OPTIMALISASI *KNOWLEDGE EXTRACTION* DAN *KNOWLEDGE RETRIEVAL* PADA RAG UNTUK *CHATBOT* KESEHATAN MENTAL

SKRIPSI

THE SUSTAINABLE DEVELOPMENT GOALS
Good Health and Well-Being
Industry, Innovation, and Infrastructure
Reduced Inequalities

Disusun oleh:

Ega Rizky Setiwan
21/479314/TK/52861

PROGRAM SARJANA PROGRAM STUDI TEKNOLOGI INFORMASI
DEPARTEMEN TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI
FAKULTAS TEKNIK UNIVERSITAS GADJAH MADA
YOGYAKARTA
2025

HALAMAN PENGESAHAN

OPTIMALISASI *KNOWLEDGE EXTRACTION* DAN *KNOWLEDGE RETRIEVAL* PADA RAG UNTUK *CHATBOT* KESEHATAN MENTAL

SKRIPSI

Diajukan Sebagai Salah Satu Syarat untuk Memperoleh
Gelar Sarjana Teknik
pada Departemen Teknik Elektro dan Teknologi Informasi
Fakultas Teknik
Universitas Gadjah Mada

Disusun oleh:

Ega Rizky Setiwan
21/479314/TK/52861

Telah disetujui dan disahkan

Pada tanggal . . . . . .

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Bimo Sunarfri Hantono, S.T., M.Eng.	**Dr. Guntur Dharma Putra, S.T., M.Sc.**
197701312002121003	**111199104201802102**

PERNYATAAN BEBAS PLAGIASI

Saya yang bertanda tangan di bawah ini :

Nama	: Ega Rizky Setiawan
NIM	: 21/479314/TK/52861
Tahun terdaftar	: 2021
Program	: Sarjana
Program Studi	: Teknologi Informasi
Fakultas	: Teknik Universitas Gadjah Mada

Menyatakan bahwa dalam dokumen ilmiah Skripsi ini tidak terdapat bagian dari karya ilmiah lain yang telah diajukan untuk memperoleh gelar akademik di suatu lembaga Pendidikan Tinggi, dan juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang/lembaga lain, kecuali yang secara tertulis disitasi dalam dokumen ini dan disebutkan sumbernya secara lengkap dalam daftar pustaka.

Dengan demikian saya menyatakan bahwa dokumen ilmiah ini bebas dari unsur-unsur plagiasi dan apabila dokumen ilmiah Skripsi ini di kemudian hari terbukti merupakan plagiasi dari hasil karya penulis lain dan/atau dengan sengaja mengajukan karya atau pendapat yang merupakan hasil karya penulis lain, maka penulis bersedia menerima sanksi akademik dan/atau sanksi hukum yang berlaku.

Yogyakarta, 25 Agustus 2025

Materai Rp10.000

(Tanda tangan)

Ega Rizky Setiawan
21/479314/TK/52861

HALAMAN PERSEMBAHAN

Tugas akhir ini kupersembahkan kepada kedua orang tuaku, saudara, dan keluarga. Kupersembahkan pula kepada teman-teman semua, serta untuk bangsa, negara, dan agamaku.

KATA PENGANTAR

Puji syukur ke hadirat Allah SWT yang telah melimpahkan rahmat dan barokah-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "Optimalisasi *Knowledge Extraction* dan *Knowledge Retrieval* pada RAG untuk *Chatbot* Kesehatan Mental". Laporan skripsi ini disusun untuk memenuhi salah satu syarat dalam memperoleh gelar Sarjana Teknik (S.T.) pada Program Studi S1 Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada Yogyakarta.

Dalam melakukan penelitian dan penyusunan laporan skripsi ini penulis telah mendapatkan banyak dukungan dan bantuan dari berbagai pihak. Penulis mengucapkan terima kasih yang tak terhingga kepada:

1. Prof. Ir. Hanung Adi Nugroho, S.T., M.E., Ph.D., IPM., SMIEEE. selaku Ketua Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada
2. Ir. Lesnanto Multa Putranto, S.T., M.Eng., Ph.D., IPM., SMIEEE. selaku Sekretaris Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada
3. Dr. Bimo Sunarfri Hantono, S.T., M.Eng. selaku dosen pembimbing pertama yang telah dengan penuh kesabaran dan ketulusan memberikan arahan, bimbingan, serta dukungan dalam menyelesaikan tugas akhir ini.
4. Dr. Guntur Dharma Putra, S.T., M.Sc. selaku dosen pembimbing kedua yang telah dengan penuh kesabaran dan ketulusan memberikan arahan, bimbingan, serta dukungan dalam menyelesaikan tugas akhir ini.
5. Orang Tua dan Adik yang selalu memberikan arahan selama belajar dan menyelesaikan tugas akhir ini.
6. Teman-teman penulis dan berbagai pihak yang tidak dapat disebutkan satu per satu yang telah memberikan dukungan, semangat, serta bantuan dalam menyelesaikan tugas akhir ini.

Penulis menyadari sepenuhnya bahwa laporan skripsi ini masih jauh dari sempurna, untuk itu semua jenis saran, kritik dan masukan yang bersifat membangun sangat penulis harapkan. Akhir kata, semoga tulisan ini dapat memberikan manfaat dan memberikan wawasan tambahan bagi para pembaca dan khususnya bagi penulis sendiri.

Yogyakarta, 25 Agustus 2025

Ega Rizky Setiawan

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SINGKATAN

AI	=	Artificial Intelligence
CPMH	=	Center for Public Mental Health
DL	=	Deep Learning
DPR	=	Dense Passage Retriever
GPQA	=	Graduate-Level Google-Proof Question and Answer
HPU	=	Health Promoting University
HTML	=	Hyper Text Markup Language
JSON	=	JavaScript Object Notation
KE	=	Knowledge Extraction
KG	=	Knowledge Graph
KR	=	Knowledge Retrieval
LAMBADA	=	Language Modeling Broadened to Account for Discourse Aspects
LLM	=	Large Language Model
ML	=	Machine Learning
MIPS	=	Maximum Inner Product Search
MRR	=	Mean Reciprocal Rank
NLP	=	Natural Language Processing
NLG	=	Natural Language Generation
NLU	=	Natural Language Understanding
PDF	=	Portable Document Format
RAG	=	Retrieval-Augmented Generation
RAGAS	=	Retrieval-Augmented Generation Assessment
REST	=	Representational State Transfer
STS	=	Semantic Text Similarity
UGM	=	Universitas Gadjah Mada
URL	=	Uniform Resource Locator

INTISARI

Pemanfaatan *chatbot* sebagai sarana pendukung kesehatan mental yang aksesibel dihadapkan pada tantangan krusial terkait akurasi informasi. Arsitektur *chatbot* yang umum digunakan, seperti pendekatan *Retrieval-Augmented Generation* (RAG) berbasis potongan dokumen (Naive RAG) memiliki risiko dalam menghasilkan misinformasi dan halusinasi akibat kurangnya pemahaman kontekstual secara komprehensif. Penelitian ini bertujuan untuk mengatasi masalah tersebut dengan merancang, mengimplementasikan, dan mengevaluasi sebuah arsitektur RAG yang diperkaya dengan basis pengetahuan terstruktur berupa *Knowledge Graph* (KG). Metodologi yang diusulkan berfokus pada tiga pilar utama yaitu perancangan arsitektur RAG yang mengintegrasikan KG untuk menjamin konsistensi dan akurasi, optimalisasi proses *Knowledge Extraction* (KE) menggunakan LLM untuk membangun KG dari literatur kesehatan mental, dan evaluasi berbagai metode *Knowledge Retrieval* (KR) untuk menemukan informasi yang paling relevan. Hasil pengujian menunjukkan bahwa arsitektur RAG-KG secara signifikan mengungguli pendekatan Naive RAG berbasis potongan dokumen pada hampir semua metrik uji yang ditunjukkan dengan peningkatan *correctness* sebesar 8% *faithfulness* sebesar 4% dan *relevancy* sebesar 17%. Penelitian ini mengonfirmasi bahwa penggunaan *Knowledge Graph* yang dibangun dan diakses melalui metode yang teroptimalkan merupakan pendekatan yang kuat untuk meningkatkan keandalan dan keamanan *chatbot*. Hasilnya adalah sebuah sistem yang mampu menyajikan informasi kesehatan mental yang lebih faktual, relevan, dan berbasis bukti, sekaligus meminimalkan risiko halusinasi.

Kata kunci : RAG, *knowledge graph*, *knowledge extraction*, *knowledge retrieval*, *chatbot*, kesehatan mental, LLM.

ABSTRACT

The use of chatbots as an accessible facilities for mental health support faces a crucial challenge regarding information accuracy. Common chatbot architectures, such as the conventional Retrieval-Augmented Generation (RAG) approach (Naive RAG) which relies on document chunks, carry a risk of generating misinformation and hallucinations due to a lack of comprehensive contextual understanding. This research aims to address this issue by designing, implementing, and evaluating a RAG architecture enriched with a structured knowledge base in the form of a Knowledge Graph (KG). The proposed methodology focuses on three main pillars that is the design of a RAG architecture that integrates a KG to ensure consistency and accuracy, the optimization of the Knowledge Extraction (KE) process using an LLM to build the KG from mental health literature, and the evaluation of various Knowledge Retrieval (KR) methods to find the most relevant information. Experimental results show that the RAG-KG architecture significantly outperforms the Naive RAG approach, demonstrated by an 8% increase in correctness, a 4% increase in faithfulness, and a 17% increase in relevancy. This study confirms that the use of a Knowledge Graph, built and accessed through an optimized method, is a robust approach to enhance the reliability and security of chatbots. The result is a system capable of delivering more factual, relevant, and evidence-based mental health information, while minimizing the risk of hallucinations.

Keywords : RAG, knowledge graph, knowledge extraction, knowledge retrieval, chatbot, mental health, LLM.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kesehatan mental merupakan komponen fundamental bagi kesejahteraan individu dan masyarakat secara keseluruhan. Isu kesehatan mental global menunjukkan tren yang mengkhawatirkan, ditandai dengan peningkatan prevalensi berbagai gangguan mental seperti depresi, kecemasan, dan stres di berbagai negara. Sebagai contoh, penelitian yang dilakukan oleh Institute for Health Metrics and Evaluation (IHME) memperkirakan bahwa sekitar 970 juta orang, atau 12% dari populasi global, mengalami gangguan mental pada 2019 [3]. Di Indonesia, situasinya juga tidak kalah mengkhawatirkan. Khususnya pada populasi remaja, hasil temuan penelitian Indonesia - National Adolescence Mental Health Survey (I-NAMHS) yang dirilis pada tahun 2022 menyajikan gambaran yang lebih detail. Satu dari tiga remaja Indonesia (34,9%), setara dengan 15,5 juta jiwa, memiliki setidaknya satu masalah kesehatan mental dalam 12 bulan terakhir. Lebih lanjut, satu dari dua puluh remaja (5,5%), atau sekitar 2,45 juta jiwa, terdiagnosis mengalami gangguan mental dalam periode yang sama, dengan gangguan kecemasan sebagai jenis yang paling umum ditemui [4]. Dampak dari gangguan kesehatan mental ini sangat luas, tidak hanya memengaruhi kualitas hidup, produktivitas, dan interaksi sosial individu, tetapi juga memberikan beban signifikan pada sistem layanan kesehatan dan perekonomian. Oleh karena itu, investasi dalam upaya peningkatan kesehatan mental sangat krusial, karena berpotensi meningkatkan kualitas kesehatan masyarakat secara umum, yang pada gilirannya akan mendukung peningkatan fungsi sosial, produktivitas, dan pertumbuhan ekonomi.

Sayangnya, akses terhadap layanan kesehatan mental yang berkualitas masih menjadi tantangan besar. Faktor-faktor seperti stigma sosial yang kuat, keterbatasan jumlah tenaga profesional kesehatan mental yang terdistribusi tidak merata, serta biaya layanan yang sering kali tidak terjangkau oleh sebagian besar masyarakat, menciptakan kesenjangan besar antara kebutuhan dan ketersediaan layanan. Sebanyak 71% dari penderita psikosis tidak mendapatkan penanganan kesehatan mental [5]. Data I-NAMHS 2022 semakin memperjelas jurang ini: hanya 2,6% dari remaja dengan masalah kesehatan mental yang pernah mengakses layanan yang menyediakan dukungan atau konseling untuk masalah emosi dan perilaku dalam 12 bulan terakhir. Lebih lanjut, secara keseluruhan, hanya satu dari lima puluh remaja (2%) yang pernah menggunakan layanan dalam 12 bulan terakhir, dan dua pertiga dari mereka (66,5%) hanya pernah menggunakan layanan sebanyak satu kali. Ironisnya, ketika ditanyakan mengenai penyedia layanan yang paling banyak diakses oleh mereka yang mencari bantuan, hampir dua perlima (38,2%)

pengasuh utama menjawab petugas sekolah, yang mengindikasikan peran penting namun mungkin belum optimal dari sistem pendidikan dalam deteksi dini dan rujukan. Fenomena ini juga diperkuat oleh temuan bahwa hanya 4,3% dari pengasuh utama yang menyatakan bahwa remaja mereka membutuhkan bantuan untuk masalah emosi dan perilaku, meskipun data menunjukkan 34,9% remaja mengalami masalah kesehatan mental. Dari pengasuh utama yang menyadari kebutuhan bantuan, lebih dari dua perlima (43,8%) melaporkan tidak mencari bantuan karena lebih memilih untuk menangani sendiri masalah remaja tersebut atau dengan dukungan dari keluarga dan teman-teman, yang mungkin juga mencerminkan stigma atau ketidakpercayaan terhadap layanan formal. Kondisi ini mendesak perlunya inovasi untuk memperluas jangkauan dan meningkatkan kualitas dukungan kesehatan mental. Pandemi COVID-19 juga turut memperburuk situasi, di mana 4,6% remaja melaporkan sering merasa lebih cemas, lebih depresi, lebih kesepian, atau lebih sulit berkonsentrasi dari biasanya selama periode tersebut [4].

Teknologi kecerdasan buatan atau *Artificial Intelligence* (AI) telah berkembang secara masif selama beberapa dekade terakhir dan mulai menggeser perubahan aktivitas manusia. AI mengubah proses-proses tradisional yang kaku dan membutuhkan campur tangan manusia secara intens menjadi proses modern yang luwes dan minim campur tangan manusia. Teknologi ini sudah banyak mengintervensi kegiatan sehari-hari di berbagai sektor industri seperti *online shopping*, *online banking*, dan *mental health care* [6]. Salah satu aplikasi teknologi AI dalam kesehatan mental yang paling banyak dikembangkan adalah *chatbot* atau agen percakapan. *Chatbot* kesehatan mental menawarkan potensi solusi yang menjanjikan dengan memberikan dukungan emosional awal dan informasi psikoedukasi yang mudah diakses kapan saja dan di mana saja melalui perangkat digital. Keunggulan tersebut diharapkan dapat membantu mengatasi beberapa hambatan penderita mental disorder dalam mencari bantuan, terutama terkait dengan stigma dan aksesibilitas geografis maupun finansial [7]. Salah satu penerapan *chatbot* pada bidang kesehatan mental adalah Lintang *Chatbot* yang dikembangkan oleh Health Promoting University (HPU) Universitas Gadjah Mada (UGM). *Chatbot* ini menggunakan pendekatan berbasis aturan (*rule-based*) yang dirancang untuk memberikan pertolongan pertama psikologis dan informasi kesehatan mental. Pada era modern ini, *chatbot* yang populer dibagun dengan basis *Large Language Model* (LLM), seperti ChatGPT, Gemini, Claude, dll. LLM merupakan model kecerdasan buatan yang telah dilatih dengan data dalam jumlah sangat besar, memungkinkannya untuk menghasilkan respons yang sangat luwes dan alami.

Meskipun menawarkan potensi yang menjanjikan, *chatbot* kesehatan mental yang ada saat ini, utamanya yang berbasis pada aturan atau model generatif sederhana, masih memiliki keterbatasan. Sering kali respons yang diberikan terasa kaku, repetitif, dan gagal memahami nuansa dalam ekspresi emosi atau pertanyaan pengguna. Tantangan

utama terletak pada kemampuan *chatbot* dalam mengakses, memproses, dan menyajikan informasi yang tidak hanya akurat dan relevan, tetapi juga kontekstual dan empatik. Pengguna sering kali membutuhkan informasi yang sangat spesifik mengenai kondisi mereka, opsi perawatan, atau fasilitas yang tersedia. Informasi ini sering kali tidak tersedia oleh *chatbot* pada umumnya karena keterbatasan terhadap data yang selalu berubah dan berkembang setiap waktu. Keterbatasan akses tersebut dapat membuat model generatif mengalami "halusinasi", yaitu model menciptakan informasi yang terdengar meyakinkan namun tidak faktual atau tidak bersumber dari basis pengetahuan yang valid. Lebih lanjut, memastikan bahwa basis pengetahuan *chatbot* selalu terbarui, berbasis bukti ilmiah (*evidence-based*), dan bebas dari misinformasi merupakan hal yang krusial, mengingat sensitivitas isu kesehatan mental.

Untuk menjawab tantangan tersebut, salah satu pendekatan yang telah dikembangkan adalah *Retrieval-Augmented Generation* (RAG) [8]. RAG menggabungkan kemampuan menghasilkan respons yang luwes dan alami dari LLM ini dengan informasi relevan yang diambil dari basis pengetahuan. Pendekatan tersebut diharapkan dapat membuat *chatbot* memberikan jawaban yang lebih faktual, mengurangi halusinasi, dan menyajikan informasi yang lebih spesifik dan terkini karena memiliki konteks yang relevan dengan pertanyaan. Pada umumnya, arsitektur RAG merepresentasikan basis pengetahuannya dalam bentuk kumpulan dokumen [8], sejalan dengan cara manusia secara alami merepresentasikan pengetahuan melalui buku, artikel, atau jurnal ilmiah. Kumpulan dokumen yang relevan diambil oleh sistem sebagai konteks tambahan dalam menjawab pertanyaan.

Meskipun efektif dalam mengurangi halusinasi, arsitektur RAG konvensional ini memiliki keterbatasan fundamental yang berasal dari representasi datanya yang datar (*flat*). Struktur ini menghadapi kesulitan saat sistem mencoba mengambil informasi berdasarkan hubungan antar entitas. Sistem ini seringkali kurang memiliki *contextual awareness* yang dibutuhkan untuk menjaga koherensi di berbagai entitas dan interelasinya, sehingga respons yang dihasilkan mungkin tidak sepenuhnya menjawab pertanyaan pengguna. Sebagai contoh, dalam domain kesehatan mental, sebuah sistem mungkin dapat mengambil potongan teks tentang "Hiperaktivitas" dan potongan lain tentang "Gangguan Mental Emosional", tetapi ia kesulitan untuk secara konsisten memanfaatkan hubungan eksplisit bahwa yang satu adalah gejala untuk yang lain karena informasi tersebut berada dalam dokumen yang terpisah. Akibatnya, jawaban yang dihasilkan bisa jadi terfragmentasi dan gagal menyajikan jawaban yang koheren.

Untuk mengatasi limitasi tersebut, penelitian terbaru mulai mengusulkan pendekatan graph-based RAG, yang mengorganisir basis pengetahuan ke dalam sebuah struktur graf. Dalam model ini, informasi direpresentasikan sebagai *node* yang melambangkan entitas (misalnya, gangguan, gejala, obat) dan *edge* yang menandai hubungan semantik

di antara mereka [9]. Sebagai contoh, "Depresi Mayor" memiliki gejala "Kehilangan Minat", "Terapi Kognitif Perilaku" digunakan untuk mengobati "Gangguan Panik". Representasi terstruktur ini, yang dikenal sebagai *Knowledge Graph* (KG), memungkinkan pencarian informasi yang lebih presisi, penarikan kesimpulan (*reasoning*), dan pemahaman konteks yang lebih luas dan mendalam [9]. Adopsi basis pengetahuan berupa KG ke dalam RAG berpotensi meningkatkan kemampuan *chatbot* kesehatan mental untuk memberikan penjelasan yang kaya, jawaban yang akurat atas pertanyaan-pertanyaan spesifik, dan bahkan membantu pengguna memahami hubungan antara berbagai aspek kondisi mental mereka.

Namun, efektivitas KG dalam arsitektur RAG sangat ditentukan oleh dua proses fundamental, yaitu *Knowledge Extraction* (KE) dan *Knowledge Retrieval* (KR). KE adalah proses mengidentifikasi dan mengekstrak informasi terstruktur (entitas, relasi, atribut) dari berbagai sumber data (misalnya, literatur medis, jurnal psikologi, pedoman klinis, forum diskusi terkurasi) untuk membangun atau memperkaya KG [10]. Dalam domain kesehatan mental, proses ini dihadapkan pada tantangan seperti variasi terminologi, sinonim, polisemi, dan kebutuhan krusial akan akurasi serta validitas informasi. Ekstraksi yang tidak optimal akan menghasilkan KG yang tidak lengkap, tidak akurat, atau bias, yang pada akhirnya akan menurunkan kualitas informasi yang dapat diakses oleh sistem RAG.

