

BAB II

TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

2.1.1. Penelitian Tentang Pencarian Nomor Surat dalam Sistem Arsip Elektronik

Beberapa penelitian telah membahas pengembangan sistem pencarian dan klasifikasi kode arsip berbasis digital. dalam "Implementasi Algoritma *Sequential Searching* untuk Pencarian Nomor Surat pada Sistem Arsip Elektronik" [3] mengembangkan sistem arsip elektronik yang memungkinkan pencarian nomor surat menggunakan algoritma *Sequential Searching*. Penelitian ini membantu proses pencarian nomor surat secara lebih efisien dibandingkan metode manual, tetapi tidak mengotomatiskan klasifikasi kode surat.

Sementara itu, dalam "Pembangunan Aplikasi Klasifikasi Kode Surat Berbasis Android Menggunakan Algoritma *Boyer-Moore* di Kantor Kecamatan Ciparay"[4] mengembangkan aplikasi berbasis Android untuk pencocokan kode klasifikasi arsip secara lebih cepat. Dengan menggunakan algoritma *Boyer-Moore*, pencocokan *string* kode arsip menjadi lebih efisien dibandingkan metode pencarian biasa. Namun, meskipun sistem ini dapat mempercepat pencarian kode arsip, pegawai masih harus melakukan pencocokan kode secara manual, sehingga tetap memerlukan waktu dan keahlian dalam memilih kode yang sesuai.

Dari penelitian-penelitian ini, dapat disimpulkan bahwa pencarian kode arsip secara digital dapat meningkatkan efisiensi dibandingkan metode manual, tetapi masih memiliki keterbatasan karena belum mengotomatiskan proses klasifikasi kode surat. Oleh karena itu, penelitian ini dikembangkan lebih lanjut dengan menggunakan AI untuk otomatisasi klasifikasi kode arsip, sehingga pegawai tidak perlu lagi mencari dan mencocokkan kode secara manual, melainkan sistem secara otomatis menentukan kode klasifikasi yang tepat berdasarkan isi surat, sesuai dengan Permendagri Nomor 83 Tahun 2022.

2.1.2. Penelitian yang menggunakan *Semantic Search*

Dalam studi berjudul "Implementasi Semantic Search untuk Mesin Pencarian pada Ensiklopedia Masjid Bersejarah di Indonesia" [5] menunjukkan bahwa penerapan semantic search secara signifikan meningkatkan efisiensi pencarian informasi dibandingkan metode konvensional. Dalam penelitian ini, dilakukan proses indexing, pembuatan dan pemetaan ontologi menggunakan perangkat lunak Smore, serta penandaan SPO (Subject, Predicate, Object) melalui library Jena API. Dengan pendekatan tersebut, sistem mampu menghasilkan pencarian yang lebih tepat karena mempertimbangkan konteks kalimat, bukan hanya mencocokkan kata demi kata. Hasilnya, pengguna tidak perlu lagi melakukan pencarian ulang dengan kata kunci berbeda, karena sistem sudah mampu menyajikan dokumen yang relevan sejak awal.

Studi lain yang mendukung adalah penelitian berjudul "Development of a Semantic Web Based Scientific Journal Search Application" [6], yang memanfaatkan teknologi Web Semantik untuk membangun mesin pencarian jurnal ilmiah. Dengan pendekatan pengembangan ontologi dan pemanfaatan framework Apache Jena, sistem ini diuji dan menghasilkan nilai recall, precision, dan F1-score sebesar 0.8, yang menandakan bahwa hasil pencarian cukup relevan dan akurat. Penelitian ini menegaskan bahwa semantic search mampu memahami hubungan antar entitas dan makna kata dalam konteks tertentu, sehingga pencarian informasi menjadi lebih cerdas dan efisien.

2.1.3. Penelitian tentang Digitalisasi Pemberian Nomor Surat dan Pengarsipan Arsip

Beberapa penelitian telah membahas pengembangan sistem berbasis web untuk pengelolaan surat masuk dan surat keluar di instansi pemerintahan. Dalam studi "Perancangan Aplikasi Dalam Pemberian Nomor Surat Di Dinas Pariwisata Kebudayaan Dan Ekonomi Kreatif Sumatera Utara Berbasis Web" [7] mengembangkan sistem untuk penyimpanan dan pencatatan nomor surat secara digital. Penelitian ini menunjukkan bahwa pengarsipan surat berbasis elektronik dapat mencegah kehilangan atau kerusakan dokumen, tetapi masih mengandalkan

penentuan nomor surat secara manual.

