder Representations from Transformers (BERT) merupakan model bahasa yang revolusioner dalam pemrosesan bahasa alami *Natural Language Processing* (NLP) yang dikembangkan oleh Google AI pada tahun 2018 [49]. Model ini memperkenalkan pendekatan bidirectional, yang memungkinkan pemahaman konteks dari kedua arah kalimat. BERT termasuk model *deep learning* yang berbasis arsitektur *Transformers* dan memiliki kemampuan untuk memahami konteks teks secara dua arah. Berbeda dengan model-model sebelumnya yang hanya menganalisis urutan teks dari kiri ke kanan atau sebaliknya. BERT menggunakan teknik yang disebut dengan *Masked Language Modeling* [50]. Teknik ini memungkinkan pelatihan dua arah. Dalam keadaan default, arsitektur *Transformers* terdiri dari dua mekanisme terpisah: encoder yang memproses teks input dan decoder yang menghasilkan prediksi. Tujuan utama BERT adalah untuk membangun model bahasa, sehingga hanya mekanisme encoder yang digunakan. Google meluncurkan dua versi dari BERT, yaitu BERTBASE dan BERTLARGE, yang memiliki spesifikasi sebagai berikut [50]:

1. BERTBASE → L = 12(jumlah lapisan), H =768 (dimensi output), A = 12 (jumlah multi headed attenton), total parameter = 110M
2. BERTLARGE → L = 24, H = 1023, A =16, total parameter = 340M



Gambar 2. 3 BERT *Pre-training* dan *Fine-Tuning*

Penggunaan BERT terbagi menjadi dua fase: pre-trained dan fine-tuning yang ditunjukan pada Gambar 2.3. Pada fase *pre-training*, model dilatih dengan data tak berlaber. Sedangkan pada fase *fine-tuning*, model BERT diinisialisasi dengan parameter yang telah dilatih sebelumnya dan kemudian disesuaikan dengan data berlabel untuk tugas tertentu. Setiap tugas downstream memiliki model fine-tuned yang terpisah meskipun mereka dimulai dari parameter pre-trained yang sama. Fase *pre-training* terdiri dari dua tugas prediktif tanpa pengawasan: *Masked Language Model*.