Selanjutnya, *Knowledge Retrieval*, yaitu proses untuk menemukan informasi yang paling relevan dengan kueri pengguna untuk diumpankan ke model generatif dalam RAG, juga merupakan aspek krusial. Ketika pengguna berinteraksi dengan *chatbot* menggunakan bahasa alami, *retriever* harus mampu menyediakan informasi dari basis data yang dimiliki agar cukup untuk menjawab pertanyaan. Proses ini harus mampu menangani ambiguitas bahasa, memahami intensi pengguna secara akurat, dan mengambil informasi yang relevan tanpa membebani model generatif dengan informasi yang berlebihan atau tidak perlu. Metode *retrieval* yang suboptimal akan menghasilkan respons yang kurang relevan, tidak lengkap, atau bahkan menyesatkan [2].

Meskipun potensi RAG berbasis KG untuk *chatbot* kesehatan mental sangat besar, penelitian yang secara spesifik berfokus pada optimalisasi proses *Knowledge Extraction* dan *Knowledge Retrieval* dalam konteks ini masih relatif terbatas. Kebanyakan penelitian lebih terfokus pada pembangunan KG secara umum atau aplikasi RAG dengan sumber data teks bebas. Padahal, optimalisasi kedua proses ini, yang disesuaikan dengan karakteristik unik dan tuntutan domain kesehatan mental seperti kebutuhan akan akurasi tinggi, empati, dan penyampaian informasi yang mudah dicerna, sangat esensial untuk memaksimalkan efektivitas dan keamanan *chatbot*. Oleh karena itu, penelitian ini bertujuan untuk menginvestigasi dan mengembangkan metode optimalisasi untuk kedua proses tersebut, guna membangun chatbot kesehatan mental yang lebih akurat, andal, dan mampu

memberikan pemahaman kontekstual yang mendalam.

1.2 Rumusan Masalah

Pemanfaatan chatbot untuk layanan kesehatan mental yang aksesibel dihadapkan pada tantangan fundamental, yaitu memastikan respons yang akurat, andal, dan berbasis bukti untuk mencegah risiko misinformasi yang berbahaya bagi pengguna. Meskipun arsitektur canggih seperti RAG yang diperkaya KG menawarkan solusi menjanjikan, implementasinya masih dihadapkan pada kendala teknis, khususnya pada komponen *Knowledge Extraction* dan *Knowledge Retrieval*. Oleh karena itu, penelitian ini difokuskan untuk menjawab permasalahan sebagai berikut.

1. Belum adanya rancangan arsitektur *chatbot* layanan kesehatan mental yang teruji untuk menjamin konsistensi dan akurasi respons berbasis bukti, sehingga berisiko menimbulkan misinformasi yang dapat membahayakan pengguna dalam konteks kesehatan mental yang sensitif.
2. Adanya tantangan dalam merancang proses KE yang mampu memetakan terminologi dan hubungan kompleks dari literatur kesehatan mental ke dalam struktur KG yang koheren. Kegagalan dalam memodelkan hubungan ini secara akurat akan menghasilkan KG yang tidak mampu mendukung proses *retrieval* secara mendalam, menyebabkan jawaban menjadi dangkal atau tidak lengkap.
3. Ketidakmampuan proses KR untuk secara konsisten menemukan potongan informasi yang paling relevan dari KG sebagai respons terhadap pertanyaan pengguna, berisiko membuat arsitektur RAG mengambil basis bukti yang salah atau tidak lengkap, sehingga respons yang dihasilkan menjadi tidak akurat dalam menjawab pertanyaan spesifik pengguna.

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dipaparkan, penelitian ini memiliki beberapa tujuan spesifik yang dirancang untuk menjawab tantangan tersebut. Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang dan mengimplementasikan sebuah arsitektur *chatbot* kesehatan mental yang mengintegrasikan KG dan RAG untuk menghasilkan respons berbasis bukti yang konsisten dan akurat.
2. Mengembangkan metode *Knowledge Extraction* menggunakan LLM yang mampu membangun KG dengan struktur relasi yang efektif untuk mendukung proses *Knowledge Retrieval*.
3. Merancang dan menerapkan sebuah metode KR yang mampu secara akurat dan efisien menemukan informasi paling relevan dari KG sebagai respons terhadap perta-

nyaan spesifik pengguna, guna meningkatkan akurasi sistem RAG.

1.4 Batasan Penelitian

1. **Objek Penelitian**: Penelitian ini akan berfokus pada metode RAG berbasis *Knowledge Graph* yang mencakup Metode optimalisasi *Knowledge Extraction* dari sumber data tekstual kesehatan mental, struktur, kualitas, dan pemanfaatan KG kesehatan mental sebagai basis bukti, algoritma dan strategi optimalisasi *Knowledge Retrieval* dari KG untuk mendukung sistem RAG, dan kualitas dan efektivitas respons berbasis bukti yang dihasilkan oleh prototipe *chatbot* kesehatan mental yang dikembangkan.
2. **Metode Penelitian**: Penelitian ini akan menggunakan pendekatan metode eksperimen dan kuantitatif. Pada tahap pengembangan akan menggunakan pendekatan desain dan rekayasa sistem (*design and engineering*) untuk merancang, membangun, dan mengimplementasikan metode KE dan KR pada KG. Ini melibatkan studi literatur, perancangan algoritma, implementasi perangkat lunak, dan pengujian iteratif. Selanjutnya evaluasi performa komponen sistem dilakukan secara kuantitatif. Performa komponen *retriever* dan jawaban akhir akan dibandingkan dengan *golden standard*.
3. **Waktu dan Tempat Penelitian**: Penelitian ini akan dilakukan selama periode enam bulan, dimulai pada bulan Februari 2025 hingga Juli 2025. Tempat penelitian adalah di Gedung DTETI FT UGM, dengan menggunakan VS Code untuk pengembangan dan pengujian model.
4. **Populasi dan Sampel**: Populasi dalam penelitian ini adalah seluruh korpus data teks yang relevan dengan domain kesehatan mental, seperti artikel ilmiah dan buku teks. Sampel data dipilih dari populasi tersebut yang berupa buku dan artikel ilmiah yang tersedia pada *website* UGM, CPMH, HPU UGM, dan Kementerian Kesehatan.
5. **Variabel**: Penelitian ini menggunakan variabel bebas berupa metode KE dan KR yang diterapkan. Variabel terikat adalah performa *chatbot* yang berupa kualitas respons yang dinilai berdasarkan metrik tertentu.
6. **Hipotesis**: Hipotesis yang diusulkan dalam penelitian ini adalah *chatbot* kesehatan mental yang menggunakan RAG berbasis KG dengan optimalisasi pada KE dan KR menunjukkan peningkatan performa dibandingkan dengan *chatbot* yang menggunakan pendekatan RAG standar atau tanpa KG.
7. **Keterbatasan Penelitian**: Penelitian ini memiliki keterbatasan pada *dataset* yang digunakan sebagai basis pengetahuan dalam sistem RAG. Akurasi dan kelengkapan data bergantung pada *dataset* yang mungkin belum mencakup semua aspek permasalahan kesehatan mental. Penelitian ini tidak mengevaluasi dampak klinis *chat-*

bot pada kondisi psikologis pengguna melainkan hanya mengevaluasi hasil respons *chatbot* dengan metrik yang tersedia.

1.5 Manfaat Penelitian

Penelitian ini diharapkan memberikan kontribusi pada pengembangan arsitektur *chatbot* yang lebih andal dan terpercaya, khususnya dalam domain yang memerlukan informasi berbasis bukti seperti layanan kesehatan mental. Selain itu, penelitian ini juga diharapkan memberikan wawasan mengenai efektivitas arsitektur *chatbot* menggunakan RAG berbasis *Knowledge Graph* dalam memberikan respons yang akurat. Dari sisi praktis, *chatbot* ini dapat diimplementasikan dan memberikan dampak positif dengan membantu pengguna dalam membuat keputusan terkait dengan kesejahteraan mental mereka, dengan tetap menekankan pentingnya konsultasi profesional untuk diagnosis dan perawatan.

1.6 Sistematika Penulisan

Sistematika penulisan berisi hal yang akan dibahas dalam penelitian ini yang akan disusun menjadi 5 bab, sebagai berikut,

1. **Bab I Pendahuluan** menjelaskan hal yang melatarbelakangi adanya penelitian ini, rumusan masalah yang akan coba dicari solusinya, tujuan yang ingin dicapai, serta manfaat bagi pihak-pihak terkait.
2. **Bab II Tinjauan Pustaka dan Dasar Teori** berisi ulasan literatur penelitian yang sudah pernah dilakukan sebelumnya mengenai topik-topik yang relevan dengan pengembangan chatbot. Bab ini akan menguraikan beberapa metode yang pernah dilakukan untuk meningkatkan performa *chatbot* pada aplikasi tertentu, khususnya pada bidang layanan kesehatan mental. Setiap metode tersebut akan diulas potensinya untuk pengembangan lebih lanjut, peningkatan yang telah dicapai, dan tantangan yang mungkin dihadapi. Selain itu, bab ini juga akan disertai dengan penjelasan mengenai teori-teori yang berkaitan dengan pengembangan *chatbot*, arsitektur RAG, dan penggunaan *Knowledge Graph*.
3. **Bab III Metode Penelitian** membahas langkah-langkah detail yang akan ditempuh dalam penelitian, seperti mulai dari identifikasi masalah, studi literatur, proses pengumpulan data, perancangan dan pengembangan model, hingga metode evaluasi dan analisis hasil. Bab ini juga akan menjelaskan metode penelitian eksperimen dan kuantitatif yang akan dilakukan dalam penelitian ini.
4. **Bab IV Hasil dan Pembahasan** menyajikan hasil dari implementasi dan pengujian model *chatbot* yang telah dikembangkan. Pada bab ini akan membahas mengenai kualitas respons dari *chatbot* yang mengimplementasikan arsitektur RAG berbasis

Knowledge Graph dan dibandingkan dengan yang tidak menggunakan arsitektur tersebut. Hasil temuan tersebut akan dianalisis secara mendalam dan dibandingkan dengan hipotesis serta penelitian sebelumnya.

5. **Bab V Kesimpulan dan Saran** berisi kesimpulan yang ditarik dari keseluruhan hasil penelitian dan pembahasan. Bagian ini akan memberikan gambaran mengenai kontribusi penelitian ini dalam pengembangan teknologi *chatbot* pada layanan kesehatan mental. Selain itu, akan disampaikan pula saran-saran untuk pengembangan lebih lanjut dari sistem yang dibangun atau untuk penelitian terkait di masa mendatang

BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

2.1.1 *Chatbot* dengan Sistem Berbasis Aturan (*Rule-Based System*)

Sistem berbasis aturan (*rule-based system*) merupakan salah satu pendekatan yang paling awal dalam pengembangan *chatbot*. Dalam sistem ini, alur percakapan, pertanyaan yang dapat dikenali, serta respons yang diberikan telah ditentukan sebelumnya melalui serangkaian aturan eksplisit. Aturan-aturan tersebut umumnya dirancang secara manual oleh ahli di bidang tertentu. Sebagai contoh, dalam konteks kesehatan mental, perancang aturan bisa berupa psikiater, psikolog, atau peneliti di bidang kesehatan. Implementasi dari aturan-aturan tersebut dapat berbentuk struktur kondisional seperti *if-then-else* yang digunakan untuk mengatur jalannya percakapan. Karena aturan-aturan ini tertulis secara eksplisit, pengembang memiliki kendali penuh terhadap arah percakapan dan jawaban yang diberikan oleh *chatbot*. Hal ini memungkinkan *chatbot* merespons pertanyaan pengguna secara konsisten dan sesuai dengan skenario yang telah ditentukan.

Salah satu pionir *chatbot* berbasis aturan adalah ELIZA, yang dikembangkan oleh Joseph Weizenbaum di Massachusetts Institute of Technology (MIT) pada tahun 1966. ELIZA dirancang untuk menyimulasikan percakapan antara pasien dan psikoterapis aliran Rogerian, dengan memanfaatkan teknik *pattern matching* (pencocokan pola) dan *substitution* (penggantian kata). Sistem ini mencocokkan masukan pengguna dengan pola-pola tertentu yang telah didefinisikan dalam skrip. Pola-pola tersebut biasanya mengandung *wildcard* seperti simbol * (*asterisk*), yang memungkinkan sistem menangkap frasa dari masukan pengguna dan menyusunnya kembali dalam bentuk respons yang sesuai [11].

Di Indonesia, salah satu contoh implementasi *chatbot* berbasis aturan adalah Lintang, sebuah *chatbot* yang dikembangkan oleh HPU UGM. *Chatbot* ini dirancang untuk melakukan skrining awal masalah kesehatan mental dan memberikan pertolongan pertama psikologis. Lintang dilengkapi dengan berbagai fitur utama seperti psikoedukasi yang menyediakan informasi penting mengenai kesehatan mental, termasuk berbagai gangguan, gejala, dan strategi bantuan diri. Tips kesehatan mental yang memberikan saran untuk merawat dan meningkatkan kesehatan mental, seperti teknik relaksasi, manajemen stres, dan promosi kesejahteraan emosional. Fitur swaperiksa yang memungkinkan pengguna melakukan evaluasi awal kondisi kesehatan mental dengan menjawab sejumlah pertanyaan. Terakhir, direktori layanan yang menyajikan informasi mengenai sumber daya dan layanan kesehatan mental yang tersedia di lingkungan fakultas maupun universitas [12].

Sistem berbasis aturan menawarkan keunggulan dari segi kontrol penuh akan alur

percakapan dan respons yang dihasilkan. Hal tersebut menjadikannya cocok diimplementasikan pada aplikasi yang membutuhkan struktur dialog ketat dan dapat diprediksi seperti pada ELIZA dan Lintang. Namun, pendekatan ini memiliki tantangan pada skalabilitas dan fleksibilitas. Sistem akan menjadi sangat kompleks dan sulit untuk dipelihara seiring bertambahnya variasi masukan pengguna karena semua aturan harus di bangun secara manual. Selain itu, *chatbot* cenderung tidak mampu untuk memahami dan menangani percakapan yang sifatnya terbuka atau tidak terduga.

2.1.2 *Chatbot* dengan *Fine-Tuning Large Language Model*

Chatbot mengalami perkembangan yang signifikan seiring dengan kemajuan teknik *Natural Language Processing* (NLP), khususnya dengan adanya *Large Language Model* (LLM) seperti GPT, Gemini, Qwen, Llama, dll. LLM memungkinkan terjadinya interaksi percakapan yang luwes antara manusia dan mesin. *Language model* ini dilatih menggunakan sumber data yang masif, sehingga mampu memahami dan menghasilkan teks dengan konteks yang luas dan alami. Namun, LLM mengalami kesulitan saat diimplementasikan pada kasus yang spesifik, seperti menjadi *chatbot* kesehatan mental untuk suatu instansi. Jawaban yang diberikan LLM sering kali bermasalah, seperti terlalu umum, berhalusinasi, dan tidak faktual. Untuk itu, banyak dikembangkan metode seperti *fine-tuning* untuk menutupi celah tersebut.

Fine-tuning banyak dilakukan untuk melatih suatu *language model* sehingga dapat sangat mahir pada domain tertentu. Sebuah penelitian oleh Yu pada tahun 2024 mencoba mengeksplorasi metode untuk meningkatkan performa *chatbot* untuk dukungan kesehatan mental. Penelitian tersebut menunjukkan adanya peningkatan performa umpan balik *chatbot* setelah dilakukan *fine-tuning*. Hasil evaluasi menunjukkan penggunaan *fine-tuning* pada LLM DialoGPT (sekitar 1,5 B parameter) memiliki kualitas percakapan yang lebih relevan dan akurat, dengan skor BLEU mencapai 0,32 yang lebih tinggi dibandingkan GPT 3 (sekitar 175 B parameter) tanpa *fine-tuning* yang hanya mencapai 0,13 [13]. Dari hasil tersebut terlihat potensi besar *fine-tuning* dalam meningkatkan performa LLM untuk kasus tertentu.

Fine-tuning terbukti secara signifikan meningkatkan performa *language model* dalam menghasilkan umpan balik yang lebih relevan dan kontekstual, sebuah kapabilitas yang krusial khususnya dalam aplikasi layanan kesehatan mental. Tingkat keberhasilan peningkatan ini sangat ditentukan oleh kualitas dan relevansi data yang digunakan selama proses *fine-tuning*. Model yang dihasilkan secara inheren akan mencerminkan karakteristik data pelatihannya. Meskipun demikian, implementasi *fine-tuning* menghadapi tantangan, terutama ketika berhadapan dengan basis pengetahuan yang sering berubah atau dinamis. Kebutuhan untuk melakukan *fine-tuning* ulang secara berkala demi menjaga model tetap *up-to-date* memerlukan alokasi sumber daya komputasi yang besar, baik

dari segi waktu, maupun daya pemrosesan.

2.1.3 Penggunaan *Prompt Engineering* pada *Large Language Model* (LLM)

Large Language Model, yang dilatih pada korpus data tekstual masif, menunjukkan kinerja yang luar biasa dalam berbagai tugas pemahaman dan *text generation*, bahkan sering melampaui manusia pada tugas pengetahuan umum. LLM beroperasi dengan mengkodekan input teks ke dalam suatu ruang vektor berdimensi tinggi yang merepresentasikan hubungan semantik antar kata dan frasa. Representasi vektor tersebut yang kemudian menjadi dasar bagi model untuk menghasilkan output teks yang relevan, koheren, dan sesuai dengan konteks yang diminta. Kualitas respons dapat dipengaruhi oleh berbagai faktor, termasuk *prompt* yang diberikan, *hyperparameter* model, dan keragaman data pelatihan [14]. Dalam aplikasinya *prompt* merupakan input dari model, mengubah *prompt* dapat menghasilkan hasil yang berbeda secara signifikan [15]. Untuk itu, *prompt engineering*, pendekatan sistematis dalam merancang *prompt*, memiliki peran krusial dalam memaksimalkan kinerja LLM.

Publikasi *"Language Models are Few-Shot Learners"* oleh Brown et al. 2020 memberikan gambaran bahwa LLM seperti GPT-3 dapat dimaksimalkan dalam menyelesaikan tugas NLP menggunakan metode *in-context learning* [16]. *In-context learning* merupakan paradigma dalam *Natural Language Understanding* (NLU), di mana demonstrasi tugas ditambahkan ke dalam input *prompt* sehingga *pre-trained model* dapat menangani tugas tersebut tanpa memodifikasi parameter di dalamnya [17]. Penelitian ini membuka pengetahuan baru bahwa untuk memaksimalkan sebuah model dalam suatu tugas khusus dapat dilakukan tanpa memodifikasi bobot model (*fine-tuning*). Model yang cukup besar dapat "belajar" untuk melakukan tugas hanya dari beberapa contoh yang diberikan dalam *prompt* itu sendiri, tanpa memodifikasi model itu sendiri [16]. Brown et al. mengeksplorasi tiga variasi *in-context learning* yang dibedakan berdasarkan banyaknya contoh demonstrasi yang diberikan dalam *prompt*, yaitu

1. ***Few-Shot Prompting***
   Model diberikan beberapa contoh demonstrasi tugas pada saat inferensi. Suatu contoh biasanya memiliki konteks dan penyelesaian yang diinginkan. *Few-shot* bekerja dengan memberikan K buah contoh konteks dan penyelesaiannya, diikuti oleh input target yang harus diselesaikan oleh model. K biasanya berentang antara 10 s.d 100 bergantung dengan *context window* model [16].
2. ***One-Shot Prompting***
   Model diberikan 1 contoh demonstrasi ($K = 1$) beserta deskripsi dari tugas yang akan dikerjakan dalam bahasa alami. Contoh demonstrasi tersebut berfungsi sebagai panduan bagi model untuk memahami input sehingga menghasilkan output yang diharapkan [16].

3. ***Zero-Shot Prompting***

   Model diminta untuk melakukan tugas hanya berdasarkan deskripsi bahasa alami dari tugas tersebut tanpa diberikan contoh demonstrasi ($K = 0$). Sebagai contoh untuk tugas terjemah bahasa *"Terjemahkan kalimat bahasa Inggris berikut ke Bahasa Indonesia"* [16].

Tiga variasi *prompt* tersebut kemudian diuji pada model GPT-3 untuk menyelesaikan beberapa tugas khusus. Pada tugas *cloze and completion*, GPT-3 dengan 3 metode *prompt* secara signifikan melampaui SOTA (*State of the Art*) pada *dataset* LAMBADA (*LAnguage Modeling Broadened to Account for Discourse Aspects*) dengan peningkatan akurasi dari 68,0 (SOTA Turing-NLG) menjadi 86,4 (*few-shot*), 72,5 (*one-shot*), dan 76,2 (*zero-shot*). Pada kasus uji lain, penggunaan *prompt engineering* juga menunjukkan performa yang mendekati *state-of-the-art* sistem yang sudah dilakukan *fine-tuning*. Hasil tersebut menunjukkan bahwa metode *few-shot* memberikan peningkatan performa paling tinggi, disusul dengan *one-shot* dan *zero-shot*.