Sementara itu, dalam "Informasi Surat Masuk dan Surat Keluar Berbasis Web Pada Suatu Instansi" [8] juga mengembangkan sistem berbasis web untuk mempermudah penyimpanan, pencarian, dan pengelolaan dokumen surat. Dengan menggunakan model pengembangan waterfall dan database MySQL, penelitian ini menunjukkan bahwa sistem digital dapat mempercepat proses pencarian dan pengarsipan dokumen, tetapi belum mengimplementasikan sistem otomatis untuk klasifikasi kode surat.

Dari kedua penelitian ini, dapat disimpulkan bahwa penggunaan sistem berbasis web dapat meningkatkan efisiensi dalam pengelolaan dokumen administratif. Namun, keduanya masih mengandalkan metode manual dalam penentuan kode klasifikasi surat, sehingga belum sepenuhnya mengatasi permasalahan efisiensi dan akurasi dalam pemberian nomor surat.

2.1.4. Penelitian tentang Penerapan AI dalam Klasifikasi Dokumen

Dalam studi yang berjudul "Rancang Bangun Aplikasi Klasifikasi dan Digitalisasi Dokumen Berbasis Android Menggunakan ML KIT SDKs dan Algoritma BERT" [9] membahas penggunaan BERT dalam mengklasifikasikan dokumen digital secara otomatis. Penelitian ini mengembangkan aplikasi berbasis Android yang mampu mengenali teks menggunakan ML Kit SDK dan mengklasifikasikan dokumen seperti KTP dan SIM.

Hasil penelitian menunjukkan bahwa model BERT yang telah di-fine-tune dapat memberikan akurasi tinggi dalam klasifikasi dokumen, serta integrasi ML Kit SDK mampu melakukan pengenalan teks secara real-time. Namun, penelitian ini lebih berfokus pada digitalisasi dokumen identitas dan belum diterapkan dalam konteks klasifikasi kode arsip pemerintahan. Dengan adanya penelitian ini, dapat disimpulkan bahwa penerapan model NLP seperti BERT sangat potensial untuk digunakan dalam klasifikasi dokumen berbasis teks. Oleh karena itu, penelitian ini menjadi dasar bagi penelitian yang dilakukan, di mana pendekatan serupa akan diterapkan dalam klasifikasi kode arsip sesuai dengan Permendagri Nomor 83 Tahun 2022 dalam sistem berbasis web.

2.1.5. Pemanfaatan OpenAI API dalam Aplikasi Teks Otomatis

Dalam beberapa tahun terakhir, pemanfaatan layanan API dari penyedia model bahasa seperti OpenAI telah meningkat pesat seiring dengan kebutuhan otomatisasi dalam berbagai bidang, termasuk pemasaran, pendidikan, dan pengarsipan. Salah satu studi yang relevan dilakukan dalam pengembangan aplikasi bernama "Implementasi Model Bahasa OpenAI GPT-3 Untuk Aplikasi Text Content Generator Berbasis Web dan Aplikasi Mobile" [10], yang dirancang untuk membantu pelaku usaha kecil dan menengah menghasilkan konten pemasaran secara otomatis.

Aplikasi ini memanfaatkan OpenAI API, yang dalam penelitian tersebut menggunakan model bahasa GPT-3, untuk menghasilkan teks berbentuk caption pemasaran. Teknologi ini memungkinkan aplikasi menghasilkan teks yang kontekstual hanya dengan memberikan beberapa contoh input, menggunakan pendekatan few-shot learning. Sistem dikembangkan dalam dua platform, yaitu web dan aplikasi Android, sehingga mudah diakses oleh berbagai kalangan pengguna.

Penelitian ini menjadi bukti bahwa OpenAI API dapat diintegrasikan secara efektif ke dalam sistem informasi untuk mengotomatisasi proses pemrosesan dan penyusunan teks, serta dapat berfungsi sebagai solusi alternatif terhadap proses manual dalam kegiatan pengolahan dokumen. Dalam konteks sistem klasifikasi arsip, pendekatan ini juga dapat diadaptasi untuk menghasilkan kode klasifikasi berdasarkan isi teks surat, dengan dukungan model bahasa yang disediakan oleh OpenAI API.