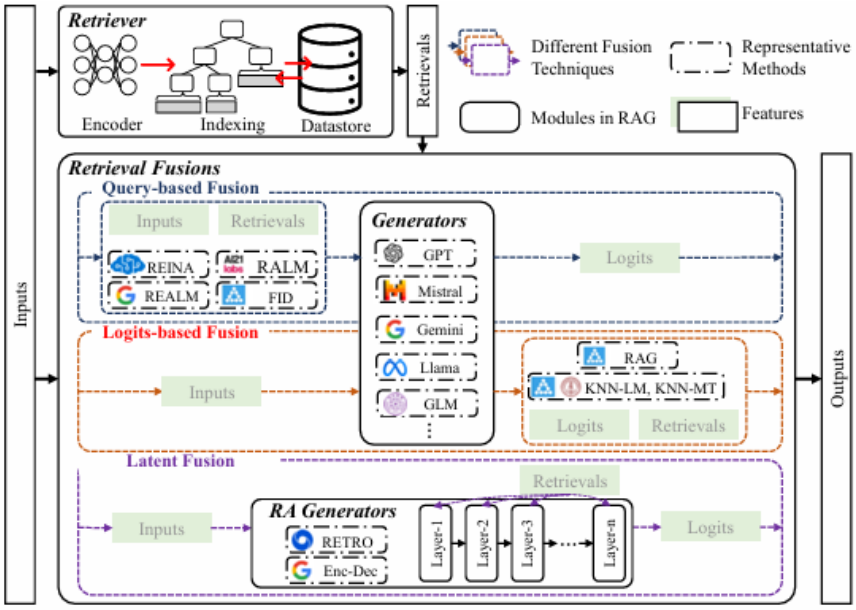


Gambar 2. 4 BERT *input representation*

Gambar 2.4 menggambarkan proses pengolahan input teks menggunakan model seperti BERT. Teks "*My dog is cute. He likes playing.*" dipisahkan menjadi token-token, dan masing-masing token dikonversi menjadi vektor numerik atau *token embeddings*, yang mencerminkan makna semantik dari setiap token [51]. Setiap token kemudian diberi *segment embedding* untuk menandakan segmen kalimat mana yang sedang diproses, serta *position embedding* untuk menunjukkan urutan token dalam kalimat. Semua vektor *embedding* ini dijumlahkan untuk membentuk representasi yang lebih kaya dan kontekstual dari teks yang akan diproses lebih lanjut oleh model [51].

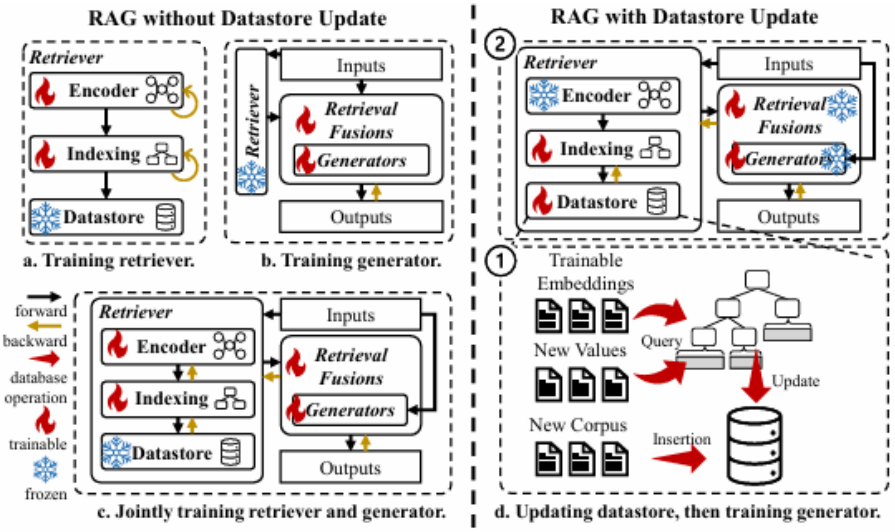
2.3.5 *Retrieval-Augmentasi Generation* (RAG)

Retrieval-Augented Generation (RAG) merupakan sebuah paradigm yang bertujuan untuk meningkatkan kemampuan *generative* model bahasa alami dengan integrasi informasi dari basis data eksternal. RAG dapat memperkaya informasi yang disimpan sehingga dapat meningkatkan akurasi dan kredibilitas generasi, terutama untuk tugas-tugas yang intensif pengetahuan [52]. Terdapat beberapa komponen utama dalam RAG yaitu *retriever*, *fusion techiques*, dan *generator* yang ditunjukan pada Gambar 2.5.



Gambar 2. 5 Overview RAG

Retriever bertanggung jawab untuk mencari informasi yang relevan dari basis data eksternal berdasarkan input yang diberikan. Proses ini melibatkan penggunaan algoritma pencarian yang efektif untuk menemukan informasi yang paling relevan. *Fusion techniques* merupakan informasi yang diretrieved kemudian difuse dengan input atau status intemediat menggunakan teknik fusi yang berbeda-beda. Sedangakan *generator* mernggunakan informasi yang telah difuse untuk membuat prediksi final berdasarkan input dan retrival yang relevan. Sehingga, hasil akhir dari generator dapat berupa teks yang lebih akurat dan relevan [52]. Strategi model RAG dapat dilakukan dengan dua strategi utama yaitu RAG dengan *Update Datastore*, dimana model akan dilatih dengan basis data eskternal yang dapat di perbaharui secara berkala untuk memastikan informasi yang up-to-date. Dan yang kedua RAG tanpa *Update Datastore*, dimana model dilatih dengan basis data eksternal statis tanpa update.