Sebuah penelitian oleh Islam et al. (2025) yang berjudul *"LLM-based Prompt Ensemble for Reliable Medical Entity Recognition from EHRs"* menunjukkan bahwa teknik *prompting* mempengaruhi output model secara signifikan. Penelitian ini secara khusus menyelidiki teknik *entity extraction* pada bidang medis berbasis *prompt* menggunakan LLM GPT-4o dan DeepSeek-R1. Teknik *prompting* seperti *zero-shot*, *few-shot*, dan *ensemble* pada GPT-4o dan DeepSeek-R1 dievaluasi untuk mencari tahu efektivitas dari teknik dan LLM tersebut dalam tugas *medical entity recognition*. Khusus pada *few-shot*, contoh demonstrasi yang ditambahkan pada *prompt* dibagi menjadi 3 perlakuan, yaitu *few-shot 1* berupa 1 sampel dokumen berlabel, *few-shot 2* berupa 100 kalimat berlabel yang diambil dari 5 dokumen, dan *few-shot 3* berupa sejumlah 5.355 contoh entitas. Untuk teknik *ensemble* melibatkan 4 jenis *prompting* sebelumnya yang kemudian dilakukan semacam voting untuk menentukan hasilnya. Dari hasil pengujian diperoleh teknik *few-shot 3* memperoleh akurasi paling tinggi sebesar 0,65 pada ekstraksi entitas disusul dengan *few-shot 2*, *few-shot 1*, *zero-shot*, dan *ensemble* sebesar 0,59, 0,56, 0,37, dan 0,37. Namun, pada pengujian klasifikasi entitas, teknik *few-shot 3* justru mendapat nilai F1 paling rendah sebesar 0,86 yang berkebalikan dengan teknik *ensemble* yang memiliki nilai F1 paling tinggi sebesar 0,95 dan *few-shot 2* dan *few-shot 1* sebesar 0,94. Hasil tersebut membuktikan bahwa *prompt engineering* meningkatkan performa LLM dalam tugas tertentu, khususnya ekstraksi entitas.

2.1.4 Implementasi Arsitektur *Retrieval-Augmented Generation* pada *Chatbot*

Retrieval-Augmented Generation (RAG) merupakan metode yang ada seiring dengan perkembangan LLM yang pesat. LLM standar sangat mahir dalam memberikan umpan balik yang bersifat umum. Namun, pada kasus khusus, LLM mengalami kesulit-

an dalam memberikan respons dan cenderung menghasilkan informasi yang tidak akurat dan relevan (halusinasi). Informasi yang dimiliki oleh LLM hanya terbatas oleh data yang digunakan pada saat *training* yang bersifat statis. Untuk itu, RAG hadir dengan memberikan pengetahuan tambahan kepada LLM agar respons yang diberikan dapat lebih akurat dan relevan.

RAG dirancang untuk meningkatkan kemampuan LLM dengan mengintegrasikan pengetahuan eksternal yang memperluas dan memperdalam basis pengetahuannya. LLM dapat mengakses pengetahuan tambahan melalui suatu mekanisme pengambilan informasi eksternal, sehingga memungkinkan model untuk memanfaatkan basis data pengetahuan yang relevan dan terkini saat menghasilkan respons. Metode ini dirancang untuk mengatasi keterbatasan model generatif umum dengan menggabungkan dua komponen yaitu mekanisme *retrieval* dan modul generatif. Mekanisme *retrieval* bertugas untuk mengambil informasi relevan dari sumber pengetahuan eksternal. Informasi ini kemudian disajikan sebagai konteks tambahan kepada modul generatif, biasanya berupa LLM, untuk menghasilkan respons teks yang sesuai.

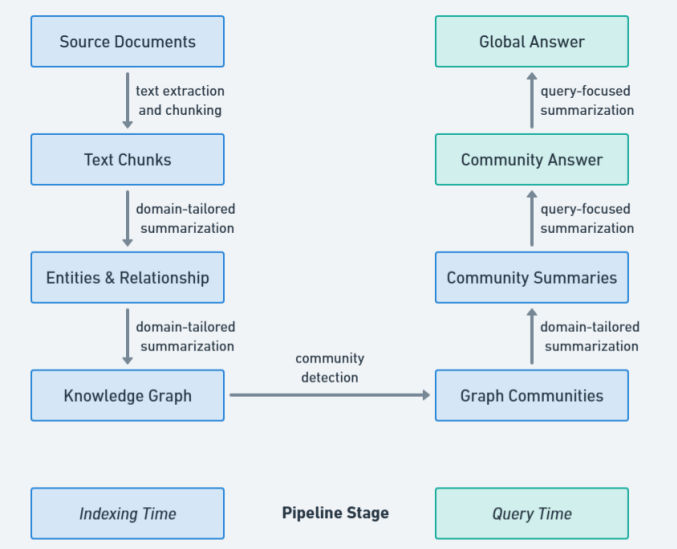
Arsitektur RAG yang dipopulerkan oleh Lewis et al, (2020) muncul dengan paradigma yang menjanjikan dengan menggabungkan keunggulan generatif LLM dengan kemampuan untuk mengakses dan mendasarkan respons pada pengetahuan eksternal. Model RAG dievaluasi dengan berbagai *knowledge-intensive task* seperti *Open-domain Question Answering*, *Abstractive Question Answering*, *Jeopardy Question Geneeretion*, dan *Fact Verification*. Hasil evaluasi tersebut menunjukkan bagaimana RAG dapat meningkatkan kinerja sebuah language model BART (400 juta parameter) dapat melampaui language model dengan parameter jauh lebih banyak (T5 11 miliar parameter) pada *knowledge-intensive task* [8]. Arsitektur RAG pada penelitian ini terdiri dari 2 komponen utama yaitu:

1. ***Retriever***
   Komponen *retriever* bertanggung jawab untuk menemukan dokumen atau potongan teks yang relevan dari korpus pengetahuan besar (dalam penelitian ini, Wikipedia) berdasarkan input pengguna. Lewis et al. menggunakan *Dense Passage Retriever* (DPR), yang terdiri dari *query encoder* yang mengubah input pengguna ke dalam representasi vektor, *document encoder* yang mengubah dokumen dalam korpus pengetahuan menjadi representasi vektor, dan mekanisme pencarian seperti BM25 dan MIPS (*Maximum Inner Product Search*) untuk mengambil dokumen yang relevan dengan kueri.
2. ***Generator***
   Komponen generator berupa model *sequence-to-sequence*, dalam penelitian ini BART-large, yang menghasilkan output akhir. *Generator* menerima input asli ditambah

dengan dokumen yang dihasilkan oleh *retriever* sebagai konteks tambahan untuk menghasilkan jawaban final.

2.1.5 Penggunaan *Knowlegde Graph* pada Arsitektur *Retrieval-Augmented Generation*

Knowledge Graph menggunakan model data berbasis *graph* untuk menangkap pengetahuan pada skenario aplikasi yang melibatkan integrasi, manajemen, dan ekstraksi data dari banyak sumber. Menggunakan abstraksi pengetahuan dengan model berbasis *graph* memberikan beberapa keuntungan dibandingkan model relasional atau alternatif NoSQL lain [18]. Model ini memberikan pemodelan pengetahuan lebih alami berupa *graph* yang terdiri dari *nodes* dan *edges* yang merepresentasikan entitas dan relasi [19].



Gambar 2.1. Alur kerja GraphRAG yang memanfaatkan indeks graf dan deteksi komunitas [2]

Penggunaan graf untuk memodelkan pengetahuan pada arsitektur RAG memberikan beberapa keuntungan dibandingkan RAG berbasis vektor. Pada RAG konvensional berbasis vektor, sistem akan mengambil sekumpulan dokumen yang secara individu relevan dengan kueri [8]. Namun, saat kueri membutuhkan pemahaman *dataset* secara global, RAG konvensional mengalami kesulitan karena basis pengetahuan disusun hanya sebagai potongan dokumen yang saling independen. Dalam penelitian Edge et al. 2025 yang berjudul "*From Local to Global: A GraphRAG Approach to Query-Focused Summarization*" mengintegrasikan model pengetahuan berbasis *graph* dalam RAG [2]. Alur kerja GraphRAG seperti pada 2.1 terdiri dari beberapa proses meliputi

1. ***Source Documents → Text Chunks***
   Pada proses ini, dokumen-dokumen awal dipecah menjadi potongan-potongan teks

(*text chunks*) dengan ukuran tertentu. Menentukan ukuran dari potongan teks ini merupakan perkara krusial mengingat potongan teks yang panjang mengurangi jumlah kueri ke LLM (hemat biaya) tetapi dapat menurunkan kualitas pengambilan informasi di awal potongan [2].

2. ***Text Chunks → Entities & Relationships***
   LLM akan digunakan sebagai ekstraktor entitas penting, relasi antar entitas, dan deskripsi singkat untuk keduanya dari potongan teks sebelumnya. Klaim yang berupa pernyataan faktual tentang entitas juga akan diekstrak oleh LLM [2]. *Prompt* LLM disesuaikan menggunakan teknik *few-shot* yang memberikan beberapa contoh spesifik hasil yang diharapkan agar respons yang diberikan LLM sesuai [16].

3. ***Entities Relationships → Knowledge Graph***
   Entitas, relasi, dan klaim yang telah diekstrak kemudian ditransformasi menjadi *nodes* (entitas) dan *edges* (relasi) dalam sebuah *kowledge graph*. Deskripsi entitas diagregasikan dengan entitas yang berupa *node*. Entitas, relasi, atau klaim dapat terdeteksi lebih dari satu kali karena proses ekstraksi dilakukan secara berulang dari banyak dokumen yang berbeda. Banyaknya duplikasi relasi menjadi bobot dari *edges* [2].

4. ***Knowledge Graph → Graph Communities***
   Berdasarkan *graph* yang telah terbentuk sebelumnya, komunitas akan dideteksi menggunaka algoritma tertentu yang bertujuan untuk memartisi *graph* menjadi beberapa komunitas yang terdiri dari kumpulan *nodes* yang terhubung dengan erat. Dalam penelitian ini digunakan *Leiden community detection* (Trag et al. 2019) secara hierarkis, mendeteksi sub-komunitas secara rekursif dalam setiap komunitas yang terdeteksi hingga mencapai komunitas yang tidak dapat lagi dipartisi [20].

5. ***Graph Communities → Community Summaries***
   Proses ini membuat ringkasan dari setiap komunitas dalam hierarki komunitas menggunakan metode tertentu. Ringkasan ini secara independen berguna untuk memahami struktur global dan semantik *dataset* serta dapat digunakan untuk memindai korpus tanpa adanya kueri tertentu. Ringkasan berfungsi sebagai bagian dari indeks graf yang digunakan untuk menjawab kueri global [2].

6. ***Community Summaries → Community Answers → Global Answer***
   Berdasarkan pertanyaan pengguna, ringkasan komunitas yang telah dibuat sebelumnya dapat digunakan untuk menghasilkan jawaban final melalui proses yang bertahap. Ringkasan komunitas diacak dan dibagi menjadi potongan-potongan (*chunks*) dengan ukuran token tertentu. LLM menghasilkan jawaban sementara (*intermediate answers*) dari *chunks* ini secara paralel, beserta skor seberapa membantu jawaban tersebut (0-100). Jawaban dengan skor 0 akan dibuang. Jawaban sementara diurut-

kan berdasarkan skornya, lalu dimasukkan ke konteks baru LLM hingga batas token tercapai untuk menghasilkan jawaban global (*global answer*) kepada pengguna [2].

2.2 Dasar Teori

2.2.1 Kesehatan Mental

Kesehatan mental, menurut definisi dari *World Health Organization* (WHO), adalah sebuah kondisi kesejahteraan mental (*state of mental well-being*) di mana setiap individu sadar akan potensi yang dimiliki, belajar dan bekerja dengan baik, dan memberikan kontribusi bagi komunitasnya [5]. Definisi tersebut menekankan bahwa kesehatan mental bukanlah sekadar kondisi bebas dari gangguan jiwa, melainkan sebuah fondasi yang memungkinkan seseorang untuk berpikir, belajar, merasakan emosi, berinteraksi dengan orang lain, dan menikmati hidup. Kesehatan mental merupakan komponen yang tidak terpisahkan dengan kesehatan secara keseluruhan dan sama pentingnya dengan kesehatan fisik.

2.2.2 *Natural Language Processing*

Pemrosesan bahasa alami (*Natural Language Processing*, NLP) merupakan serangkaian teknik komputasi yang didorong secara teoritis untuk menganalisis dan merepresentasikan teks alami pada beberapa tingkat analisis linguistik dengan tujuan mencapai pemrosesan bahasa seperti manusia untuk serangkaian tugas atau aplikasi [21]. Komputer tidak dapat mengerti bahasa manusia yang kompleks dan tidak pasti secara langsung, tetapi memerlukan sesuatu untuk menjadi penghubung antara keduanya. NLP hadir sebagai cabang dari kecerdasan buatan (AI) yang berfungsi sebagai jembatan antara bahasa manusia yang kompleks dengan komputasi komputer yang terstruktur. Tujuan utama dari NLP adalah memberdayakan komputer untuk memahami, menafsirkan, memanipulasi, dan menghasilkan bahasa manusia yang bermakna dan bermanfaat.

NLP dapat diklasifikasikan menjadi dua bagian, yaitu *Natural Language Understanding* (NLU) dan *Natural Language Generation* (NLG) [22]. Berikut merupakan penjelasan dari NLU dan NLG.

1. ***Natural Language Understanding* (NLU)**
   NLU memungkinkan komputer untuk memahami bahasa alami dan menganalisisnya dengan mengekstrak konsep, entitas, emosi, kata kunci, dll. NLU berkaitan dengan ilmu bahasa (linguistik) yang berupa morfologi atau bentuk kata, leksikal atau makna kata, sintaksis atau susunan kalimat, semantik atau makna kalimat, *discourse* atau hubungan antar kalimat, dan pragmatik atau makna kontekstual [22].
2. ***Natural Language Generation* (NLG)**
   NLG merupakan proses menghasilkan teks, dapat berupa kata, frasa, atau kalimat,

yang bermakna dari representasi internal. Proses ini terjadi dalam empat fase yaitu mengidentifikasi tujuan, merencanakan cara mencapai tujuan, mengevaluasi situasi dan sumber komunikasi yang tersedia, dan mewujudkan rencana sebagai teks [22].

Aplikasi NLP sangat luas dan meresap dalam kehidupan sehari-hari, mulai dari fungsionalitas mesin pencari, *chatbot* layanan pelanggan, asisten digital seperti Bixby, Siri dan Alexa, hingga aplikasi khusus seperti analisis sentimen, mesin penerjemah otomatis, dan ekstraksi informasi dari dokumen. Pada domain kesehatan mental, NLP telah menunjukkan potensi signifikan dalam menganalisis catatan klinis elektronik, unggahan di media sosial, dan respons survei untuk membantu dalam deteksi dini dan pemantauan kondisi seperti depresi, kecemasan, dan *Post-Traumatic Stress Disorder* (PTSD) [23].

2.2.3 *Machine Learning*

Pembelajaran mesin (*Machine Learning*, ML) adalah bidang AI yang fokus pada pembelajaran dengan mengembangkan algoritma terbaik untuk merepresentasikan data yang tersedia. Berbeda dengan pemrograman klasik, di mana algoritma dapat dikodekan secara eksplisit menggunakan fitur yang diketahui, ML menggunakan subset data untuk menghasilkan algoritma yang paling cocok untuk pola data tersebut [24]. Inti dari ML adalah kemampuannya untuk secara otomatis menemukan pola dalam data dan menggunakan pola tersebut untuk membuat inferensi atau prediksi pada data baru yang belum pernah dilihat.

Algoritma ML membangun model matematis berdasarkan kumpulan data sampel, yang dikenal sebagai "data pelatihan" (*training* data), untuk melakukan berbagai tugas seperti klasifikasi, regresi, atau pengelompokan. Dalam ML terdapat beberapa metode pembelajaran yang umum digunakan, yaitu *supervised* di mana model belajar dari data yang telah diberi label dengan output yang benar, *unsupervised* di mana model belajar menemukan pola tersembunyi dari data tanpa label, dan *reinforcement* di mana model belajar untuk membuat urutan keputusan dengan berinteraksi dengan lingkungannya untuk memaksimalkan imbalan kumulatif [24]. Penerapan ML sangat luas, mencakup berbagai domain seperti penyaringan email spam, sistem rekomendasi, pengenalan objek, pengenalan ucapan, diagnosis medis berdasarkan data pasien, dan analisis pasar keuangan untuk prediksi tren. ML menjadi sangat berguna dalam situasi di mana perancangan dan pengkodean algoritma konvensional yang eksplisit untuk melakukan tugas tertentu dianggap sulit atau tidak praktis karena kompleksitas masalah atau sifat data yang dinamis.

2.2.4 *Deep Learning*

Pembelajaran mendalam (*Deep Learning, DL*) merupakan sebuah pendekatan dalam *Machine learning* yang berusaha mempelajari representasi data melalui struktur mo-

del yang terdiri dari banyak lapisan (*layers*) [25]. Beberapa algoritma pembelajaran yang dikembangkan, awalnya dikembangkan sebagai model komputasi dari pembelajaran biologis yang meniru cara otak bekerja. Maka dari itu, nama lain dari *deep learning* adalah jaringan saraf tiruan (*Artificial Neural Network*, ANN) [25]. Jaringan tersebut digunakan untuk mempelajari representasi data dengan berbagai tingkat abstraksi. Istilah mendalam (*deep*) merujuk pada kedalaman atau banyaknya lapisan dalam jaringan saraf ini. Setiap lapisan dalam arsitektur DL menerima output dari lapisan sebelumnya, mentransformasikannya, dan meneruskannya ke lapisan berikutnya, secara bertahap mengekstraksi fitur-fitur yang semakin kompleks dan relevan dari data input mentah [25].

Sebagai subset dari *Machine Learning* (ML), yang juga merupakan subset dari AI, DL memiliki perbedaan signifikan dibandingkan dengan ML. DL memiliki kemampuan *feature engineering* secara otomatis. ML konvensional sering kali memerlukan intervensi manusia untuk merancang dan memilih fitur-fitur input yang relevan dari data, model DL dapat secara mandiri mempelajari hierarki fitur yang optimal langsung dari data mentah, seperti piksel dalam gambar atau kata-kata dalam teks [25]. Aplikasi DL telah merambah berbagai bidang, termasuk pengenalan gambar dan ucapan, deteksi objek dalam video, pemrosesan bahasa alami untuk terjemahan dan pemahaman, serta pengembangan kendaraan otonom.

2.2.5 *Large Language Model*

Model bahasa besar (*Large Language Model*, LLM) merupakan model bahasa berskala besar yang mampu memahami, memprediksi, dan menghasilkan bahasa manusia secara alami dan kontekstual [26]. LLM merupakan bentuk lanjutan dari *language models* (LMs), yang secara umum adalah sistem komputasi yang mempelajari probabilitas urutan kata dalam teks. Model bahasa awal seperti *n-gram models* memprediksi kata berikutnya berdasarkan konteks kata-kata sebelumnya, namun memiliki keterbatasan seperti ketidakmampuan menangani kata yang jarang muncul, kecenderungan *overfitting*, serta kesulitan dalam memahami struktur linguistik yang kompleks. LLM hadir untuk mengatasi tantangan tersebut dengan memanfaatkan arsitektur modern dan kapasitas pembelajaran yang jauh lebih besar [26].

Keunggulan lain dari LLM terletak pada kemampuan *in-context learning*, yaitu kemampuan untuk melakukan adaptasi dan menyelesaikan berbagai tugas hanya dari petunjuk yang diberikan dalam bentuk *prompt*, tanpa perlu pelatihan ulang secara eksplisit. LLM dapat "belajar" dari konteks input yang diberikan pengguna untuk memberikan respons yang relevan dan akurat. Fitur ini membuat LLM sangat fleksibel dan dapat digunakan untuk berbagai keperluan, mulai dari menjawab pertanyaan, membuat ringkasan, menerjemahkan teks, menulis kode program, hingga mengekstrak entitas dari sebuah dokumen [26].

2.2.6 Gemini

Gemini adalah famili model AI multimoda yang dikembangkan oleh Google DeepMind. Model Gemini dibangun berdasarkan arsitektur transformer *decoders*. Model ini dilatih dengan data multimoda dan multilingual yang berasal dari *website*, buku, dan kode, termasuk juga gambar, audio, dan video [27]. Saat penelitian dilakukan, Gemini telah berada pada versi 2.5 yang menggunakan arsitektur *sparse mixture-of-experts* (MoE) *transformer*. Gemini 2.5 mendukung input konteks dengan panjang lebih dari 1 juta token yang memungkinkan untuk memahami data dengan cakupan yang luas [28].