2.1.6. Implementasi *Retrieval-Augmented Generation* (RAG) dalam Pembangunan Sistem Kecerdasan Buatan

Retrieval-Augmented Generation (RAG) adalah metode yang menggabungkan model generatif dengan sistem retrieval berbasis dokumen untuk memberikan respons yang lebih akurat dan berbasis bukti. dalam penelitian yang berjudul "Implementasi *Retrieval-Augmented Generation* (RAG) Dalam Perancangan Chatbot Kesehatan Pencernaan" [11] mengimplementasikan RAG

untuk meningkatkan kualitas chatbot di bidang kesehatan pencernaan. Dalam penelitian ini, RAG digunakan untuk mengurangi bias dan meningkatkan akurasi informasi yang disampaikan oleh chatbot, terutama pada topik kritis seperti kesehatan. Dengan menggunakan model bahasa Indonesia dan database vektor untuk pencarian dokumen, penelitian ini menunjukkan bagaimana penerapan RAG dapat mengurangi halusinasi informasi dan meningkatkan relevansi jawaban. Hasil evaluasi menunjukkan bahwa sistem dapat memberikan jawaban yang lebih akurat dan relevan, yang menunjukkan potensi besar penerapan RAG dalam berbagai aplikasi berbasis AI, termasuk sistem klasifikasi kode arsip.

Pendekatan serupa dapat diterapkan dalam penelitian ini untuk menggunakan RAG dalam klasifikasi kode arsip, di mana informasi relevan tentang kode arsip dapat diambil dari *database* atau dokumen lainnya, kemudian digabungkan dengan model klasifikasi untuk menyediakan *output* yang lebih tepat dan sesuai dengan konteks.

Tabel 2.1 yang ada dibawah ini adalah tabel perbandingan penelitian terkait.

Tabel 2. 1 Tabel Perbandingan Penelitian Terkait

No	Nama dan Tahun	Judul Penelitian	Metode / Teknologi	Hasil Penelitian
1	A. Sonita and M. Sari, 2018	<i>Implementasi Algoritma Sequential Searching untuk Pencarian Nomor Surat pada Sistem Arsip Elektronik</i>	Sequential Searching	Pencarian nomor surat lebih efisien dibanding manual, tetapi belum melakukan klasifikasi kode arsip secara otomatis.

No	Nama dan Tahun	Judul Penelitian	Metode / Teknologi	Hasil Penelitian
2	Y. Muharam and Y. Bahtiar, 2021	<i>Pembangunan Aplikasi Klasifikasi Kode Surat Berbasis Android Menggunakan Algoritma Boyer-Moore di Kantor Kecamatan Ciparay</i>	Algoritma Boyer-Moore (string matching)	Pencocokan kode arsip lebih cepat, tetapi pegawai masih harus mencocokkan kode secara manual.
3	R. D. Farisah, 2014	<i>Implementasi Semantic Search untuk Mesin Pencarian pada Ensiklopedia Masjid Bersejarah di Indonesia</i>	Semantic Search, Ontologi, Indexing, Jena API	Pencarian lebih akurat karena memahami konteks; pengguna tidak perlu mengulang pencarian dengan kata berbeda.

No	Nama dan Tahun	Judul Penelitian	Metode / Teknologi	Hasil Penelitian
4	Vaiye E. J. Karame, Agustinus Jacobus, and Dirko G.S. Ruindungan, 2024	<i>Development of a Semantic Web Based Scientific Journal Search Application</i>	Web Semantik, Ontologi, Apache Jena	Mendapatkan nilai recall, precision, dan F1-score = 0.8; pencarian jurnal menjadi lebih relevan & akurat.
5	R. Pardomuan, M. R. S. Pahlevi, dkk, 2023	<i>Perancangan Aplikasi Dalam Pemberian Nomor Surat Berbasis Web di Dinas Pariwisata Kebudayaan dan Ekonomi Kreatif Sumatera Utara</i>	Aplikasi Web, Database digital	Penyimpanan dan pencatatan surat lebih teratur, tetapi penentuan nomor surat masih manual.