Gambar 2. 6 RAG with/without Datastore Update

2.4 *Tools* Penelitian

2.4.3 Python

Python merupakan bahasa pemograman yang diinterpretasikan, iteraktif, dan berorientasi objek. Bahasa ini mengintegrasikan berbagai fitur seperti modul, penanganan pengecualian, pengetikan dinamis, dan tipe data dinamis yang berlevel tinggi, serta kelas-kelas. Python mendukung berbagai paradigma berorientasi objek, prosedural, dan fungsional. Sintaks python yang jelas dan sederhana menjadikannya pilihan yang populer di kalangan pengembang. Python dapat digunakan untuk berbagai aplikasi, termasuk pengembangan web dengan *framework* seperti Django dan Flask, pengembangan antarmuka grafis (GUI) menggunakan Tkinter atau PySide, serta untuk analisis ilmiah dan numerik dengan pustaka seperti Pandas dan SciPy. Selain itu, Python juga banyak digunakan dalam pengembangan perangkat lunak dengan alat seperti Buildbot dan Trac [53].

2.4.4 Flask

Flask adalah sebuah *microframework* yang ditulis dalam bahasa pemrograman Python, dirancang untuk memudahkan pengembang dalam membangun aplikasi web dengan cara yang sederhana dan fleksibel. Kerangka kerja ini memanfaatkan dua pustaka utama, yaitu Werkzeug untuk penanganan HTTP dan Jinja2 untuk templating. Salah satu keunggulan Flask adalah sifatnya yang minimalis, tidak memaksakan struktur tertentu pada pengembang, sehingga mereka memiliki kebebasan untuk memilih komponen dan pustaka yang ingin digunakan.

Flask juga mudah dipelajari berkat sintaks yang sederhana, menjadikannya pilihan ideal bagi pemula yang ingin belajar pengembangan web. Meskipun merupakan micro *framework*, Flask mendukung berbagai ekstensi yang dapat menambah fungsionalitas, seperti pemetaan objek-relasional dan sistem otentikasi. Selain itu, Flask sangat cocok untuk membangun RESTful API, memungkinkan pengembang untuk membuat aplikasi yang ringan dan efisien. Dikenal karena kemampuannya menangani proyek-proyek skala besar, Flask telah digunakan dalam berbagai aplikasi terkenal seperti Pinterest dan LinkedIn. Dengan komunitas yang aktif dan dokumentasi yang lengkap, Flask menjadi pilihan yang tepat bagi banyak pengembang Python yang ingin menciptakan aplikasi web yang dinamis dan responsif [54].

2.4.5 Laravel

Laravel adalah *framework* PHP open-source yang dirancang untuk mempermudah dan mempercepat proses pengembangan aplikasi web. Menggunakan pola arsitektur *Model-View-Controller* **(**MVC**)**, Laravel memisahkan antara logika , tampilan, dan pengelolaan data dalam struktur yang terorganisir., sehingga memudahkan pengembang dalam membangun aplikasi yang terstruktur dan mudah dipelihara. Salah satu fitur unggulan dari Laravel adalah Eloquent ORM, yang memungkinkan interaksi dengan basis data menggunakan sintaks yang intuitif dan elegan, sehingga mengurangi kebutuhan untuk menulis *query* SQL secara manual. Laravel juga dilengkapi dengan berbagai alat dan pustaka yang mendukung manajemen rute, otentikasi pengguna, dan pengolahan formulir, serta menyediakan sistem *caching* yang efisien untuk meningkatkan performa aplikasi. Dengan komunitas yang aktif dan dokumentasi yang komprehensif, Laravel telah menjadi salah satu pilihan utama bagi pengembang dalam menciptakan aplikasi web dari skala kecil hingga besar [55].