2.2.7 GPT

Generative Pre-trained Transformer (GPT) adalah *language model* berbasis arsitektur transformer yang dikembangkan oleh OpenAI sejak tahun 2018 saat rilis model transformer pertamanya. Saat penelitian ini dilakukan, OpenAI telah memperbarui model GPT hingga generasi keempat seperti gpt-4.1. Model ini dilatih secara unsupervised pada kumpulan data dalam jumlah besar yang berasal dari *website* publik, kode dan data matematis, serta data multimodal seperti gambar, audio, dan video. Pelatihan tersebut bertujuan untuk memahami pola bahasa alami, lalu digunakan untuk menghasilkan teks baru yang koheren. GPT bekerja dengan mekanisme *self-attention* yang memungkinkan model memahami konteks panjang dalam sebuah teks, sehingga mampu menghasilkan jawaban yang relevan dan konsisten [29].

2.2.8 *Retrieval-Augmented Generation*

Retrieval-Augmented Generation (RAG) adalah teknik yang menggabungkan kekuatan *Large Language Model* (LLM) dengan sistem pengambilan (*retrieval*) informasi dari sumber eksternal (seperti basis pengetahuan atau dokumen) untuk meningkatkan akurasi, ketepatan, dan kedalaman pengetahuan dalam tugas *knowledge-intensive* seperti tanya-jawab terbuka atau penjelasan faktual. Tujuan utama RAG adalah untuk mengatasi beberapa keterbatasan yang melekat pada LLM standar, seperti kecenderungan untuk menghasilkan informasi yang tidak akurat atau "berhalusinasi", menyajikan pengetahuan yang tidak relevan, atau kurangnya kemampuan untuk menyediakan sumber atau justifikasi atas informasi yang dihasilkannya [8].

2.2.9 Token

Dalam bidang *Natural Language Processing* (NLP), token merujuk pada satuan terkecil dari teks yang dapat dikenali dan diproses oleh sistem komputasi bahasa. Token dapat berupa kata, bagian dari kata (*subword*), tanda baca, atau bahkan karakter individual, tergantung pada strategi tokenisasi yang digunakan [30]. Proses pemecahan teks menjadi token disebut tokenisasi, dan merupakan tahap awal penting dalam *pipeline*

pemrosesan bahasa alami. Dalam model *Large Language Model* seperti GPT atau BERT, tokenisasi umumnya dilakukan dengan pendekatan berbasis *subword*, seperti *Byte Pair Encoding* (BPE) atau WordPiece, yang memungkinkan model mengenali dan memproses kata-kata baru atau langka dengan efisien [31]. Tokenisasi berbasis *subword* menyusun token dari unit-unit kata yang lebih kecil sehingga kosakata tetap terbatas namun tetap mencakup variasi linguistik yang luas. Misalnya, kata "memperjuangkan" dapat dipecah menjadi beberapa token seperti "mem", "perjuang", dan "kan".

2.2.10 Vektor

Vektor adalah sebuah objek matematis yang memiliki dua karakteristik utama yaitu magnitudo (besar) dan arah. Secara fundamental, sebuah vektor dapat direpresentasikan sebagai sebuah larik (*array*) terurut yang berisi angka. Vektor tersebut merupakan elemen dari suatu ruang vektor (*vector space*) yang merupakan himpunan objek yang memiliki operasi penjumlahan dan perkalian skalar serta memenuhi sejumlah aksioma tertentu [32]. Sebuah vektor dalam ruang berdimensi n dapat direpresentasikan sebagai barisan terurut dari n bilangan real seperti pada Persamaan 2-1.

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \in \mathbb{R}^n \tag{2-1}$$

dengan $v_1, v_2, \cdots, v_n$ adalah komponen-komponen dari vektor $\mathbf{v}$ dan n adalah dimensi ruang tempat vektor tersebut berada. Dalam konteks kecerdasan buatan, vektor biasanya digunakan untuk merepresentasikan objek kompleks seperti teks, gambar, atau video ke dalam bentuk numerik yang dapat diproses oleh mesin.

2.2.11 *Cosine Similarity*

Kesamaan kosinus (*cosine similarity*) merupakan salah satu metrik matematis untuk yang digunakan untuk mengukur tingkat kemiripan antara dua vektor *non-zero*. Secara fundamental, metrik ini menghitung nilai kosinus dari sudut yang dibentuk oleh dua vektor tersebut. Fokus utama dari *cosine similarity* adalah pada orientasi atau arah vektor-vektor tersebut dan mengabaikan besar atau magnitudonya [33]. Hal ini menjadikannya sangat berguna dalam konteks di mana besaran absolut dari vektor kurang relevan dibandingkan dengan "arah" atau "profil" relatifnya. Secara matematis, formula untuk menghitung nilai *cosine similarity* dapat diungkapkan dengan Persamaan 2-2.

$$Similarity(\vec{A}, \vec{B}) = \cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\|\|\vec{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (2\text{-}2)$$

dengan $\vec{A} \cdot \vec{B}$ adalah *dot product* dari vektor A dan B, $\|\vec{A}\|$ adalah magnitudo atau besar dari vektor A dan $\|\vec{B}\|$ adalah magnitudo atau besar dari vektor B. *Dot product* $\vec{A} \cdot \vec{B}$ dilakukan dengan mengalikan komponen-komponen yang bersesuaian dari kedua vektor kemudian menjumlahkan hasilnya. Magnitudo dari sebuah vektor dihitung sebagai akar kuadrat dari jumlah kuadrat setiap komponennya. Nilai *cosine similarity* akan selalu berada pada rentang -1 hingga 1. Nilai 1 menunjukkan bahwa kedua vektor memiliki orientasi yang sama persis atau sudut yang dibentuk keduanya 0 derajat. Nilai 0 menunjukkan bahwa kedua vektor ortogonal atau saling tegak lurus (sudut 90 derajat) yang sering diartikan sebagai tidak adanya kesamaan orientasi. Nilai -1 menunjukkan bahwa kedua vektor memiliki orientasi yang berlawanan arah (sudut 180 derajat). Dua buah vektor dikatakan mirip apabila nilai *cosine similarity* antar keduanya mendekati 1.

2.2.12 *Vector Embedding*

Vector embedding merupakan representasi numerik dari sebuah data, terutama yang tidak terstruktur seperti teks, gambar, audio, atau video. Pada prinsipnya *vector embedding* bekerja dengan melakukan transformasi data ke dalam bentuk vektor berdimensi tinggi. Bentuk vektor tersebut merepresentasikan fitur atau karakteristik dari objek asli, seperti makna dalam teks, objek dalam gambar, atau suara dalam audio. Dalam konteks NLP *embedding* digunakan untuk menangkap hubungan semantik, sintaksis, atau kontekstual yang mendasari sebuah teks. Hal tersebut membuat hubungan antar objek seperti kemiripan semantik teks dapat dianalisis secara matematis [34].

Proses pembentukan *vector embedding* biasanya melibatkan model *machine learning* tradisional seperti Word2Vec dan GloVe atau model berbasis *transformer* yang lebih modern seperti BERT, GPT, dan Gemini. Model-model tersebut telah dilatih untuk menghasilkan nilai-nilai dalam *vector embedding* berdasarkan makna atau konteks setiap unit data. Model tradisional seperti GloVe bekerja dengan mentransformasi setiap kata menjadi 1 representasi vektor unik tanpa memperhatikan bagai mana kata tersebut digunakan dalam kalimat. Sebaliknya, model yang lebih modern seperti BERT, GPT, dan Gemini mampu menghasilkan representasi vektor yang berbeda untuk kata yang sama tergantung pada konteks spesifik kalimat di mana kata tersebut muncul.

2.2.13 *Full-Text Search*

Dalam konteks *information retrieval*, *full-text search* merujuk kepada teknik untuk melakukan pencarian keseluruhan konten tekstual dari sebuah atau koleksi dokumen

yang tersimpan. *Full-text search* adalah jenis pencarian yang dilakukan dengan mencocokkan istilah (*terms*) yang terkandung dalam kueri pencarian dengan *terms* dalam setiap dokumen dalam basis data, kemudian hasilnya di-*ranking* dengan algoritma tertentu. Jenis pencarian ini banyak digunakan di berbagai bidang, mencakup pencarian bahasa alami yang banyak dijumpai pada mesin pencari, pencarian situs web, dan banyak basis data [35]. Keunggulan utama dari metode pencarian ini adalah kemampuannya dalam menangani variasi teks, termasuk *stemming*, *stopwords*, dan peringkat relevansi hasil. Misalnya, sistem dapat mendeteksi bahwa kata "berlari" dan "lari" memiliki makna dasar yang sama, sehingga hasil pencarian menjadi lebih akurat. Dalam implementasi modern, *full-text search* biasanya dilengkapi dengan teknik seperti *inverted index*, *ranking model*, dan skor relevansi [36].

2.2.14 *Vector Database*

Basis data vektor (*vector database*) adalah salah satu jenis basis data yang dirancang khusus untuk secara efektif mengelola data yang direpresentasikan sebagai vektor berdimensi tinggi. Vektor-vektor tersebut merupakan representasi numeris dari fitur atau atribut suatu objek yang umumnya dibentuk melalui proses *vector embedding*. Setiap vektor dapat memiliki puluhan hingga ribuan dimensi tergantung pada kompleksitas data yang direpresentasikan. Vektor-vektor ini kemudian sering kali dikelompokkan atau diindeks berdasarkan kedekatan mereka dalam ruang vektor multidimensi untuk memudahkan dalam pencarian [37].

Basis vektor berbeda secara fundamental dengan basis data relasional berbasis baris dan kolom, basis data ini unggul dalam menangani volume besar data tidak terstruktur yang telah ditransformasikan menjadi representasi vektor padat berdimensi tinggi. Fitur kueri pencarian kemiripan (*similarity search*) menjadikannya sangat unggul dibandingkan basis data lain karena fitur tersebut merupakan operasi krusial bagi banyak aplikasi AI. Umumnya basis data vektor seperti ChromaDB, Pinecone, Melvis, dll. mendukung algoritma *Approximate Nearest Neighbor* (ANN) yang efisien untuk menemukan vektor-vektor yang paling mirip dengan vektor kueri dalam data yang sangat besar [37]. Hal tersebut menjadikannya pilihan utama saat berinteraksi dengan data berbentuk vektor.

2.2.15 *Graph Database*

Basis data graf (*graph database*) adalah basis data yang dirancang untuk secara efisien mengelola data yang dimodelkan dalam bentuk graf. Struktur data pada basis data graf secara fundamental terdiri atas *node* atau *vertex* yang merepresentasikan sebuah entitas atau objek dan *edge* yang merepresentasikan koneksi atau relasi antar node tersebut. Baik *node*, maupun *edge* dapat memiliki properti yang menyimpan atribut atau metadata yang dapat berisi deskripsi dari entitas atau relasi tersebut [38]. Salah satu keunggulan

dari basis data graf adalah adanya konsep *index-free adjacency* yang berarti setiap *node* dapat secara langsung mengetahui dan dapat mengakses *node* lain yang berhubungan tanpa harus menggunakan indeks tambahan seperti pada basis data relasional. Hal tersebut mempercepat performa pencarian hubungan antar *node* dan memungkinkan algoritma seperti *graph traversal* yang efisien [38].

2.2.16 *Knowledge Graph*

Konsep Graf Pengetahuan (*Knowledge Graph*, KG) pertama kali diperkenalkan secara luas oleh Google pada tahun 2012 melalui proyek Google *Knowledge Graph* untuk meningkatkan relevansi hasil pencarian dan pengalaman pencarian pengguna [39]. Bermula dari itu, banyak perusahaan dan institusi mengembangkan KG mereka sendiri seperti Bing dengan Stori, DBpedia, YAGO, Wikidata, dan Freebase sebagai bagian dari *semantic web* [40]. *Knowledge Graph* adalah representasi semantik terstruktur dari pengetahuan yang memodelkan entitas (seperti orang, tempat, organisasi, dll.) dan hubungan antar entitas dalam bentuk graf. Setiap entitas direpresentasikan sebagai *node* dan relasi antar entitas direpresentasikan sebagai *edge* [9]. Secara matematis, sebuah struktur KG dapat direpresentasikan dengan Persamaan 2-3

$$KG = \langle E, R, T \rangle \tag{2-3}$$

dengan E adalah himpunan semua *node* atau entitas dalam pengetahuan, R adalah himpunan semua relasi, dan T adalah himpunan triplet yang merepresentasikan fakta. T dapat dinyatakan secara matematis, seperti pada Persamaan 2-4

$$T = \{(h, r, t) \mid h, t \in E, r \in R\} \tag{2-4}$$

Setiap triplet T terdiri dari tiga bagian yaitu h (*head*) adalah entitas asal, r adalah relasi, dan t (*tail*) adalah entitas tujuan.

2.2.17 *Knowledge Extraction*

Ekstraksi pengetahuan (*Knowledge Extraction*, KE) merupakan proses sistematis untuk mengidentifikasi dan mengekstrak informasi bermakna dari data tidak terstruktur atau semi terstruktur seperti teks untuk direpresentasikan ke dalam bentuk terstruktur yang dapat dipahami oleh mesin. Dalam konteks *Knowledge Graph*, sumber data yang berupa dokumen akan diekstrak menjadi bentuk triplet (subjek-predikat-objek) yang akan membentuk KG. Pada praktiknya proses ekstraksi ini terdiri dari ekstraksi entitas, relasi antar entitas, dan klasifikasi entitas. Proses ini merupakan tahap awal yang krusial dalam konstruksi KG karena akan menentukan kualitas representasi pengetahuan oleh *Knowledge Graph* [10].

2.2.18 *Knowledge Retrieval*

Pengambilan pengetahuan (*Knowledge Retrieval*, KR) merupakan proses pencarian dan pengambilan pengetahuan secara sistematis dari basis data pengetahuan seperti *Knowledge Graph* [41]. Dalam konteks KG, proses pengambilan ini memungkinkan suatu sistem untuk melakukan inferensi logis dan menjawab kueri secara presisi dan benar. Penekanan utama dalam *Knowledge Retrieval* adalah pada pengambilan pengetahuan yaitu representasi formal pengetahuan yang dapat diolah untuk penalaran, bukan sekadar pengambilan data mentah atau informasi yang terkait dengan kueri. KR bertujuan untuk mengambil makna dan hubungan terstruktur yang kemudian dapat dimanfaatkan untuk tugas lain yang lebih kompleks.

2.2.19 *Precision*

Precision (presisi) merupakan metrik yang mengukur seberapa banyak item yang berhasil diidentifikasi (*retrieved*) yang benar-benar relevan. Metrik ini juga dikenal sebagai *confidence* dalam bidang *data mining* dan menjadi fokus utama dalam *information retrieval* dan *machine learning* untuk mengukur keyakinan pada hasil prediksi. Presisi menunjukkan proporsi kasus positif yang diprediksi dengan positif yang sebenarnya [42]. Definisi *Precision* diberikan pada Persamaan 2-5.

$$Precision = \frac{TP}{TP + FP} = \frac{\textit{Relevant items retrieved}}{\textit{Retrieved items}} \tag{2-5}$$

dengan:

- TP adalah *True Positives*, yaitu jumlah prediksi positif yang benar.
- FP adalah *False Positives*, yaitu jumlah prediksi positif yang salah.

Dalam kasus RAG, presisi berarti proporsi banyaknya informasi relevan dari semua informasi yang dihasilkan dibandingkan dengan total informasi yang dihasilkan. Metrik ini dapat menunjukkan tingkat *noise* dari konteks yang dihasilkan *retriever*.

2.2.20 *Recall*

Recall merupakan metrik yang mengukur seberapa banyak item relevan yang berhasil diidentifikasi dari semua item yang relevan. Metrik ini juga dikenal sebagai *sensitivity* atau *True Positive Rate* (TPR). *Recall* menunjukkan proporsi kasus positif yang diprediksi dengan total kasus positif yang seharusnya muncul [42]. Definisi *Recall* diberikan pada Persamaan 2-6.

$$Recall = \frac{TP}{TP + FN} = \frac{\textit{Relevant items retrieved}}{\textit{Relevant items}} \tag{2-6}$$

dengan:

- TP adalah *True Positives*, yaitu jumlah prediksi positif yang benar.
- FN adalah *False Negatives*, yaitu jumlah kasus positif yang gagal diprediksi (salah diprediksi negatif).

Dalam kasus RAG *recall* berarti proporsi banyaknya informasi relevan dari semua informasi yang dihasilkan dibandingkan dengan total informasi relevan yang seharusnya muncul. Metrik ini dapat menunjukkan seberapa banyak informasi relevan yang dihasilkan *retriever* dari yang seharusnya muncul.

2.2.21 *Mean Reciprocal Rank* (MRR)

Metrik *Reciprocal Rank* (RR) menghitung kebalikan dari peringkat (*rank*) di mana dokumen relevan pertama ditemukan. RR bernilai 1 jika dokumen relevan ditemukan pada peringkat 1, 0,5 jika ditemukan pada peringkat 2, 0,33 pada peringkat 3, dan seterusnya. Ketika dirata-ratakan di seluruh kueri, ukuran ini disebut *Mean Reciprocal Rank* (MRR) [43]. MRR didefinisikan dalam Persamaan 2-7

$$\mathrm{MRR} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\mathrm{rank}_i} \tag{2-7}$$

Dalam konteks RAG, MRR menunjukkan performa *retriever* dalam memprioritaskan urutan munculnya konteks yang relevan. Hal tersebut cukup krusial karena beberapa LLM memiliki kecenderungan memprioritaskan informasi yang muncul di awal.

2.2.22 *Retrieval-Augmented Generation Assessment* (RAGAS) *Framework*

RAGAS (*Retrieval-Augmented Generation Assessment*) adalah sebuah kerangka kerja yang dirancang untuk mengevaluasi sistem AI yang menggunakan arsitektur *Retrieval-Augmented Generation* (RAG). Tujuan utamanya adalah untuk mengatasi tantangan dalam menilai kualitas *pipeline* RAG, yang terdiri dari komponen pengambilan (*retrieval*) dan generasi (*generation*). Kerangka kerja ini menyediakan serangkaian metrik otomatis yang dapat mengukur berbagai aspek kinerja berbasis LLM sehingga mampu untuk mengevaluasi data dalam bentuk bahasa alami. Dengan demikian, RAGAS memungkinkan pengembang untuk secara efisien mendiagnosis kelemahan, baik pada kemampuan sistem untuk menemukan informasi yang relevan, maupun pada kemampuannya untuk menghasilkan jawaban yang akurat dan koheren dari informasi tersebut [44].

Untuk memberikan evaluasi yang komprehensif, RAGAS menyediakan beberapa metrik utama untuk mengevaluasi sistem RAG.

1. ***Faithfulness***

   Metrik pertama adalah *faithfulness* atau kepatuhan. *Faithfulness* adalah metrik kru-

sial yang mengukur konsistensi faktual jawaban yang dihasilkan terhadap konteks yang diambil. Metrik ini memastikan bahwa model tidak berhalusinasi dengan memverifikasi bahwa setiap klaim dalam jawaban didukung oleh sumber yang diberikan.

2. ***Response Relevancy***
   Response Relevancy menilai seberapa relevan jawaban terhadap pertanyaan asli. Jawaban yang tidak lengkap atau mengandung informasi berlebihan yang tidak diminta akan dikenakan penalti. Semakin tinggi nilainya menunjukkan respons selaras dengan pertanyaan, sedangkan semakin kecil nilainya menunjukkan respons kurang lengkap atau mengandung informasi yang redundan. Metrik ini tidak mengukur kebenaran, melainkan fokus pada ketepatan dalam menjawab pertanyaan.

3. ***Answer Similarity***
   Answer Similarity secara spesifik mengukur kemiripan semantik antara jawaban yang dihasilkan dan jawaban referensi, biasanya dihitung menggunakan *cosine similarity* dari *embedding* kedua jawaban tersebut. Secara bersama-sama, metrik-metrik ini memberikan pandangan holistik yang memungkinkan pengembang untuk memahami secara mendalam apakah jawaban yang dihasilkan tidak hanya relevan dan setia pada konteks, tetapi juga benar dan mirip secara semantik dengan jawaban yang ideal.

4. ***Correctness***
   Selanjutnya, *Correctness* mengevaluasi keakuratan jawaban secara keseluruhan dengan membandingkannya dengan jawaban referensi (*ground truth*). Metrik ini bergantung pada *ground truth* dan jawaban, dengan skor berkisar antara 0 hingga 1. Skor yang lebih tinggi menunjukkan kesesuaian yang lebih dekat antara jawaban yang dihasilkan dan kebenaran dasar, yang menandakan ketepatan yang lebih baik.