No	Nama dan Tahun	Judul Penelitian	Metode / Teknologi	Hasil Penelitian
6	A. Nouvel, S. Sutrisno, and R. Indriani, 2021	<i>Informasi Surat Masuk dan Surat Keluar Berbasis Web Pada Suatu Instansi</i>	Aplikasi Web, Model Waterfall, MySQL	Pencarian dan penyimpanan surat lebih cepat, namun klasifikasi kode surat tidak otomatis.
7	I. M. ISMAIL, 2024	<i>Rancang Bangun Aplikasi Klasifikasi dan Digitalisasi Dokumen Berbasis Android Menggunakan ML KIT SDKs dan Algoritma BERT</i>	OCR ML Kit, Model BERT (fine-tune)	BERT memberikan akurasi tinggi untuk klasifikasi dokumen identitas, tetapi belum diterapkan ke klasifikasi kode arsip pemerintahan.

No	Nama dan Tahun	Judul Penelitian	Metode / Teknologi	Hasil Penelitian
8	K. M. Fitria, 2022	<i>Implementasi Model Bahasa OpenAI GPT-3 Untuk Aplikasi Text Content Generator Berbasis Web dan Mobile</i>	OpenAI API (GPT-3), Few-shot learning	Berhasil menghasilkan teks otomatis (caption), membuktikan API dapat mengotomatisasi pembuatan konten berbasis teks.
9	G. Samudra, Ahmad Turmudi Zy, and Ermanto, 2025.	<i>Implementasi Retrieval-Augmented Generation (RAG) Dalam Perancangan Chatbot Kesehatan Pencernaan</i>	RAG, Model Bahasa Indonesia, Vector Database	Meningkatkan akurasi jawaban, mengurangi halusinasi, dan relevansi jawaban meningkat; menunjukkan potensi besar RAG untuk klasifikasi dokumen berbasis konteks.

2.2. Landasan Teori

2.2.1. Klasifikasi Dokumen

Klasifikasi arsip merupakan salah satu aspek fundamental dalam pengelolaan arsip dinamis. Menurut Peraturan Kepala Arsip Nasional Republik Indonesia Nomor 19 Tahun 2012 [12], klasifikasi arsip didefinisikan sebagai pola pengaturan arsip secara berjenjang dari hasil pelaksanaan fungsi dan tugas instansi menjadi beberapa kategori unit informasi kearsipan. Klasifikasi ini dibuat agar arsip dapat dikelola, ditelusuri, dan dipelihara secara sistematis, logis, dan efisien. Klasifikasi Arsip adalah alat rekam kegiatan organisasi merupakan pola pengaturan arsip secara berjenjang dari hasil pelaksanaan tugas dan fungsi instansi menjadi beberapa kategori unit informasi kearsipan [13]. Klasifikasi Arsip sebagai sarana untuk penataan dan penemuan kembali arsip di lingkungan Pemerintah Daerah dalam memberkaskan arsip yang diciptakan dan diterima.

Secara teknis, sistem klasifikasi arsip dibangun berdasarkan jenjang fungsi organisasi yang tersusun secara hierarkis. Pada jenjang paling atas adalah fungsi (primer), yaitu urusan atau tanggung jawab besar yang menjadi wewenang suatu instansi atau unit kerja, seperti pendidikan, kesehatan, atau keuangan. Selanjutnya, di bawah fungsi tersebut terdapat kegiatan (sekunder), yaitu berbagai aktivitas yang dilakukan untuk menjalankan fungsi tersebut, misalnya pengadaan barang, penyusunan laporan, atau pelaksanaan pelatihan. Terakhir, pada jenjang paling rinci terdapat transaksi (tersier), yaitu bentuk komunikasi atau tindakan administratif spesifik yang terjadi dalam pelaksanaan suatu kegiatan, seperti surat permohonan, nota dinas, atau laporan hasil kegiatan. Struktur ini membantu dalam mengorganisasi arsip secara sistematis sehingga memudahkan pencarian dan pengelolaan dokumen di lingkungan instansi pemerintah.