2.4.6 MySQL

MySQL adalah sistem manajemen basis data relasional (RDBMS) yang menggunakan *Structured Query Language* (SQL) sebagai bahasa utama untuk mengakses dan mengelola data. MySQL bersifat open-source, yang berarti pengguna dapat mengunduh, memodifikasi, dan mendistribusikannya secara gratis, meskipun ada juga versi komersial dengan dukungan tambahan dari *Oracle Corporation.* MySQL dikenal karena kecepatan dan kinerjanya yang tinggi dalam menangani kueri kompleks serta volume data yang besar, menjadikannya pilihan populer untuk aplikasi

web, *e-commerce*, dan sistem manajemen perusahaan. Dengan arsitektur *client-server*, MySQL memungkinkan beberapa pengguna untuk terhubung ke *server* dan melakukan operasi *database* secara bersamaan. Fitur-fitur unggulan MySQL termasuk keamanan yang kuat, kemampuan replikasi data, dan skalabilitas yang baik untuk menangani pertumbuhan data. MySQL telah digunakan oleh banyak situs web besar dan aplikasi terkenal seperti Facebook, Twitter, dan YouTube, menunjukkan kemampuannya dalam menangani proyek-proyek berskala besar [56].

2.4.7 SQLAlcemy

SQLAlchemy adalah *toolkit* SQL dan *Object-Relational Mapping* (ORM) yang kuat untuk Python, yang memungkinkan pengembang untuk berinteraksi dengan basis data menggunakan objek Python alih-alih menulis kueri SQL secara langsung. Dengan SQLAlchemy, pengembang dapat memetakan kelas Python ke tabel dalam basis data, sehingga memudahkan pengelolaan data dan menjaga keterpisahan antara model objek dan skema basis data. SQLAlchemy terdiri dari dua komponen utama: *Core*, yang menyediakan lapisan abstraksi untuk mengelola koneksi basis data dan membangun kueri SQL, serta ORM, yang memungkinkan pemetaan objek ke tabel dan operasi berbasis objek pada data. Fitur-fitur unggulan SQLAlchemy termasuk dukungan untuk berbagai jenis basis data, manajemen sesi yang efisien, serta kemampuan untuk melakukan operasi CRUD (Create, Read, Update, Delete) dengan cara yang intuitif [57].

2.4.8 API Chat GPT-4

API Chat GPT-4 adalah antarmuka pemrograman aplikasi yang disediakan oleh OpenAI, memungkinkan pengembang untuk mengintegrasikan kemampuan model bahasa ChatGPT-4 ke dalam aplikasi mereka. Dengan menggunakan API ini, pengembang dapat memanfaatkan teknologi pemrosesan bahasa alami (NLP) untuk menciptakan interaksi yang lebih cerdas dan responsif dalam aplikasi, seperti chatbot atau sistem rekomendasi. ChatGPT-4 API mendukung berbagai bahasa pemrograman dan dirancang untuk mudah diintegrasikan, sehingga memudahkan pengembang dalam membangun antarmuka percakapan yang interaktif. Selain itu, API ini memberikan akses ke model-model canggih seperti GPT-4-3 dan GPT-4, yang mampu menghasilkan teks berkualitas tinggi dengan konteks yang relevan [58].

2.4.9 Visual Studio Code

Visual Studio Code (VSCode) adalah *source-code editor* yang dikembangkan oleh Microsoft, dirancang untuk meningkatkan produktivitas pengembang dengan menyediakan berbagai fitur canggih. Tersedia untuk macOS, Linux, dan Windows, VSCode dilengkapi dengan *IntelliSense*, yang memberikan saran otomatis mengenai fungsi dan parameter saat menulis kode, serta dukungan untuk berbagai bahasa pemrograman melalui ekstensi yang dapat diunduh. Fitur debugging interaktif memungkinkan pengguna untuk melacak dan memperbaiki kesalahan dalam kode secara efisien, sementara integrasi kontrol versi memudahkan manajemen proyek dan kolaborasi tim. VSCode mendukung berbagai bahasa pemrograman seperti JavaScript, TypeScript, dan Node.js secara bawaan, serta memiliki ekosistem yang kaya dengan ekstensi yang memungkinkan dukungan untuk bahasa lain seperti C++, C#, Java, dan Python [57].