2.3 Analisis Perbandingan Metode

Dalam tinjauan pustaka telah diulas berbagai metode untuk mengembangkan *chatbot*. Komparasi dari masing-masing metode tersebut terdapat pada Tabel 2.1.

Tabel 2.1. Ringkasan dan Perbandingan Metode Pengembangan *Chatbot*

Metode	**Kelebihan**	**Kekurangan**
***Fine-Tuning LLM*:** Melatih lebih lanjut sebuah *Large Language Model* (LLM) yang sudah ada dengan menggunakan *dataset* spesifik domain. Tujuannya adalah untuk mengadaptasi pengetahuan dan gaya respons model agar sesuai dengan kebutuhan.	1. Menghasilkan respons yang sangat relevan dan akurat untuk domain spesifik. 2. Gaya bahasa dan "kepribadian" *chatbot* dapat disesuaikan. 3. Latensi lebih rendah saat inferensi karena tidak butuh konteks eksternal.	1. Membutuhkan *dataset* pelatihan yang besar dan berkualitas tinggi. 2. Prosesnya mahal secara komputasi dan memakan waktu. 3. Pengetahuan sulit diperbarui tanpa melatih ulang. 4. Masih rentan terhadap "halusinasi".
***Prompt Engineering LLM*:** Merancang dan menyusun instruksi (*prompt*) yang detail dan terstruktur untuk mengarahkan LLM agar memberikan respons yang diinginkan tanpa mengubah model itu sendiri.	1. Tidak memerlukan pelatihan ulang. 2. Biaya rendah dan fleksibel. 3. Sangat mudah dilakukan pembaruan sesuai kebutuhan.	1. Performa sangat tergantung kualitas *prompt*. 2. Konsistensi jawaban tidak selalu terjamin. 3. Sulit untuk menangani domain yang sangat spesifik.

Metode	Kelebihan	Kekurangan
***Retrieval-Augmented Generation* Konvensional:** Menggabungkan LLM dengan sistem pencarian informasi dari basis data eksternal (misalnya, dokumen teks). Sebelum menjawab, sistem mencari informasi relevan terlebih dahulu, lalu memberikannya sebagai konteks bagi LLM.	1. Mengurangi halusinasi karena jawaban didasarkan pada sumber nyata. 2. Memudahkan pembaruan informasi tanpa proses *training* ulang.	1. Kualitas jawaban sangat bergantung pada kualitas *retriever*. 2. Butuh *pipeline* tambahan untuk pencarian data yang dapat menambah latensi.
***Retrieval-Augmented Generation* dengan *Knowledge Graph*:** Varian dari RAG yang menggunakan *Knowledge Graph* (KG) sebagai basis data eksternal. KG merepresentasikan informasi sebagai entitas dan hubungan yang terstruktur, memberikan konteks yang lebih kaya kepada LLM.	1. Lebih terstruktur dalam memodelkan informasi dibanding dokumen teks biasa. 2. Mendukung pencarian dengan konteks yang lebih luas. 3. Memperkuat konteks relasional antar data. 4. Dapat meng-*update* informasi tanpa *retraining*	1. Pembuatan *Knowledge Graph* memerlukan pemodelan yang optimal. 2. Kompleksitas implementasi tinggi.

Metode *Fine-Tuning* LLM menawarkan beberapa keunggulan utama, seperti kemampuan untuk menghasilkan respons yang sangat relevan dan akurat untuk domain spesifik. Selain itu, metode ini memungkinkan pengembang untuk menyesuaikan gaya bahasa dan "kepribadian" *chatbot*, yang didukung oleh latensi yang lebih rendah karena tidak memerlukan konteks eksternal saat inferensi.

Namun, di balik kelebihannya, terdapat kekurangan yang signifikan, proses *fine-tuning* membutuhkan *dataset* pelatihan yang besar dan berkualitas tinggi, menjadikannya mahal dan memakan waktu secara komputasi. Pengetahuan yang ditanamkan ke dalam

model juga bersifat statis, sehingga sulit untuk diperbarui tanpa melalui proses pelatihan ulang yang intensif. Kelemahan yang paling krusial adalah metode ini masih rentan terhadap "halusinasi", yaitu memberikan informasi yang salah atau tidak faktual.

Dalam konteks yang sangat sensitif seperti layanan kesehatan mental, risiko halusinasi dari metode *fine-tuning* menjadi opsi yang riskan. Oleh karena itu, metode RAG berbasis *Knowledge Graph* muncul sebagai solusi yang lebih unggul. Dengan mendasarkan jawaban pada basis data terstruktur yang memetakan entitas dan relasinya, RAG dengan KG secara signifikan mengurangi ambiguitas dan meningkatkan akurasi fakta. Kemampuannya untuk melakukan penalaran yang kompleks berdasarkan sumber terverifikasi menjadikannya pilihan yang lebih aman dan andal untuk membangun *chatbot* kesehatan mental yang bertanggung jawab.

BAB III
METODE PENELITIAN

Bab ini menyajikan ulasan mengenai alat dan bahan yang akan dipakai untuk menyelesaikan tugas akhir ini. Alat dan bahan tersebut meliputi perangkat keras, perangkat lunak, dan data yang digunakan. Selanjutnya akan dijelaskan mengenai rancangan serta langkah-langkah metodologis yang akan diimplementasikan dalam pelaksanaan penelitian ini.

3.1 Alat dan Bahan Tugas Akhir

3.1.1 Alat Tugas Akhir

Alat-alat yang digunakan dalam penelitian berupa perangkat keras dan perangkat lunak antara lain.

3.1.1.1 Perangkat Keras

1. Laptop Lenovo Ideapad Slim 5i dengan spesifikasi sistem operasi Windows 11, *processor* Intel Core i5 1135G7, RAM 8 GB LPDDR4X, grafis NVIDIA MX 450 2GB, SSD M.2 NVME 512 GB.
2. Neo4j Aura console Free Tier
3. Pinecone Free Tier

3.1.1.2 Perangkat Lunak

1. Visual Studio Code
2. Python 3.11.5
3. Kumpulan pustaka python seperti numpy, pandas, fastapi, dll.
4. RAGAS *framework*
5. Google Gemini 2.5 Flash
6. Google text-embedding-001
7. OpenAI o4-mini
8. OpenAI gpt-4.1-nano
9. OpenAI text-embedding-ada-002
10. Neo4j Graph Database
11. Pinecone Vector Database
12. Layanan konversi file FreeConvert

3.1.2 Bahan Tugas Akhir

Berikut merupakan bahan yang akan digunakan dalam tugas akhir.

1. Buku "Petunjuk Teknis Pencegahan dan Pengendalian Gangguan Mental Emosional" yang diakses dari *website* Kementerian Kesehatan.
2. Buku "Panduan Kesehatan Jiwa pada Masa Pandemi COVID-19" yang diakses dari *website* Kementerian Kesehatan
3. Buku "Panduan Pertolongan Pertama Psikologis Pada Upaya Bunuh Diri" yang diakses dari *website* CPMH
4. Artikel dari *website* CPMH
5. Artikel dari *website* HPU UGM

3.2 Metode yang Digunakan

Penelitian ini menerapkan pendekatan hibrida yang mengintegrasikan kekuatan *Large Language Models* (LLM) untuk pemahaman bahasa alami dengan representasi pengetahuan terstruktur dari *Knowledge Graph*. Metode yang digunakan berfokus pada dua area utama yaitu ekstraksi pengetahuan berbasis *few-shot learning* dan mekanisme pengambilan pengetahuan yang adaptif.

Bagian ekstraksi pengetahuan, berbeda dengan metode *supervised learning* yang memerlukan *dataset* beranotasi skala besar, penelitian ini menggunakan pendekatan *few-shot learning*. Sebuah *pre-trained* LLM diberi beberapa contoh (*shots*) untuk mengekstrak entitas dan relasi dari teks sumber secara langsung. Informasi yang diekstrak kemudian tidak disimpan dalam format vektor mentah, melainkan direpresentasikan sebagai *node* dan *edge* dalam tipe data graf yang disebut *Knowledge Graph*. Setiap entitas yang diekstrak juga akan memiliki representasi vektor yang memungkinkan pencarian semantik. Metode ini memungkinkan representasi pengetahuan yang kaya secara semantik dan terstruktur, yang menjadi dasar untuk penalaran relasional. Proses penyerapan data diperkuat dengan mekanisme resolusi entitas untuk memastikan konsistensi dan meminimalkan redundansi dalam KG.

Proses pengambilan informasi dari KG tidak bersifat monolitik, melainkan adaptif terhadap niat pengguna. Mekanisme *retrieval* dimulai dengan klasifikasi kueri untuk membedakan antara pertanyaan yang berfokus pada atribut sebuah entitas dan pertanyaan yang mengeksplorasi hubungan antar entitas. Berdasarkan klasifikasi ini, sistem akan menjalankan strategi *graph traversal* yang paling efisien. Pencarian *node* awal di dalam *graph* juga menggunakan pendekatan *hybrid search*, yang menggabungkan kecepatan *full-text search* untuk pencocokan kata kunci yang tepat dengan fleksibilitas *dense vector search* untuk pemahaman makna semantik.

3.3 Alur Tugas Akhir

Gambar 3.2. Alur Penelitian

Pelaksanaan penelitian ini akan mengikuti suatu alur yang terdiri dari beberapa tahapan utama yang saling terkait dan berurutan, namun juga memungkinkan adanya iterasi pada beberapa fase tertentu, terutama pada tahap pengembangan dan evaluasi model. Visualisasi alur penelitian ini disajikan dalam Gambar 3.2. Tahapan-tahapan fundamental tersebut secara terperinci adalah sebagai berikut.

3.3.1 Identifikasi Masalah

Tahap awal ini merupakan fondasi dari keseluruhan penelitian. Proses ini dimulai dengan identifikasi masalah umum mengenai adanya kesenjangan akses terhadap informasi kesehatan mental yang akurat dan terstruktur di Indonesia. Dari masalah umum tersebut, dirumuskan masalah teknis yang spesifik: bagaimana teknologi RAG dapat dioptimalkan untuk mengatasi tantangan ini. Identifikasi masalah mencakup analisis kelemahan RAG berbasis vektor standar, yang cenderung kesulitan dalam melakukan penalaran atas hubungan kompleks. Berdasarkan analisis ini, diajukan hipotesis bahwa penggunaan *Knowledge Graph* dapat memberikan konteks yang lebih kaya dan terstruktur.

Tahapan ini menghasilkan rumusan masalah utama yang berfokus pada optimalisasi ekstraksi pengetahuan untuk membangun KG yang konsisten dan optimalisasi pengambilan pengetahuan dari KG untuk menjawab kueri pengguna secara akurat.

3.3.2 Studi Literatur

Studi literatur dilakukan secara sistematis, komprehensif, dan berkelanjutan. Hal tersebut bertujuan membangun landasan teoritis yang kokoh, memahami tren dan perkembangan terkini *state-of-the-art* dalam bidang terkait, mengidentifikasi celah penelitian (*research gap*) yang dapat diisi, serta mempelajari berbagai metode dan alat bantu yang relevan. Proses ini melibatkan pencarian, pengumpulan, analisis, dan sintesis informasi dari berbagai sumber ilmiah jurnal internasional seperti IEEE, ArXiv, JMIR, dll. Fokus utama area studi literatur meliputi aplikasi teknologi AI pada kesehatan mental, RAG dan *Knowledge Graph*. Setelah itu, dilakukan analisis mendalam mengenai metode *Knowledge Extraction* dan *Knowledge Retrieval* yang digunakan.

3.3.3 Pengumpulan Data

Pemilihan sumber data didasarkan pada serangkaian kriteria validasi yang telah ditetapkan demi menjaga kualitas sumber pengetahuan. Kriteria ini bertujuan untuk menyaring informasi dan memastikan hanya konten yang berkualitas tinggi yang akan diintegrasikan ke dalam sistem. Prinsip utama pemilihan data meliputi kredibilitas dan otoritas lembaga penyusun serta relevansinya terhadap isu kesehatan mental. Sumber data harus berasal dari lembaga atau institusi yang memiliki otoritas dan rekam jejak yang diakui secara nasional dalam bidang kesehatan dan psikologi, seperti Kementerian Kesehatan, universitas, psikolog, dan lembaga lain. Selain itu, konten yang terkandung harus relevan dengan permasalah kesehatan mental yang ada, berikut dengan fasilitas yang tersedia, pencegahan, edukasi, dsb. Mengingat target implementasi sistem ini adalah untuk civitas akademika UGM, maka prioritas diberikan pada sumber data yang tidak hanya relevan untuk konteks kesehatan mental secara umum, tetapi juga spesifik dan dekat dengan lingkungan UGM. Penggunaan sumber internal ini bertujuan untuk meningkatkan kepercayaan, kedekatan, dan relevansi informasi bagi pengguna akhir.

Berdasarkan kriteria yang ditetapkan, korpus data untuk penelitian ini dikumpulkan dari beberapa sumber, antara lain:

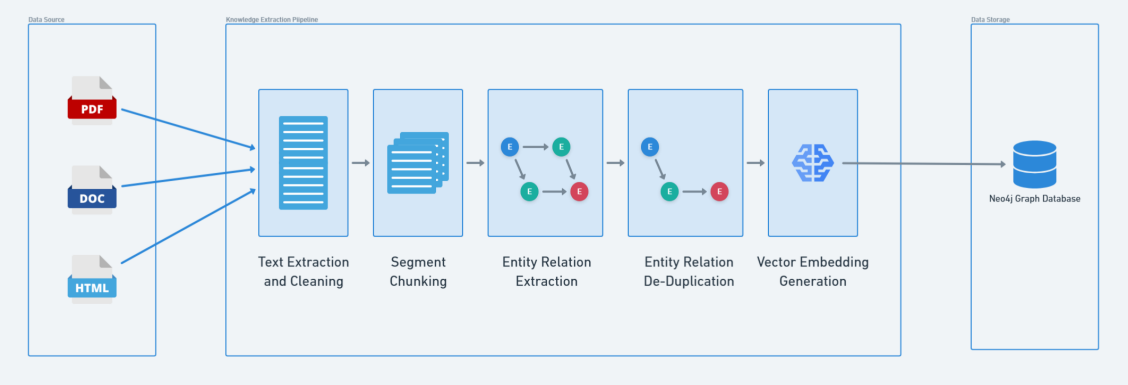
1. **Kementerian Kesehatan (Kemenkes) RI**: Dokumen yang dipublikasikan oleh Kemenkes digunakan sebagai pedoman dasar yang berlaku secara nasional, memberikan pengetahuan mengengai kesehatan mental secara umum di Indonesia.
2. ***Center for Public Mental Health* (CPMH) Fakultas Psikologi UGM**: Sebagai pusat studi kesehatan mental, publikasi dari CPMH tidak hanya valid secara akademis,

tetapi juga sangat relevan bagi civitas akademika UGM.

3. ***Health Promoting University* (HPU) UGM**: Materi dari HPU UGM dipilih karena secara spesifik dirancang untuk mempromosikan kesehatan di lingkungan kampus UGM. Informasi dari sumber ini, seperti mengenai layanan dukungan yang tersedia di UGM, akan sangat praktis dan langsung dapat dimanfaatkan oleh pengguna.

Data yang terkumpul akan melalui proses kurasi untuk memastikan kesesuaian konten dan konsistensi informasi. Setelah divalidasi, data kemudian akan melalui tahap prapemrosesan untuk membersihkan teks dan menyiapkannya untuk fase pengembangan arsitektur.

3.3.4 Pengembangan Arsitektur Ekstraksi dan Penyerapan Pengetahuan



Gambar 3.3. *knowledge extraction pipeline*

Pengembangan model akan berfokus pada optimalisasi dua komponen krusial pada RAG berbasis *Knowledge Graph*, yaitu *Knowledge Extraction* dan *Knowledge Retrieval*. Tahap ini bertujuan untuk membangun *Knowledge Graph* yang bersih, konsisten, dan kaya informasi dari berbagai sumber data yang mayoritas tidak terstruktur seperti PDF, DOCX, dan situs web. Proses ekstraksi pengetahuan diimplementasikan melalui sebuah *pipeline* seperti pada Gambar 3.3. Berikut merupakan penjelasan detail setiap langkah dalam *pipeline*.

3.3.4.1 Ekstraksi Teks dari Dokumen

Hampir semua *language model* dapat memahami informasi dalam bentuk teks. Untuk itu, langkah awal membangun sebuah *Knowledge Graph* adalah mengekstrak teks mentah dari berbagai sumber data. Banyak data memiliki format, baik ekstensi, maupun struktur yang beragam karena berasal dari banyak sumber. Setiap tipe dokumen memiliki cara mengekstraknya masing-masing, misalnya dokumen HTML dapat dikenali strukturnya melalui tagar HTML seperti `<h1>` untuk menyatakan judul dan `<p>` untuk

menyatakan paragraf. Dokumen bertipe DOCX terdiri dari kumpulan *file* ZIP yang harus diekstrak untuk mengambil isi dokumen. Setiap dokumen tersebut juga memiliki struktur penulisan yang tidak seragam, seperti pada dokumen HTML belum tentu mengandung tagar *headline* (`<h1>`, `<h2>`, sampai `<h6>`) untuk menulis, melainkan memakai tagar `<div>`. Ketakseragaman format tersebut menjadi tantangan tersendiri dalam mengekstrak teks dari dokumen yang beragam.

Dokumen yang digunakan dalam membangun KG meliputi.

1. Buku "Petunjuk Teknis Pencegahan dan Pengendalian Gangguan Mental Emosional" dari Kementerian Kesehatan.
2. Buku "Panduan Kesehatan Jiwa pada Masa Pandemi COVID-19" dari Kementerian Kesehatan.
3. Buku "Panduan Pertolongan Pertama Psikologis Pada Upaya Bunuh Diri" dari CPMH.
4. Artikel dari situs web CPMH
5. Artikel dari situs web HPU UGM

Sebagian besar dokumen tersebut berada dalam format PDF, sedangkan artikel dari situs web CPMH dan HPU UGM berada dalam format HTML. Mengekstrak *file* PDF dengan mempertahankan struktur dokumen cukup sulit untuk dilakukan secara langsung karena PDF merupakan tipe dokumen berbasis halaman (*page*) yang mengabaikan struktur keseluruhan seperti judul, paragraf, atau tabel. Informasi mengenai struktur dokumen penting dijaga untuk langkah selanjutnya. Untuk itu, dilakukan strategi konversi *file* PDF ke dalam bentuk dokumen yang lebih terstruktur seperti DOCX dan HTML. Konversi dilakukan menggunakan layanan konversi file online FreeConvert. Setelah dilakukan konversi dokumen PDF menjadi DOCX barulah isi dari dokumen dapat diambil. Ekstraksi teks mentah dari dokumen dilakukan menggunakan pustaka python-docx dengan mengambil semua teks yang ada beserta dengan tabel. Sementara itu, dokumen artikel dari situs web CPMH dan HPU UGM berformat HTML, sehingga proses ekstraksinya lebih sederhana. Ekstraksi dari HTML lebih mudah karena format ini sudah memiliki struktur bawaan melalui penggunaan tag. Misalnya, tag `<h1>` digunakan untuk judul, `<p>` untuk paragraf, dan `<table>` untuk tabel. Untuk merepresentasikan tabel dalam bentuk teks mentah, digunakan format tabel markdown yang intuitif dan mudah dipahami oleh sebagian besar LLM.

3.3.4.2 *Structural Chunking*

Teks yang diekstrak dari dokumen pada langkah selanjutnya akan diekstrak lagi entitas dan relasi yang terkandung di dalamnya. Ekstraksi entitas dan relasi dilakukan dengan menggunakan LLM. Namun, LLM modern memiliki ukuran *context window* yang terbatas berkisar ratusan ribu (GPT 4o) hingga 10 juta token (Meta Llama 4 Scout). Ke-

terbatasan tersebut berimplikasi pada ukuran dokumen (dalam satuan token) menjadi hal yang perlu diperhatikan mengingat sebuah dokumen dapat memiliki ukuran yang besar. Analisis menggunakan LLM dengan dokumen yang melebihi atau mendekati ukuran maksimal *context window* tidak akan maksimal karena LLM gagal memahami konteks dokumen secara keseluruhan. Untuk itu, dokumen perlu dipecah menjadi beberapa bagian yang disebut *chunk*.