Kode Klasifikasi Arsip berfungsi tidak hanya sebagai pengenal struktur informasi, namun juga menjadi bagian utama dalam sistem penomoran Naskah Dinas. Berdasarkan Peraturan Arsip Nasional Republik Indonesia Nomor 9 Tahun 2022 [14], disebutkan bahwa Kode Klasifikasi Arsip digunakan sebagai dasar

penomoran naskah dinas, serta menjadi acuan untuk pemeliharaan arsip aktif dan inaktif, Penyusutan arsip, dan Penelusuran arsip berdasarkan urusan dan fungsi.

Berdasarkan Buku Petunjuk Penggunaan Permendagri Nomor 83 Tahun 2023 [1], kode klasifikasi arsip digabungkan dengan nomor urut dokumen dan tahun penerbitan untuk membentuk format penomoran arsip yang baku dan seragam. Format ini bertujuan untuk menjaga keteraturan administrasi, kemudahan pelacakan, dan akuntabilitas arsip di lingkungan instansi, terutama instansi pemerintahan.

Format penomoran tersebut dituliskan sebagai berikut: kodefikasi/urut/tahun

Tabel 2. 2 Format penamaan kode klasifikasi arsip

kodefikasi	kode klasifikasi arsip sesuai dengan Permendagri 83 Tahun 2023
urut	nomor urut dokumen selama tahun berjalan dalam suatu instansi
tahun	Tahun dokumen diarsipkan

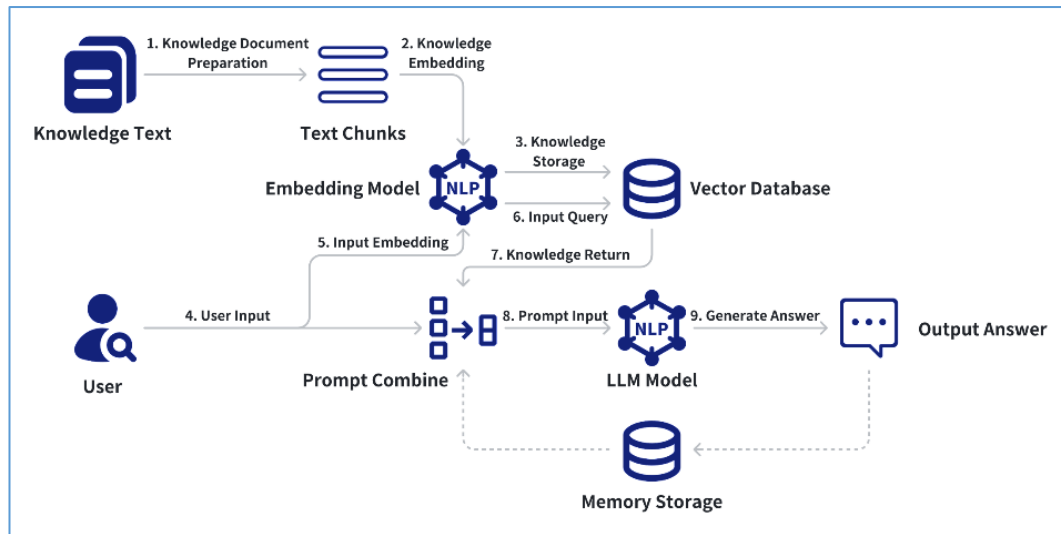
2.2.2. *Natural Language Processing (NLP)*

Natural Language Processing (NLP) adalah cabang dari kecerdasan buatan (*Artificial Intelligence/AI*) yang berfokus pada interaksi antara komputer dan bahasa manusia alami. Tujuan utama NLP adalah memungkinkan komputer untuk memahami, menginterpretasikan, dan menghasilkan bahasa manusia secara efisien dan bermakna [15]. Teknologi ini menggabungkan aspek linguistik komputasional, pemodelan berbasis aturan, serta teknik pembelajaran mesin dan pembelajaran mendalam untuk memproses bahasa manusia.