Proses *chunking* secara umum dapat dilakukan dengan memecah dokumen menjadi beberapa *chunk* dengan panjang tertentu. Metode *chunking* seperti itu cukup mudah dilakukan, tetapi ada potensi bagian kata atau kalimat dapat terpisah antar *chunk* akibat pemotongan yang menyebabkan kehilangan makna dan konteks. Dokumen yang masih memiliki informasi strukturnya dapat dimanfaatkan untuk melakukan pemecahan dokumen. Alih-alih memotong dokumen menjadi potongan berukuran tetap, dokumen dibagi berdasarkan format struktural tertentu, misal dalam dokumen HTML berupa tag `<h1>` atau pada dokumen DOCX berupa *style paragraph heading* atau *title*. Pada buku "Petunjuk Teknis Pencegahan dan Pengendalian Gangguan Mental Emosional" yang telah dikonversi menjadi DOCX, dipecah menjadi beberapa chunk berdasarkan judul-judul bagian (seperti BAB I, BAB II, dst). Pemecahan dokumen berdasarkan judul bagian menghasilkan pecahan dokumen yang memiliki konteks utuh karena pada umumnya setiap bagian dalam dokumen menjelaskan suatu topik bahasan secara utuh. Hasil dari pemecahan dokumen tersebut yang berjumlah 13 *chunk* kemudian disimpan dalam format TXT untuk kebutuhan *tracing*. Untuk dokumen yang tidak terlampau panjang tidak akan dipecah, tetapi langsung menjadi 1 bagian dokumen utuh.

3.3.4.3 Ekstraksi Entitas dan Relasi

Ekstraksi entitas dan relasi dalam penelitian ini dilakukan melalui mekanisme inferensi dokumen menggunakan LLM. Penggunaan LLM untuk mengekstrak entitas dan relasi didasarkan pada kemampuannya dalam memahami struktur dan konteks dokumen yang kompleks dan berukuran besar berkat arsitekturnya yang dilengkapi dengan jumlah parameter yang sangat besar hingga triliunan parameter. Kapasitas ini memungkinkan LLM untuk menangkap pola linguistik, semantik, dan hubungan antar entitas secara lebih mendalam. LLM yang digunakan sebagai basis ekstraksi entitas dan relasi adalah Google Gemini 2.5 Flash. Pemilihan model tersebut didasarkan pada performanya yang sangat baik dan masuk dalam *leaderboard* paling atas pada LLM Stats setidaknya saat penelitian ini dilakukan dengan skor GPQA (*Graduate-Level Google-Proof Question and Answer*) sebesar 82,8%. Faktor krusial lainnya yang tidak kalah penting adalah Gemini 2.5 Flash memiliki ukuran *context window* yang sangat besar hingga 1 juta dibandingkan model LLM lain seperti Claude 3.7 Sonet, Grok-3, dan GPT o3. Perbandingan performa dan kapasitas LLM tersebut dapat dilihat pada Tabel 3.1.

Tabel 3.1. Perbandingan performa dan kapasitas LLM modern

Model	**GPQA**	***Context Window***
Gemini 2.5 Flash	82,8%	1.048.576
Claude Sonnet 4	83,8%	200.000
OpenAI o3	83,3%	200.000
Grok-3	84,6%	128.000

Setiap *chunk* teks dari dokumen kemudian diproses oleh Gemini 2.5 Flash untuk mengekstrak entitas dan relasi yang terkandung di dalamnya. Proses ini memanfaatkan teknik *few-shot prompting*, di mana LLM diberikan beberapa contoh konkret untuk memandunya dalam melakukan ekstraksi sesuai dengan ontologi yang telah didefinisikan. *Prompt* dirancang sejelas mungkin untuk menjelaskan apa yang seharusnya dilakukan oleh LLM dalam proses ekstraksi. Untuk menjaga jenis entitas yang ditemukan dalam dokumen tidak keluar dari tujuan layanan kesehatan mental maka beberapa istilah yang berkaitan telah didefinisikan terlebih dahulu dalam *prompt*. Format respons dari LLM juga didefinisikan berupa tipe data JSON yang memudahkan dalam pemrosesan selanjutnya. Format instruksi *prompt* dapat dilihat pada Gambar 3.4.

```
### INSTRUKSI:
Anda adalah sebuah pipeline ekstraksi Knowledge Graph untuk domain kesehatan mental.
Dari DOKUMEN yang diberikan, ekstrak semua entitas penting dan relasi di antara mereka.
Patuhi struktur JSON dan daftar tipe yang telah ditentukan dengan ketat.

### LANGKAH-LANGKAH:
1.  Ekstrak Entitas: Identifikasi entitas, klasifikasikan tipenya, dan berikan deskripsi singkat dari teks.
2.  Ekstrak Relasi: Identifikasi hubungan langsung antar entitas tersebut. `name` relasi harus berupa
    frasa kerja dari teks, dan `type` adalah kategorisasi formalnya.

### Sistem Tipe (Ontologi)

A. Tipe Entitas yang Diizinkan:
* Gangguan & Kondisi: `Gangguan_Mental`, `Gejala`, `Kondisi_Medis_Terkait`
* Intervensi & Perawatan: `Terapi_Psikologis`, `Obat_Medis`, `Layanan_Kesehatan`, `Aktivitas_Kesejahteraan`
* Aktor & Pemangku Kepentingan: `Profesional_Kesehatan`, `Organisasi`, `Individu`
* Faktor & Konteks: `Faktor_Risiko`, `Faktor_Pelindung`, `Dokumen_Hukum_Kebijakan`, `Lokasi`
* Data & Konsep: `Data_Statistik`, `Konsep_Abstrak`
* Jaring Pengaman: `Lainnya` (Gunakan untuk entitas penting yang tidak cocok dengan kategori lain)

B. Tipe Relasi yang Diizinkan:
* Sebab-Akibat: `Menyebabkan`, `Berkontribusi_Pada`, `Mengurangi_Risiko`, `Merupakan_Gejala_Dari`
* Intervensi: `Mendiagnosis`, `Menangani`, `Meresepkan`, `Menawarkan_Layanan`
* Hierarki & Keanggotaan: `Adalah_Jenis_Dari`, `Bekerja_Di`, `Berlokasi_Di`
* Deskriptif: `Memiliki_Atribut`, `Didasarkan_Pada`, `Direkomendasikan_Untuk`
* Jaring Pengaman: `Terkait_Dengan` (Gunakan untuk hubungan yang jelas tapi tidak cocok kategori lain)

### Struktur Output JSON
{
"entities": [{
    "name": "nama_entitas",
    "type": "tipe_entitas_dari_daftar_di_atas",
    "description": "penjelasan singkat atau kutipan dari dokumen"
    }
  ],
"relations": [{
    "source_entity": "nama_entitas_sumber",
    "target_entity": "nama_entitas_target",
    "name": "frasa_kerja_dari_dokumen",
    "type": "tipe_relasi_dari_daftar_di_atas"
    }
  ]
}
```

Gambar 3.4. Instruksi LLM untuk melakukan ekstraksi entitas dan relasi

Selain pembatasan istilah dan pemberian instruksi, teknik *few-shot prompting* juga digunakan untuk menghasilkan respons sesuai dengan apa yang diinginkan. *Few-shot prompting* telah teruji memberikan peningkatan yang signifikan terhadap performa LLM dibandingkan dengan *zero-shot prompting* yang hanya memberikan instruksi saja tanpa contoh [16]. Teknik ini dilakukan dengan memberikan beberapa contoh (*shot*) input yang akan dihadapi beserta respons yang diharapkan. Sebuah potongan dari dokumen ditambahkan sebagai contoh input dokumen yang akan diekstrak diikuti dengan keluaran yang diharapkan menggunakan format JSON. Contoh masukkan dan keluaran pada *prompt* dapat dilihat pada Gambar 3.5

```
### CONTOH EKSEKUSI

### DOKUMEN:
Kesehatan Jiwa adalah kondisi dimana seorang individu dapat berkembang secara fisik, mental, spiritual, dan sosial.
Karakteristik gangguan jiwa secara umum yaitu kombinasi pikiran yang abnormal, emosi, dan persepsi.
Faktor psikologis seperti trauma yang mengakibatkan stres dan faktor biologis seperti genetik merupakan faktor yang berkontribusi
terhadap terjadinya gangguan jiwa. Sebesar 50% gangguan jiwa berawal pada usia 14 tahun.

### JSON_OUTPUT:
{ "entities": [{
    "name": "Kesehatan Jiwa",
    "type": "Konsep_Abstrak",
    "description": "kondisi dimana seorang individu dapat berkembang secara fisik, mental, spiritual, dan sosial."},

    {"name": "Gangguan Jiwa",
    "type": "Gangguan_Mental",
    "description": "Sebuah kondisi yang memiliki karakteristik seperti kombinasi pikiran abnormal,
      dan dipengaruhi oleh faktor psikologis serta biologis."},

    ...],
  "relations": [{
    "source_entity": "Gangguan Jiwa",
    "target_entity": "Karakteristik Gangguan Jiwa",
    "name": "memiliki karakteristik",
    "type": "Memiliki_Atribut"},

    {"source_entity": "Faktor Psikologis",
    "target_entity": "Stres",
    "name": "mengakibatkan",
    "type": "Menyebabkan"},

    ...]}
```

Gambar 3.5. Penggunaan *few-shot prompting* dengan memberikan potongan dokumen dan keluaran yang diharapkan.

Prompt pada Gambar 3.4 dan Gambar 3.5 kemudian dieksekusi menggunakan pustaka google genai. Eksekusi *prompt* diikuti dengan data *chunk* dokumen ditambah dengan konfigurasi struktur JSON yang diinginkan untuk mendapatkan respons bukan dalam bentuk teks, melainkan dalam bentuk JSON yang dapat direpresentasikan dalam bentuk objek pydantic. Struktur dari respons terbagi menjadi 2 bagian yaitu `entities` yang berisi daftar entitas dan `relations` yang berisi daftar relasi. Setiap entitas memiliki atribut `name`, `type`, dan `description`, sedangkan relasi memiliki atribut `source_entity`, `target_entity`, `name`, dan `type` seperti pada Gambar 3.6.

```python
class Entity(BaseModel):
    id: Optional[str] = None
    chunk_id: Optional[str] = None
    name: str
    type: Optional[str] = None
    description: str
    embedding: Optional[list[float]] = None

class Relation(BaseModel):
    id: Optional[str] = None
    chunk_id: Optional[str] = None
    source_entity: str
    target_entity: str
    name: str
    type: str

class EntityRelation(BaseModel):
    entities: List[Entity]
    relations: List[Relation]
```

Gambar 3.6. Struktur entitas dan relasi yang didefinisikan sebagai *objek* pada pustaka Pydantic

Hasil dari eksekusi *prompt* menghasilkan kumpulan entitas dan relasi dengan format persis seperti skema yang didefinisikan sebelumnya. Gambar 3.8 menunjukkan entitas dan relasi yang diekstrak dari potongan dokumen pada Gambar 3.7.

```
Gangguan mental emosional adalah suatu kondisi yang mengindikasikan seseorang mengalami perubahan psikologis yang mungkin
merupakan sebuah kondisi normal, tetapi dapat juga merupakan kondisi patologis. Istilah gangguan mental emosional mengacu
pada istilah yang digunakan pada survei kesehatan rumah tangga (SKRT) tahun 1995. Beberapa survei di negara lain menggunakan
istilah gangguan mental emosional merujuk pada istilah "distress psikologik" atau "mental distress" (Abbo et al, 2008; Gelaye et al, 2012).
Penggunaan istilah gangguan mental emosional sebagai salah satu indikator kondisi kesehatan jiwa penduduk Indonesia lebih banyak
bertujuan agar masyarakat serta para akademisi dan professional mengerti bahwa alat ukur yang digunakan adalah sama dengan survei
sebelumnya (Balitbangkes, 2014).
```

Gambar 3.7. Cuplikan dokumen yang diambil dari Buku Petunjuk Teknis Pencegahan dan Pengendalian Gangguan Mental Emosional

```
{
  "entities": [
    {
      "name": "Gangguan mental emosional",
      "type": "Gangguan_Mental",
      "description": "Suatu kondisi yang mengindikasikan seseorang mengalami perubahan psikologis,
        yang mungkin merupakan kondisi normal atau patologis."
    },
    {
      "name": "Perubahan psikologis",
      "type": "Gejala",
      "description": "Kondisi yang dialami seseorang yang mungkin mengindikasikan gangguan mental emosional,
        bisa normal atau patologis."
    },
    {
      "name": "Kondisi normal",
      "type": "Konsep_Abstrak",
      "description": "Kondisi psikologis yang tidak mengindikasikan adanya patologi."
    },
    {
      "name": "Kondisi patologis",
      "type": "Konsep_Abstrak",
      "description": "Kondisi psikologis yang mengindikasikan adanya penyakit atau gangguan."
    }, ... ],
  "relations": [
    {
      "source_entity": "Gangguan mental emosional",
      "target_entity": "Perubahan psikologis",
      "name": "mengindikasikan adanya",
      "type": "Memiliki_Atribut"
    },
    {
      "source_entity": "Perubahan psikologis",
      "target_entity": "Kondisi normal",
      "name": "merupakan",
      "type": "Adalah_Jenis_Dari"
    },
    {
      "source_entity": "Perubahan psikologis",
      "target_entity": "Kondisi patologis",
      "name": "merupakan",
      "type": "Adalah_Jenis_Dari"
    },
    {
      "source_entity": "Gangguan mental emosional",
      "target_entity": "Survei kesehatan rumah tangga (SKRT) tahun 1995",
      "name": "digunakan dalam",
      "type": "Terkait_Dengan"
    }, ... ]
}
```


Gambar 3.8. Respons LLM berupa entitas dan relasi yang diekstrak dari potongan dokumen

3.3.4.4 Resolusi Entitas dan Relasi

Hasil dari entitas dan relasi yang telah berhasil diekstrak dari setiap *chunk* kemudian diagregasi dan melalui tahap resolusi untuk memastikan konsistensi. Entitas dan relasi yang berasal dari *chunk* berbeda mungkin akan memiliki duplikat karena diekstrak secara independen. Untuk itu, perlu untuk melakukan deduplikasi pada entitas dan relasi guna menghindari redudansi yang tidak perlu. Entitas yang redundan dapat membuat graf semakin kompleks dan merusak mekanisme *retrieval*. Deduplikasi dilakukan dengan strategi penggabungan di mana entitas yang memiliki nama yang sama setelah dinormalisasi akan digabungkan menjadi satu. Untuk mengatasi kehilangan informasi maka deskripsi unik dari entitas duplikat kemudian digabungkan menjadi satu deskripsi utuh. *Entity mapping* dibentuk saat proses deduplikasi entitas untuk memetakan `source_entity` dan `target_entity` pada relasi ke entitas yang valid.

3.3.4.5 Penambahan Vektor *Embedding*

Data entitas dan relasi yang disimpan dalam basis data secara umum adalah kumpulan teks yang cukup intuitif bagi manusia untuk memahaminya. Meskipun demikian, komputer tidak dapat secara langsung "memahami" makna dari tiap-tiap entitas dan relasi. Bagi manusia mungkin akan mudah untuk mencari entitas dengan nama atau deskripsi yang berkaitan dengan kata kunci atau pertanyaan pengguna. Untuk itu, diperlukan sebuah representasi tertentu dari data yang dapat dengan mudah dipahami maknanya oleh komputer. Salah satu representasi data yang cukup mudah diolah oleh komputer sekaligus dapat menangkap makna semantik dari data adalah vektor *embedding*. Vektor *embedding* merupakan sebuah vektor berdimensi tinggi yang umum digunakan untuk menangkap makna dari sebuah data. Data dalam bentuk vektor memiliki keuntungan dalam melakukan pencarian semantik (*semantic search*) dengan hanya melakukan operasi *dot product* 2 vektor dapat mengetahui tingkat kemiripan antara 2 data.

Dalam penelitian ini, objek yang akan dilakukan pencarian semantik dalam prosen *knowledge retrieval* adalah entitas. Untuk itu, setiap entitas akan memiliki representasi vektornya masing-masing. Vektor *embedding* akan dibentuk menggunakan model gemini-embedding-001. Model *embedding* ini dipilih menjadi model *embedding* karena memiliki performa yang sangat baik dalam tugas *Semantic Text Similarity* (STS) sebesar 79,40 poin. Selain itu, model ini juga memiliki kemampuan memahami teks dalam bahasa Indonesia paling baik (67,00 poin) dibandingkan dengan model-model lain [1]. Perbandingan model *embedding* dapat dilihat pada Tabel 3.2.

Tabel 3.2. Perbandingan model *embedding* dalam tugas *Semantic Text Similarity* (STS) dan klasifikasi dalam bahasa Indonesia [1]

Model	**IndoneisanIdClickbaitClassification**	**STS**
gemini-embedding-001	67,00	79,40
Qwen3-Embedding-8B	65,07	81.08
Qwen3-Embedding-0,6B	64,77	76,17
Qwen3-Embedding-4B	64,29	80,86
voyage-multiligual-2	63,23	68,58

Data yang digunakan untuk membuat vektor *embedding* berasal dari gabungan nama, tipe, dan deskripsi dari setiap entitas. Proses pembuatannya dilakukan secara berulang dengan 100 entitas setiap satu kali pemanggilan API gemini-embedding-001. Pemanggilan tersebut menghasilkan sebuah vektor dengan 768 dimensi untuk setiap entitas. Hasil yang berupa vektor tersebut kemudian digabungkan dengan entitas yang bersesuaian dengan atribut `embedding`.

3.3.4.6 Pemuatan Data ke Penyimpanan Basis Data Berbasis Graf

Entitas dan relasi yang telah dilengkapi dengan vektor *embedding* kemudian dimuat ke dalam basis data graf. Basis data graf yang dipilih adalah Neo4j karena kemampuannya dalam menyimpan dan menavigasi struktur relasi antar entitas secara efisien, serta mendukung kueri berbasis graf melalui bahasa Cypher. Neo4j juga mendukung algoritma pencarian berbasis vektor seperti *Approximate Nearest Neighbor* (ANN) secara *native* dengan menerapkan algoritma *Hierarchical Navigable Small World* (HNSW). Dukungan ini menjadi krusial untuk memungkinkan pencarian semantik dijalankan secara optimal. Selain itu, Neo4j merupakan sistem basis data yang telah matang dan banyak digunakan, sehingga memiliki dukungan komunitas yang kuat. Dalam pemodelan data, setiap entitas direpresentasikan sebagai *node* dengan atribut seperti `name`, `type`, dan `embedding`, sementara relasi dimodelkan sebagai *edge* yang menghubungkan dua *nodes* dan memiliki atribut seperti `source_entity`, `target_entity`, `name`, dan `type`. Proses pemuatan data ke Neo4j dilakukan dengan melakukan inisialisasi koneksi ke server, lalu menyusun kueri untuk menambahkan *node* dan *edge* sesuai skema. Untuk menghindari duplikasi *node*, digunakan pendekatan `MERGE` dalam Cypher, sehingga entitas dengan identitas yang sama tidak dibuat ulang seperti pada Gambar 3.9. Proses ini juga mempertimbangkan efisiensi, khususnya ketika jumlah entitas dan relasi sangat besar, dengan menerapkan strategi *batch insert*. Dengan memanfaatkan basis data graf, relasi antar entitas dapat ditelusuri secara lebih fleksibel dan intuitif. Hal ini juga mendukung berbagai analisis struktur data seperti pencarian pola hubungan, pengukuran kedekatan antar *node*, hingga identifikasi komunitas dalam graf.

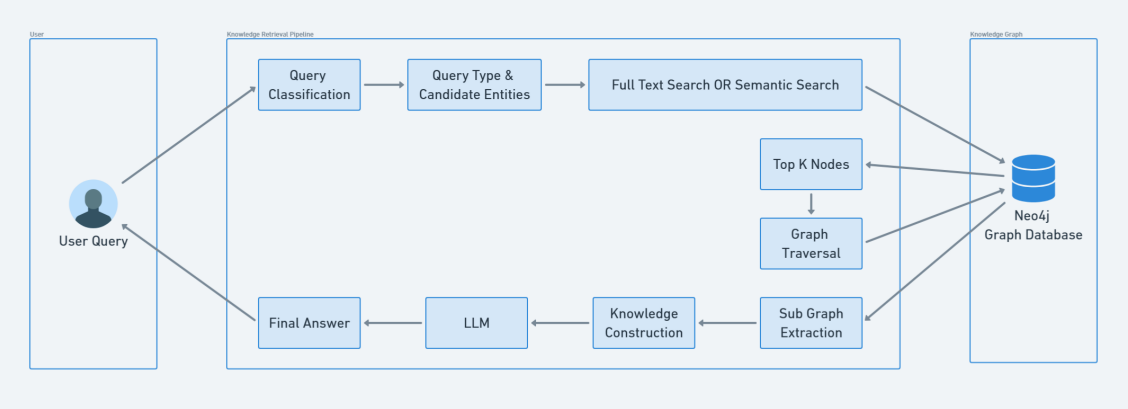
```
UNWIND $data AS row
CALL apoc.create.node([row.type, 'Entity'], {
    name: row.name,
    id: row.id,
    chunk_id: row.chunk_id,
    description: row.description,
    embedding: row.embedding
    }) YIELD node
RETURN node
```

Gambar 3.9. Kode Cypher untuk pemuatan entitas yang berupa *nodes* di basis data Neo4j.

3.3.5 Pengembangan Arsitektur Pengambilan Pengetahuan

Pengambilan Pengetahuan (*Knowledge Retrieval*) merupakan proses penting dalam sistem berbasis pengetahuan yang bertujuan untuk memberikan jawaban yang rele-

van berdasarkan pertanyaan pengguna. *Pipeline* pengambilan pengetahuan yang dikembangkan memiliki beberapa komponen utama seperti klasifikasi kueri, pencarian semantik berbasis vektor dan kata kunci, penelusuran struktur graf, serta konstruksi pengetahuan seperti pada Gambar 3.10. Setiap komponen ini berkontribusi terhadap ketepatan dan relevansi hasil yang diberikan oleh sistem.



Gambar 3.10. *Pipeline Knowledge Retrieval*.

3.3.5.1 Klasifikasi Kueri dan Ekstraksi Entitas

Langkah pertama dalam pengambilan pengetahuan adalah klasifikasi kueri terhadap input pengguna. Proses ini bertujuan untuk mengidentifikasi intensi semantik dari kueri, yang nantinya akan dijadikan sebagai variabel penentu strategi pencarian dalam *Knowledge Graph* yang paling sesuai. Kueri akan diklasifikasikan menjadi dua kategori seperti pada Tabel 3.3

Tabel 3.3. Klasifikasi kueri menjadi dua kategori, yaitu `entity_query` dan `path_query`

Kategori	**Keterangan**
`entity_query`	Jenis kueri yang berfokus pada satu entitas utama dan bertujuan untuk mendeskripsikan, mengambil atribut, atau memahami konsep dari entitas tersebut. Kueri ini biasanya mengandung permintaan informasi faktual atau deskriptif.
`path_query`	Jenis kueri yang melibatkan dua atau lebih entitas, dan bertujuan untuk menelusuri hubungan atau alur keterkaitan antar entitas dalam graf. Kueri ini dapat digunakan untuk menemukan relasi eksplisit maupun implisit antara entitas.

Klasifikasi kueri ini sangat berkaitan dengan strategi pencarian yang akan dilakukan pada basis data graf. Sebuah kueri diklasifikasikan sebagai `entity_query` jika intensi utamanya adalah untuk mendeskripsikan atau melihat lebih detail pada satu atau beberapa konsep sentral. Pertanyaan dari pengguna seperti *"Apa itu kesehatan mental?"* atau *"Di manakah pusat layanan kesehatan mental di UGM?"* termasuk dalam kategori ini karena memiliki intensi untuk tahu lebih dalam pada suatu konsep yaitu Untuk menjawab pertanyaan tersebut, sistem hanya perlu menemukan *node "Kesehatan Mental"* dan *"Pusat Layanan Kesehatan Mental UGM"* di dalam graf dan mengambil informasi yang melekat langsung padanya (deskripsi) beserta tetangga-tetangga terdekat yang berfungsi sebagai atribut. Penelusuran ini bersifat melebar dari satu atau beberapa titik pusat.

Sebuah kueri diklasifikasikan sebagai `path_query` apabila intensi utamanya adalah memahami relasi, interaksi, atau alur proses antara dua atau lebih konsep yang berbeda. Pertanyaan dari pengguna seperti *"Bagaimana terapi kognitif Perilaku membantu mengatasi gangguan kecemasan?"* dan *"Apakah stres dapat mengakibatkan penyakit jantung?"*. Untuk menjawab pertanyaan tersebut, sistem tidak bisa hanya melihat masing-masing *nodes* dan tetangganya. Sistem harus bisa menemukan hubungan antara beberapa *nodes* lalu menelusuri jalur (*path*) yang menghubungkannya.

Klasifikasi kueri memiliki peran krusial untuk menentukan bagaimana sistem akan melakukan pencarian dalam basis data graf. Tidak seperti basis data relasional atau basis data lain yang memiliki skema terdefinisi, basis data graf dapat memiliki objek (*node*) yang saling terhubung satu sama lain tanpa terikat pada skema tertentu. Pencarian yang buruk tidak akan menghasilkan informasi yang tepat sasaran terhadap kueri pengguna. Untuk itu strategi pencarian menjadi hal yang signifikan terhadap kualitas pengambilan pengetahuan.

Selain klasifikasi kueri, sistem akan mengekstraksi kandidat entitas yang terkandung dalam kueri. Proses ini bertujuan untuk mengidentifikasi elemen-elemen penting dalam kueri yang berpotensi merujuk pada *node* dalam basis data graf. Ekstraksi entitas ini penting dilakukan untuk memberikan petunjuk saat melakukan pencarian dalam basis data graf. Kandidat entitas yang berasal dari kueri pengguna akan dijadikan sebagai kata kunci dalam melakukan pencarian di samping pencarian berbasis vektor.

Proses klasifikasi kueri dan ekstraksi entitas dilakukan secara simultan menggunakan pendekatan berbasis LLM, yang mampu memahami konteks linguistik untuk mengidentifikasi istilah seperti nama penyakit, gejala, proses, atau entitas lainnya. *Prompt* disusun dengan memberikan instruksi, definisi, dan format keluaran yang diharapkan. Selain itu, *few-shot prompt* digunakan untuk memberikan contoh input dan keluaran yang diharapkan pada LLM untuk mendapatkan respons yang diinginkan seperti pada Gambar 3.11.

```
"""
Tugas: Klasifikasikan kueri pengguna ke dalam kategori 'entity_query' atau 'path_query', dan ekstrak semua entitas yang relevan.

Definisi:
'entity_query' (Query Atribut/Node): Pertanyaan yang fokus untuk mendeskripsikan atau mengambil properti/atribut dari sebuah konsep sentral.
  Contoh: "Apa itu X?", "Apa saja gejala dari X?".
'path_query' (Query Relasi/Edge): Pertanyaan tentang hubungan antara dua atau lebih entitas, atau mencari alur/proses.
  Contoh: "Bagaimana hubungan A dan B?", "Cari A yang terkait dengan B?".

Output Format (JSON):
{{
"category": "entity_query" | "path_query",
"entities": ["Entity 1", "Entity 2", ..., "Entity N"]
}}

Contoh:

Query: "Apa itu depresi dan bagaimana gejalanya?"
Output: {{"category": "entity_query", "entities": ["depresi"]}}

Query: "Apa saja jenis obat antidepresan yang umum diresepkan?"
Output: {{"category": "entity_query", "entities": ["obat antidepresan"]}}

Query: "Bagaimana cara kerja terapi kognitif-perilaku (CBT) untuk kecemasan?"
Output: {{"category": "path_query", "entities": ["terapi kognitif-perilaku (CBT)", "kecemasan"]}}

Query: "Jelaskan peran serotonin dalam gangguan mood."
Output: {{"category": "path_query", "entities": ["serotonin", "gangguan mood"]}}

Query: "Bisakah olahraga membantu mengurangi stres dan meningkatkan kesehatan mental?"
Output: {{"category": "path_query", "entities": ["olahraga", "stres", "kesehatan mental"]}}

Query: "Layanan dukungan kesehatan mental apa saja yang tersedia untuk remaja di Jakarta?"
Output: {{"category": "path_query", "entities": ["layanan dukungan kesehatan mental", "remaja", "Jakarta"]}}

Query: "Siapa psikolog yang spesialisasi di terapi trauma?"
Output: {{"category": "path_query", "entities": ["psikolog", "terapi trauma"]}}

---
Klasifikasikan dan ekstrak entitas untuk kueri berikut:

Query:
{query}
"""
```

Gambar 3.11. *Prompt* yang digunakan untuk memberikan instruksi LLM dalam tugas klasifikasi kueri dan ekstraksi entitas

3.3.5.2 *Hybrid Search*

Setelah diekstrak dari kueri pengguna pada proses sebelumnya, kandidat entitas menjadi basis untuk melakukan pencarian entitas dalam graf. Proses pencarian ini bertujuan untuk mencari *nodes* pada basis data graf yang berkaitan dengan kueri pengguna. Pencarian didasarkan pada dua variabel utama, yaitu kandidat entitas yang telah diekstrak pada proses sebelumnya dan kueri pengguna itu sendiri. Tahap pencarian dalam arsitektur ini menggunakan pendekatan *hybrid search* yang merupakan kombinasi dua strategi pencarian, yaitu *full-text search* dan *semantic search* berbasis vektor. Kedua metode ini dijalankan secara berurutan dengan strategi *fallback* untuk meningkatkan efektivitas pencarian entitas dalam basis data graf.

Proses dimulai dengan melakukan pencarian entitas menggunakan *full-text search* di Neo4j. Kata kunci yang digunakan berasal dari hasil ekstraksi entitas kandidat pada tahap sebelumnya. *Full-text search* memanfaatkan indeks teks untuk mencocokkan nama *node* secara eksplisit dalam basis data graf. Pendekatan ini efektif untuk kueri yang mengandung istilah eksplisit yang cocok dengan nama entitas.

Apabila pencarian berbasis teks tidak menghasilkan entitas yang relevan, misal-

nya karena variasi bahasa, sinonim, atau struktur kalimat yang tidak eksplisit, maka sistem akan secara otomatis melakukan *fallback* ke pencarian semantik. Pada tahap ini, sistem mengubah kueri pengguna menjadi vektor *embedding*, kemudian mencari entitas dengan representasi vektor yang paling mirip menggunakan algoritma *Approximate Nearest Neighbor* (ANN) yang didukung secara *native* di Neo4j, yaitu HNSW (*Hierarchical Navigable Small World*).

Dari kedua metode tersebut, sistem akan memperoleh sejumlah entitas paling relevan, yaitu Top-K entitas, yang kemudian digunakan sebagai titik awal untuk proses penelusuran graf. Pendekatan *hybrid* ini memungkinkan sistem menangani berbagai variasi bentuk pertanyaan, baik yang eksplisit, maupun implisit, serta meningkatkan cakupan pencarian dalam skenario real yang tidak dapat diprediksi.

3.3.5.3 Penelusuran Graf

Setelah diperoleh dari pencarian pada tahap sebelumnya, entitas dan jenis kueri digunakan untuk melakukan penelusuran graf (*graph traversal*) untuk mengumpulkan informasi kontekstual yang lebih komprehensif dari struktur graf. Tahap ini bertujuan menghasilkan subgraf yang relevan sebagai dasar bagi penyusunan jawaban oleh sistem. Penelusuran graf dilakukan dengan strategi yang disesuaikan dengan jenis kueri yaitu `entity_query` dan `path_query`. Algoritma n-hop neighbor expansion akan digunakan untuk menelusuri graf pada tipe kueri `entity_query`, sedangkan algoritma top k shortest path akan digunakan pada tipe kueri `path_query`. Penjelasan lebih lanjut mengenai dua jenis penelusuran graf terdapat pada Tabel 3.4.

Tabel 3.4. Klasifikasi kueri menjadi dua kategori, yaitu `entity_query` dan `path_query`

Graph Traversal	**Keterangan**
N-hop Neighbor Expansion	Untuk kueri bertipe `entity_query`, sistem menerapkan metode N-hop *Breadth-First Search* (BFS)-style *Neighbor Expansion* dari *node* pusat. Dalam sistem ini, nilai N ditetapkan sebagai 1, sehingga hanya *node* tetangga langsung (*one-hop neighbors*) yang diambil. Pendekatan ini efektif untuk menangkap atribut langsung dan relasi langsung dari suatu entitas, seperti definisi, komponen, atau karakteristik. Proses ini dilakukan dengan *query* Cypher untuk mengambil semua *node* dan *edge* yang terhubung langsung dengan entitas tersebut.

Graph Traversal	Keterangan
Top-K Shortest Paths	Untuk kueri bertipe `path_query`, sistem menggunakan pendekatan Top-K *Shortest Path Traversal*, yaitu menelusuri jalur terpendek antara setiap dua entitas kandidat yang dihasilkan dari proses ekstraksi entitas. Jalur ini dihitung berdasarkan jumlah *edge* dalam graf, dengan mempertimbangkan arah dan tipe relasi. Pendekatan ini digunakan untuk menemukan hubungan langsung maupun tidak langsung antar entitas, serta mengidentifikasi pola keterkaitan yang bermakna. Sistem dapat membatasi pencarian hingga K jalur terpendek untuk menjaga efisiensi dan relevansi hasil.

3.3.5.4 Transformasi Konteks

Transformasi konteks merupakan tahap penting yang menjembatani antara hasil penelusuran graf dan proses penyusunan jawaban oleh *Large Language Model* (LLM). Tahap ini bertujuan untuk mengubah representasi subgraf hasil penelusuran yang masih bersifat struktural menjadi format teks naratif yang dapat dipahami oleh LLM secara lebih efektif.

Secara teknis, transformasi dilakukan terhadap hasil kueri Cypher yang dikembalikan dalam bentuk struktur data bersarang (*nested*). Subgraf ini berisi *nodes* entitas beserta relasi antar entitas, termasuk atribut seperti tipe, label, dan deskripsi. Untuk memungkinkan LLM memproses informasi ini dengan baik, sistem mengubahnya menjadi representasi tekstual menggunakan format blok entitas beranotasi berbasis Markdown. Format ini mempertahankan struktur semantik penting, seperti tipe entitas, arah relasi, dan deskripsi yang menyertainya.

Alasan utama transformasi ini diperlukan adalah karena LLM bekerja optimal dengan masukan berbasis teks. Di samping itu, apabila hasil penelusuran graf digunakan secara langsung, akan banyak *overhead* informasi yang sama sekali tidak diperlukan seperti nama atribut yang justru membebani LLM karena *prompt* yang lebih panjang. Representasi berbentuk teks yang kaya konteks memungkinkan LLM untuk memahami hubungan antar entitas secara utuh, berbeda dengan daftar kalimat terpisah atau output tabel sederhana. Dengan struktur yang jelas dan eksplisit, LLM dapat melakukan *reasoning* dan menjawab pertanyaan pengguna secara lebih akurat.

Implementasi dari transformasi konteks ini dilakukan melalui fungsi Python yang secara iteratif memproses hasil Cypher *query* dan mengorganisasikannya ke dalam blok-blok informasi yang terstruktur, mirip dengan format dokumentasi. Dengan pendekatan

ini, subgraf dapat diinterpretasikan oleh LLM sebagai narasi pengetahuan, bukan sekadar data mentah, sehingga meningkatkan relevansi dan presisi jawaban yang dihasilkan.

3.3.5.5 Respons Final

Tahap akhir dari arsitektur pengambilan pengetahuan adalah penyusunan jawaban akhir (*final answer*) yang dikembalikan kepada pengguna. Pada tahap ini, representasi kontekstual yang telah ditransformasikan dari subgraf sebelumnya diproses oleh LLM untuk membentuk jawaban dalam bentuk teks alami yang informatif, ringkas, dan relevan terhadap kueri pengguna.

LLM digunakan sebagai komponen generatif yang memanfaatkan informasi terstruktur yang telah diformat menjadi teks. Dengan input yang kaya konteks, termasuk anotasi entitas, deskripsi, dan relasi antar *node*-LLM dapat melakukan *reasoning* semantik untuk menghasilkan jawaban yang tidak hanya berbasis fakta, tetapi juga disampaikan dalam bahasa yang mudah dipahami.

Sistem dapat mengatur gaya dan format jawaban yang dihasilkan, seperti bahasa formal atau santai sesuai kebutuhan aplikasi, panjang jawaban (ringkas vs. naratif), dan penyisipan kutipan entitas atau sumber dari graf, jika diperlukan. Proses ini dilakukan secara deterministik (menggunakan *prompt engineering* dengan temperatur rendah) untuk memastikan konsistensi dan menghindari halusinasi jawaban yang tidak berdasarkan pada data yang tersedia dalam graf. Dengan pendekatan ini, sistem tidak hanya berfungsi sebagai pencari informasi (*retriever*), tetapi juga sebagai penyaji pengetahuan yang mampu menyusun informasi menjadi jawaban utuh yang siap dikonsumsi pengguna.

3.3.6 Evaluasi dan Analisis

Tahapan evaluasi dan analisis merupakan fase krusial untuk mengukur performa sistem *Retrieval-Augmented Generation* (RAG) berbasis *Knowledge Graph* yang telah dibangun. Sistem RAG ini terdiri dari dua komponen utama, yaitu sistem *retrieval* yang bertugas mengambil konteks yang relevan dengan pertanyaan dan sistem *generation* yang menghasilkan respons akhir. Untuk itu, evaluasi dilakukan pada kedua komponen tersebut.

3.3.6.1 Penyiapan *Dataset* Evaluasi

Salah satu tantangan dalam mengevaluasi sistem RAG adalah ketiadaan *dataset* percakapan yang secara khusus sesuai dengan domain dan karakteristik dokumen yang digunakan. Tidak seperti metode *fine-tuning* yang sejak awal menggunakan data percakapan, RAG cukup memerlukan dokumennya saja. Untuk menjawab tantangan tersebut, dilakukan pembuatan *dataset* evaluasi sintesis mandiri yang seolah-olah meniru tanya

jawab dengan pengguna. Pendekatan ini dimaksudkan agar evaluasi cukup akurat mengukur kemampuan sistem dalam menjawab pertanyaan berdasarkan korpus pengetahuan yang telah diindeks.

Data percakapan untuk evaluasi dibuat berdasarkan dokumen yang dipakai untuk membangun *Knowledge Graph* yang berupa potongan-potongan dokumen (*document chunks*). Setiap potongan dokumen tersebut dianggap sebagai dasar kebenaran (*ground truth*) untuk menghasilkan beberapa *dataset* evaluasi. Selain potongan dokumen, *Knowledge Graph* juga berperan dalam membentuk *dataset* evaluasi, khususnya untuk evaluasi bagian sistem *retrieval*.

Sistem akan melakukan iterasi pada setiap potongan dokumen yang telah diindeks disertai dengan *nodes* dari subgraf yang berasosiasi sebagai konteks. Konteks tersebut kemudian dimasukkan ke dalam *Large Language Model* (LLM) dengan instruksi atau *prompt* yang telah dirancang. *Prompt* tersebut meminta LLM untuk berperan sebagai seorang ahli pembuatan data untuk menghasilkan *dataset* evaluasi yang terdiri dari beberapa *field* seperti pada Tabel 3.5.

Tabel 3.5. *Fields* pada *dataset* evaluasi

Fields	**Keterangan**
`query`	*Query* adalah sebuah pertanyaan yang jawabannya dapat ditemukan dari konteks yang disediakan.
`query_lable`	*Query lable* adalah kategori kueri yang nilainya dapat berupa *entity query* atau *path query*.
`golden_nodes`	*Golden nodes* adalah *nodes* yang berasal dari *knowledge graph* yang diperlukan untuk menjawab pertanyaan.
`golden_answer`	*Golden answer* merupakan jawaban akurat dari kueri dan didasarkan pada konteks yang tersedia.

Selain itu, teknik *few-shot* digunakan pada *prompt* untuk memberikan beberapa contoh data evaluasi yang diharapkan. Dalam setiap potongan dokumen akan dihasilkan pertanyaan setidaknya setengah dari jumlah *nodes* yang tersedia. Jumlah tersebut dipilih guna memberikan keleluasaan LLM untuk memberikan variasi pertanyaan dan tidak terpaku bahwa setiap *node* harus memiliki satu pertanyaan.

Prompt yang sudah dirancang kemudian dieksekusi dengan model OpenAI `o4-mini`. Pemilihan model dari OpenAI didasarkan pada model yang digunakan dalam proses evaluasi harus berbeda dengan model yang digunakan pada operasional. Hal tersebut dilakukan guna menghindari bias dalam evaluasi apabila model yang sama digunakan dalam menjawab pertanyaan sekaligus evaluasi. Model OpenAI `o4-mini` merupakan salah satu model kekinian yang dioptimalkan untuk tugas *reasoning* yang cepat. Kelebihan tersebut sangat sesuai dengan kebutuhan membuat *dataset* evaluasi berdasarkan instruksi.

3.3.6.2 Evaluasi Sistem *Retrieval*

Evaluasi sistem *retrieval* bertujuan untuk mencari tahu seberapa efektif komponen *retriever* dalam menghasilkan konteks yang bersesuaian. Konteks memiliki peranan yang krusial bagi komponen generatif dalam menghasilkan jawaban akhir yang memiliki landasan informasi akurat. Kualitas jawaban akhir yang dihasilkan secara signifikan dipengaruhi oleh relevansi dan kelengkapan konteks yang disediakan. Apabila *retriever* gagal menemukan konteks yang tepat, maka jawaban akhir akan berisiko tidak akurat bahkan cenderung menghasilkan "halusinasi", yaitu informasi yang tidak didasarkan pada data faktual. Halusinasi ini menjadi masalah besar terlebih apabila sistem ini akan diterapkan sebagai *chatbot* layanan kesehatan mental yang penggunanya sering berhubungan langsung dengan pasien yang memiliki risiko mental. Kesalahan informasi dapat mengakibatkan dampak yang fatal baik bagi penderita kesehatan mental ataupun bagi pengguna pada umumnya.