2.2.3. *Retrieval Augmented Generation (RAG)*

Retrieval Augmented Generation (RAG) adalah metode untuk meningkatkan model bahasa besar (LLM) melalui penggabungan data informasi tambahan dari sumber pengetahuan eksternal. Hal ini dapat mengurangi halusinasi dan memungkinkan respons LLM yang lebih tepat dan sadar konteks. Metode ini sangat

berguna dalam tugas-tugas seperti menjawab pertanyaan, meringkas dokumen, dan agen percakapan. [16]. Secara umum RAG dapat distrukturkan menjadi workflow seperti yang bisa dilihat pada gambar 3.1



Gambar 2. 1 Workflow RAG¹

Proses kerja RAG dimulai dari tahap Document Preparation, di mana dokumen dipersiapkan dan diubah menjadi format teks yang seragam. Pada tahap ini dilakukan pembersihan data (*cleaning*), penghapusan karakter non-informatif, serta pembagian teks menjadi potongan kecil (*chunking*) agar mudah dikelola oleh model. Tahap berikutnya adalah *Knowledge Embedding*, yaitu proses mengubah potongan teks hasil persiapan dokumen menjadi representasi numerik (vektor) menggunakan *embedding* model. Model *embedding* ini memetakan makna semantik setiap potongan teks ke dalam ruang vektor berdimensi tinggi sehingga memungkinkan sistem untuk menghitung kemiripan makna antar teks. Setelah *embedding* dihasilkan, sistem memasuki tahap *Knowledge Storage*, di mana vektor-vektor hasil *embedding* disimpan dalam Vektor *Database*. Basis data vektor ini berfungsi sebagai tempat penyimpanan pengetahuan yang dapat dicari secara semantik berdasarkan tingkat kesamaan makna antar dokumen. Ketika pengguna memberikan masukan berupa pertanyaan atau teks (*query*), sistem melakukan *Query Retrieval*, yaitu proses mencari potongan teks paling relevan di dalam basis data vektor dengan *semantic*

¹ <https://comfyai.app/article/llm-applications/retrieval-augmented-generation>

search [16]. Potongan teks yang memiliki nilai kesamaan tertinggi akan diambil sebagai konteks tambahan bagi model generatif.

Tahap terakhir adalah *Answer Generation*, di mana model bahasa besar (LLM) menggabungkan konteks hasil *retrieval* dengan pertanyaan pengguna untuk menghasilkan jawaban atau keluaran yang informatif dan relevan. Melalui pendekatan ini, sistem dapat memberikan hasil klasifikasi atau jawaban yang akurat, berbasis pada pengetahuan yang tersimpan di dalam dokumen sumber.

2.2.4. Large Language Model (LLM)

Large Language Models (LLMs) merupakan sistem berbasis kecerdasan buatan yang dirancang untuk memproses dan menghasilkan bahasa alami dalam bentuk teks. Secara fundamental, LLM dapat dipahami sebagai layanan yang menerima *string* sebagai masukan (*prompt*) dan menghasilkan *string* sebagai keluaran (*completion*) [17]. Dalam penggunaannya, LLM menerima input berupa teks dan meresponsnya dengan teks lain yang diprediksi sebagai kelanjutan yang paling masuk akal.

Namun, LLM tidak langsung memberikan respons yang bermakna sejak awal. Model ini perlu melalui proses pelatihan (*training*) dengan menggunakan kumpulan data dalam skala besar yang disebut *training set*, yang biasanya terdiri atas berbagai dokumen seperti buku, artikel, percakapan daring (misalnya dari Reddit), dan kode program (misalnya dari GitHub) [17]. Tujuan dari pelatihan ini adalah agar model mampu meniru pola bahasa dalam data latih tersebut, sehingga dapat memprediksi kelanjutan teks berdasarkan konteks yang diberikan. Dalam hal ini, LLM tidak hanya melakukan *memorization*, melainkan juga belajar menerapkan pola logis dan penalaran yang terdapat dalam data latih untuk menjawab berbagai *prompt* yang tidak pernah ia lihat sebelumnya.

Berbeda dari sistem pencarian (*search engine*) yang dapat mengakses dan menampilkan ulang informasi dari basis data, LLM dirancang untuk menghasilkan teks baru yang menyerupai hasil pelatihan tanpa menghafal secara eksplisit. Oleh karena itu, desain arsitektur internal LLM dan prosedur pelatihannya difokuskan untuk menghindari pengulangan langsung dan memprioritaskan kemampuan

generalisasi terhadap data yang belum pernah dilihat sebelumnya .

Sebagian besar aplikasi LLM saat ini menggunakan model umum (foundation models) yang telah dilatih sebelumnya oleh pihak lain, karena proses pelatihan dari awal membutuhkan sumber daya komputasi yang besar, keahlian tinggi, dan waktu yang lama. Meski demikian, pemahaman mengenai cara kerja dan tujuan pelatihan LLM sangat penting, terutama bila LLM akan digunakan secara terprogram atau dalam sistem produksi.

2.2.5. *Vector Embedding*

Embedding adalah teknik yang mengonversi kata-kata menjadi representasi numerik untuk menangkap hubungan semantik antar kata lebih baik. Pendekatan ini berhasil meningkatkan kinerja dalam berbagai aplikasi NLP. Penelitian lain menunjukkan bahwa modifikasi *embedding* dan penggunaan memori eksternal mampu meningkatkan panjang context window, seperti pada Long LLama, yang memperpanjang context window dari 2K menjadi 8K [18]

Embedding dalam LLM adalah representasi vektor berdimensi tinggi yang digunakan untuk memetakan kata, frasa, atau kalimat ke dalam bentuk numerik yang dapat diproses oleh model. Teknik embedding memungkinkan model bahasa memahami hubungan semantik antara kata-kata dengan memproyeksikannya ke dalam ruang vektor di mana kata-kata yang memiliki makna serupa terletak lebih dekat satu sama lain. Dalam *deep learning embedding* menjadi penting karena menyediakan representasi informasi yang dapat digunakan oleh model untuk mempelajari pola dan hubungan dalam data teks [19].

2.2.6. *Semantic Search*

Semantic Search atau pencarian semantik merupakan pendekatan modern dalam pencarian informasi yang berfokus pada pemahaman makna dari kata atau kalimat yang digunakan dalam query pencarian, bukan sekadar mencocokkan kata kunci secara literal. Konsep ini berupaya mengatasi keterbatasan dari sistem pencarian konvensional yang hanya menampilkan hasil berdasarkan kesamaan kata tanpa memahami konteks yang dimaksud oleh pengguna.

“Semantic search marks the transition from a ‘dumb’ search of single web pages that have a probabilistic value of containing the information we are looking for to an intelligent search that delivers real answers”[20].

Dalam *semantic search*, sistem tidak hanya melihat kata yang diketik, melainkan mencoba memahami niat dan hubungan semantik antar konsep. Misalnya, ketika seseorang mengetik "presiden pertama Indonesia", sistem tidak hanya mencari halaman dengan frasa tersebut, tetapi juga dapat mengaitkannya dengan entitas yang dikenal seperti "Soekarno", meskipun kata “Soekarno” tidak muncul dalam query.

2.2.6.1. Cosine Similarity

Cosine Similarity merupakan salah satu metode yang digunakan dalam *Semantic Search*. *Cosine Similarity* adalah metrik yang digunakan untuk mengukur kemiripan antara dua vektor dalam ruang berdimensi banyak dengan menghitung kosinus dari sudut di antara mereka. Metrik ini sering digunakan dalam analisis teks dan penambahan data untuk menentukan seberapa mirip dua dokumen atau teks satu sama lain, terlepas dari ukuran atau panjangnya [21].

Secara matematis, *Cosine Similarity* antara dua vektor A dan B dapat dihitung dengan rumus:

$$\cos \theta = \frac{A \cdot B}{\|A\| \|B\|}$$

Persamaan 2. 1

Dimana $A \cdot B$ adalah hasil perkalian titik dari vektor A dan B, serta $\|A\|$ dan $\|B\|$ adalah norma (panjang) dari vektor A dan B. Nilai *Cosine Similarity* berkisar antara -1 hingga 1, di mana 1 menunjukkan bahwa dua vektor memiliki arah yang sama (sangat mirip), 0 menunjukkan bahwa mereka ortogonal (tidak ada kemiripan), dan -1 menunjukkan bahwa mereka memiliki arah yang berlawanan.

Dalam konteks pengolahan bahasa alami, *Cosine Similarity* digunakan untuk mengukur kemiripan antara representasi vektor dari dokumen atau kalimat, yang memungkinkan sistem untuk menemukan dokumen yang relevan berdasarkan kemiripan konten, bukan hanya kemiripan kata.