Dalam kasus RAG berbasis *Knowledge Graph*, konteks berarti subgraf yang dihasilkan oleh *retriever*. Setiap kali sistem menjawab pertanyaan, subgraf yang berkaitan dengan pertanyaan akan diambil dari basis data. Subgraf tersebut terdiri dari dua komponen yaitu *node* yang berisi informasi entitas dan *edge* yang berupa relasi antar entitas. Dalam proses evaluasi ini, hanya komponen nama *node* yang digunakan dengan asumsi bahwa *nodes* yang dihasilkan sudah mewakili konteks informasi. Hal tersebut dilakukan untuk menyederhanakan evaluasi dengan membandingkan *nodes* yang dihasilkan dengan *golden nodes* pada *dataset* evaluasi sebagai *ground truth*. Untuk setiap pertanyaan, sistem *retrieval* akan diminta untuk mengambil n subgraf yang relevan dengan pertanyaan. Bentuk dari subgraf terbagi menjadi dua kategori berdasarkan jenis pertanyaannya, yaitu *neighbor expansion* dan n *shortest path*. Masing-masing jenis subgraf tersebut memiliki struktur yang berbeda, tetapi tidak menjadi masalah karena hanya diambil nama *nodes* yang terkandung di dalamnya. *Nodes* tersebut kemudian dibandingkan dengan *golden nodes* untuk menghasilkan sejumlah metrik kuantitatif seperti pada Tabel 3.6 yang digunakan untuk mengukur efektivitas sistem *retrieval*.

Tabel 3.6. Metrik evaluasi sistem *retrieval*

Metrik	**Keterangan**
Precision	Presisi mengukur proporsi *nodes* yang relevan di antara *nodes* yang dihasilkan oleh *retriever*. Metrik ini menjawab pertanyaan *"Seberapa banyak dari hasil yang diberikan oleh sistem yang benar-benar berkaitan?"*.
Recall	*Recall* mengukur proporsi *nodes* relevan yang berhasil ditemukan dari semua *nodes* relevan yang tersedia. Metrik ini menjawab pertanyaan *"Seberapa lengkap cakupan hasil yang diberikan oleh sistem?"*.
Hit Ratio	*Hit Ratio* mengukur apakah setidaknya satu *node* yang relevan berada di antara k hasil teratas yang dikembalikan. Ini adalah metrik biner yang hanya peduli apakah jawaban yang benar ada di dalam daftar hasil, tidak peduli di peringkat berapa.
***Mean Reciprocal Rank* (MRR)**	*Mean Reciprocal Rank* (MRR) mengukur seberapa cepat sistem memprioritaskan *nodes* yang relevan. Metrik ini berfokus pada dokumen peringkat dari *node* relevan pertama yang ditemukan. Skor tinggi menunjukkan sistem memprioritaskan *node* relevan di peringkat atas.

3.3.6.3 Evaluasi Respons Akhir

Evaluasi respons akhir berfokus pada penilaian terhadap kualitas jawaban akhir yang dihasilkan sistem, khususnya pada bagian generatif setelah menerima konteks dari komponen *retrieval*. Evaluasi ini mempertimbangkan beberapa parameter penting, antara lain faktualitas, relevansi, dan kebenaran jawaban, guna memastikan bahwa respons akhir yang diberikan kepada pengguna memiliki nilai informatif yang tinggi dan sesuai dengan pertanyaan yang diajukan.

Keberhasilan sistem RAG tidak hanya ditentukan oleh kemampuan dalam mengambil konteks yang relevan, tetapi juga oleh kemampuan LLM dalam menghasilkan jawaban yang benar dan relevan berdasarkan konteks tersebut. Dalam beberapa kasus, bahkan jika konteks yang diambil sudah tepat, model generatif dapat tetap menghasilkan jawaban yang tidak faktual, tidak tepat sasaran, atau terlalu umum. Oleh karena itu, evaluasi terhadap respons akhir menjadi sangat penting untuk menilai efektivitas akhir sistem RAG secara holistik, mendeteksi adanya halusinasi, dan menjamin bahwa jawaban benar-benar menjawab pertanyaan pengguna secara akurat.

Evaluasi dilakukan dengan menggunakan *framework* RAGAS (*Retrieval-Augmented Generation Assessment*), yaitu sebuah kerangka kerja modern dan terstandardisasi yang dirancang khusus untuk mengevaluasi *pipeline* RAG secara menyeluruh. RAGAS akan menghitung sejumlah metrik evaluasi terhadap jawaban yang dihasilkan dengan membandingkannya terhadap *golden answer* dan konteks. Pendekatan ini memungkinkan evaluasi yang kuantitatif, objektif, dan replikatif. Metrik yang digunakan untuk menilai kualitas jawaban akhir terdapat pada Tabel 3.7.

Tabel 3.7. Metrik evaluasi sistem *Generation*

Metrik	**Keterangan**
Faithfulness	Mengukur sejauh mana jawaban yang dihasilkan tetap berpegang pada informasi dalam konteks. Skor tinggi menunjukkan bahwa jawaban tidak mengandung halusinasi.
Answe Relevancy	Menilai apakah jawaban benar-benar menjawab pertanyaan yang diajukan. Jawaban faktual tetapi tidak menjawab pertanyaan secara langsung akan memiliki skor rendah.
Answer Correctness	Membandingkan jawaban yang dihasilkan dengan *golden answer* untuk mengukur kesesuaian dan kebenaran faktual. Ini merupakan indikator langsung dari akurasi jawaban.
Answer Similarity	Mengukur kemiripan semantik antara jawaban yang dihasilkan dan *golden answer* menggunakan *cosine similarity* atas representasi vektor dari kedua jawaban. Skor tinggi menunjukkan bahwa secara makna, jawaban yang dihasilkan sangat mirip dengan *golden answer*, meskipun berbeda secara tekstual.

BAB IV
HASIL DAN PEMBAHASAN

Bab ini menyajikan hasil dan analisis mendalam terhadap sistem RAG berbasis *Knowledge Graph* yang telah dirancang dan diimplementasikan. Penyajian dimulai dari tahap ekstraksi dan penyerapan pengetahuan dari korpus dokumen. Selanjutnya dipaparkan hasil evaluasi kuantitatif dari rancangan arsitektur RAG, baik pada komponen *retriever*, maupun jawaban final yang dihasilkan oleh sistem secara keseluruhan. Setiap hasil akan diikuti dengan pembahasan untuk menginterpretasikan data, menganalisis kekuatan dan kelemahan sistem, serta menyajikan studi kasus untuk memberikan gambaran yang lebih jelas.

4.1 Hasil *Knowledge Extraction*

Proses *knowledge extraction* dimulai dengan ekstraksi teks dari dokumen. Dokumen yang dalam format HTML atau DOCX kemudian diambil teks yang terkandung di dalamnya sekaligus dilakukan *structural chunking* untuk membaginya ke dalam beberapa potongan berdasarkan tingkatan hierarki tertentu. Proses *chunking* hanya dilakukan pada dokumen buku "Petunjuk Teknis Pencegahan dan Pengendalian Gangguan Mental Emosional" karena berukuran cukup besar, yaitu sekitar 15.000 kata, dibandingkan dengan dokumen lainnya yang kurang dari 6.000 kata. Hal tersebut dilakukan mengingat LLM kesulitan untuk secara detail memahami isi dari dokumen tersebut. Dari proses *chunking* tersebut menghasilkan total 17 potongan dokumen yang disimpan dalam format TXT.

Sejumlah 17 potongan dokumen tersebut kemudian dimasukkan ke dalam LLM untuk diekstrak entitas dan relasi yang terkandung secara berurutan. Dari proses ekstraksi tersebut menghasilkan 1.119 entitas dan 1.078 relasi secara keseluruhan. Tabel 4.1 menunjukkan statistik *Knowledge Graph* yang dihasilkan dari dokumen secara keseluruhan

Tabel 4.1. Statistik *Knowledge Graph* yang dihasilkan dari dokumen

Komponen	**Jumlah**
Dokumen	5
Potongan Dokumen	17
Entitas	1119
Relasi	1073
Jenis Entitas	18
Jenis Relasi	18

Suatu fakta yang berhasil diekstrak dari dokumen akan disimpan dalam bentuk triplet (subjek, predikat, dan objek), yang kemudian diimplementasikan sebagai *node* dan

edge dalam struktur graf pengetahuan. Sebagai contoh, berikut adalah sebuah potongan dokumen yang dianalisis:

> *Faktor psikologis yaitu kegagalan, kekecewaan, trauma yang mengakibatkan stres dan faktor biologis yaitu genetik, gizi dan infeksi perinatal serta kondisi lingkungan yang buruk juga merupakan faktor yang berkontribusi terhadap terjadinya gangguan jiwa.*
>
> ***Faktor Psikologis***
> *Faktor psikologis yang menjadi faktor risiko GME, antara lain (Kerig, Ludlow, & Wenar, 2012; PPSDM Kemenkes RI, 2016):*
>
> 1. *Regulasi emosi rendah*
> 2. *Kemampuan regulasi diri rendah yang termanifestasikan dalam kontrol perilaku yang buruk*
> 3. *Konsep diri negatif*
> 4. *Efikasi diri rendah*
> 5. *Resiliensi diri rendah*
>
> ***Cara menangani Resiliensi Diri Rendah:***
>
> a) *Melatih keterampilan manajemen stres*
> b) *Melatih fleksibilitas dalam berpikir dan keterampilan pengambilan keputusan*
> c) *Melatih keterampilan manajemen konflik*

Berdasarkan potongan dokumen tersebut, sistem ekstraksi menghasilkan beberapa triplet pengetahuan sebagai berikut:

```
(:Gangguan_Mental {name: "Gangguan Jiwa"})<-[:Berkontribusi_Pada]-(:
   Faktor_Risiko {name: "Faktor Psikologis"})

(:Gangguan_Mental {name: "Gangguan Jiwa"})<-[:Berkontribusi_Pada]-(:
   Faktor_Risiko {name: "Faktor Biologis"})

(:Gejala {name: "Resiliensi Diri Rendah"})-[:Adalah_Jenis_Dari]->(:
   Faktor_Risiko {name: "Faktor Psikologis"})

(:Aktivitas_Kesejahteraan {name: "Manajemen Konflik"})-[:Menangani
   ]->(:Gejala {name: "Resiliensi Diri Rendah"})

(:Aktivitas_Kesejahteraan {name: "Keterampilan Pengambilan Keputusan
   "})-[:Menangani]->(:Gejala {name: "Resiliensi Diri Rendah"})
```

4.2 Hasil Pengembangan Arsitektur RAG Berbasis *Knowledge Graph*

Sistem yang dikembangkan berupa AI *service* berbasis arsitektur RAG-KG. Layanan AI tersebut disediakan dalam bentuk REST API sehingga memudahkan dalam integrasi dengan platform lain. Antarmuka aplikasi REST juga memudahkan dalam pengujian dan menyimulasikan sistem ketika diimplementasikan pada skenario penggunaan sebenarnya. Arsitektur dirancang untuk mendukung proses penjawaban pertanyaan yang memprioritaskan jawaban berbasis konteks. Basis konteks tersebut diperoleh dari pengambilan informasi pada *Knowledge Graph* dengan memanfaatkan entitas dan relasi yang terkandung di dalamnya. AI *service* dapat diakses melalui HTTP *request* dengan spesifikasi seperti pada Tabel 4.2.

Tabel 4.2. Spesifikasi *endpoint* untuk menghasilkan jawaban

Komponen	**Nilai**
URL	http://localhost:8080/api/v1/retrieval/get-answer
HTTP *Method*	POST

***Body* (JSON)**	<pre>{
	  "query": "",
	  "graph_traversal": "",
	  "search_method": "",
	  "max_hop": 0,
	  "limit": 0,
	  "top_k": 0
	}</pre>

Komponen	Nilai

Respons (JSON)	{
	"message": "",
	"data": {
	"latency" : 0.0,
	"method": "",
	"query_class": {
	"category": "",
	"entities": []
	},
	"answer" : "",
	"knowledge": "",
	"graph" : {}
	}
	}

Untuk memperoleh respons dari sistem dilakukan pemanggilan API dengan spesifikasi seperti pada Tabel 4.2. Berikut merupakan potongan contoh respons sistem untuk pertanyaan *"Apa itu Gangguan Mental Emosional?"* pada Tabel 4.3.

Tabel 4.3. Contoh respons sistem dalam menghasilkan jawaban

Komponen	Isi
Kueri	Apa itu Gangguan Mental Emosional?

Request Body	{
	"query": "Apa itu Gangguan Mental Emosional ?",
	"graph_traversal": "auto",
	"search_method": "auto",
	"max_hop": 10,
	"limit": 50,
	"top_k": 3
	}

Komponen	**Isi**
Konteks Pengetahuan	Berikut adalah informasi yang ditemukan dari knowledge graph: **Entitas: Gangguan Mental Emosional** (Tipe: Gangguan_Mental) • Deskripsi: Bukan diagnosis gangguan jiwa. Dapat memengaruhi aktifitas sehari-hari yang berdampak pada menurunnya produktifitas dan kualitas hidup. Kondisi atau gangguan yang menjadi subjek utama dari 'Petunjuk Teknis Pencegahan dan Pengendalian Gangguan Mental Emosional'. .... • Skor Relevansi Awal: 5.07 • Hubungan Terkait: → *merupakan* → Masalah Kesehatan Jiwa → *mengindikasikan* → Perubahan Psikologis ← *mencegah* ← Upaya Promosi Kesehatan ... **Entitas: Deteksi Gangguan Mental Emosional** (Tipe: Layanan_Kesehatan) • Deskripsi: Proses identifikasi gangguan mental emosional yang diharapkan mampu dilakukan oleh fasilitas pelayanan kesehatan primer. • Skor Relevansi Awal: 4.52 • Hubungan Terkait: ← *mampu melakukan* ← Fasilitas pelayanan kesehatan primer → *dapat segera diintervensi* → Intervensi ... **Penderita Gangguan Mental Emosional** (Tipe: Individu) • Deskripsi: Proses identifikasi gangguan mental emosional yang diharapkan mampu dilakukan oleh fasilitas pelayanan kesehatan primer. ... (Informasi selengkapnya dapat dilihat pada Lampiran)

Komponen	Isi
Jawaban	Gangguan Mental Emosional (GME) adalah suatu kondisi yang mengindikasikan seseorang mengalami perubahan psikologis. Perubahan ini bisa merupakan kondisi normal, namun dapat juga merupakan kondisi patologis. Penting untuk diketahui bahwa Gangguan Mental Emosional **bukanlah diagnosis gangguan jiwa**. Meskipun demikian, kondisi ini merupakan masalah kesehatan jiwa yang banyak dijumpai di Indonesia, diperkirakan diderita oleh 19 juta orang. GME dapat memengaruhi aktivitas sehari-hari seseorang, yang pada akhirnya berdampak pada menurunnya produktivitas dan kualitas hidup. Istilah ini juga digunakan dalam “Petunjuk Teknis Pencegahan dan Pengendalian Gangguan Mental Emosional” sebagai subjek utama panduan tersebut.

Pada potongan contoh respons sistem untuk pertanyaan *"Apa itu Gangguan Mental Emosional"* pada tabel 4.3, sistem akan mengambil konteks pengetahuan yang diperoleh dari *Knowledge Graph*. Terlihat dalam konteks pengetahuan yang diambil sistem berhasil mengambil beberapa entitas utama (Gangguan Mental Emosional, Deteksi Gangguan Mental Emosional, dan Penderita Gangguan Mental Emosional) beserta entitas tetangganya. Konteks pengetahuan yang diperoleh tidak hanya menyediakan definisi, tetapi juga relasi antar entitas, seperti "merupakan", "mengindikasikan", dan "mencegah". Hal tersebut sangat penting untuk menyediakan konteks bagi LLM sehingga mampu menghasilkan jawaban yang holistik. Jawaban yang dihasilkan tidak hanya menjawab pertanyaan *"Apa itu Gangguan Mental Emosional?"* secara langsung, tetapi juga memberikan informasi tambahan yang relevan dan penting. Jawaban akhir secara langsung merefleksikan dan menyintesis informasi yang ada di dalam konteks pengetahuan, membuktikan bahwa sistem tidak berhalusinasi, setidaknya pada contoh ini.

4.3 Hasil Pengembangan *Dataset* Evaluasi Sitetis

Dataset evaluasi diperoleh secara sintesis melalui proses *prompting* pada model OpenAI o4-mini. Data yang dihasilkan berupa pasangan *query* atau pertanyaan dan *golden answer* atau jawaban yang dianggap sebagai acuan jawaban yang diharapkan. Selain itu dibubuhkan juga *golden nodes* yang merujuk pada *nodes* yang relevan untuk menjawab pertanyaan dan `query_lable` untuk menandai jenis pertanyaan. Dari proses *prompting* diperoleh hasil berupa 433 data evaluasi yang terdiri dari 266 `entity_query` dan 167 `path_query` seperti pada Tabel 4.4. Setiap pertanyaan tersebut dijamin memi-

liki referensi ke setidaknya satu *node* dalam *Knowledge Graph*. Contoh *dataset* sintesis yang dihasilkan dapat dilihat pada Gambar 4.12.

Tabel 4.4. Hasil sintesis *dataset* evaluasi menggunakan model OpenAI o4-mini

Kategori Kueri	**Jumlah**
`entity_query`	266
`path_query`	167
Total	433


```
{
  "id": null,
  "query": "Bagaimana Puskesmas berperan dalam upaya promotif dan preventif gangguan mental emosional?",
  "query_lable": "path_query",
  "golden_nodes": [
    "Puskesmas",
    "Pencegahan dan Pengendalian Gangguan Mental Emosional"
  ],
  "golden_answer": "Puskesmas melatih tenaga kesehatan dan masyarakat untuk promosi kesehatan jiwa, melakukan deteksi dini,
    serta menjalankan kegiatan preventif untuk menurunkan risiko gangguan mental emosional di wilayahnya."
},
{
  "id": null,
  "query": "Apa yang dimaksud Gangguan Mental dan Emosional (GME)?",
  "query_lable": "entity_query",
  "golden_nodes": [
    "Gangguan Mental dan Emosional (GME)"
  ],
  "golden_answer": "GME adalah kondisi di mana seseorang mengalami gejala emosional dan mental yang mengganggu fungsi sehari-hari,
    seperti kecemasan berlebihan atau perubahan suasana hati yang ekstrem."
},
{
  "id": null,
  "query": "Bagaimana Lingkungan Pendidikan membantu mencegah GME?",
  "query_lable": "path_query",
  "golden_nodes": [
    "Lingkungan Pendidikan",
    "Gangguan Mental dan Emosional (GME)"
  ],
  "golden_answer": "Lingkungan pendidikan yang sehat menyediakan edukasi dan deteksi dini GME melalui penyuluhan, KIE, dan kerja sama
    dengan guru agar gejala dapat dikenali lebih awal."
},
{
  "id": null,
  "query": "Bagaimana CPMH memfasilitasi sivitas akademika Fakultas Psikologi UGM?",
  "query_lable": "path_query",
  "golden_nodes": [
    "CPMH",
    "Sivitas Akademika Fakultas Psikologi UGM"
  ],
  "golden_answer": "CPMH memfasilitasi sivitas akademika Fakultas Psikologi UGM dengan menyediakan akses ke pusat kajian, pendidikan,
    pelatihan, serta kesempatan untuk terlibat dalam advokasi kebijakan dan layanan kesehatan mental masyarakat. Dengan dukungan ini,
    akademisi dapat mengembangkan ilmu dan profesinya demi kesejahteraan bersama."
}
```


Gambar 4.12. Contoh *dataset* evaluasi sintesis yang dihasilkan oleh LLM

Gambar 4.12 menunjukkan contoh *dataset* evaluasi yang mencakup kategori `entity_query` dan `path_query`. Setiap data sintetis terdiri dari 4 *field* yaitu *query*, *query lable*, *golden nodes*, dan *golden answer*. *Query* merupakan pertanyaan yang jawabannya dapat diketahui dari informasi pada *Knowledge Graph*. Setiap kueri memiliki kategori yaitu *entity query* atau *path query* yang terdapat pada *query lable*. *Golden nodes* berisi *nodes* yang diperlukan untuk menjawab pertanyaan pada *field query*, sedangkan *golden answer* merupakan jawaban yang dianggap benar untuk menjawab pertanyaan.

Pertanyaan *"Apa yang dimaksud Gangguan Mental Emosional (GME)?"* termasuk ke dalam `entity_query` karena menanyakan definisi dari suatu entitas yaitu Gangguan Mental Emosional (GME). Jawaban dari pertanyaan tersebut seharusnya dapat ditemukan pada *node* Gangguan Mental Emosional (GME) atau *nodes* tetangganya. *Golden nodes* yang digunakan sebagai konteks untuk menjawab pertanyaan ini adalah Gangguan Mental Emosional (GME), sesuai dengan pertanyaan yang spesifik menanyakan GME. *Golden answer* yang dihasilkan secara langsung menjawab pertanyaan sesuai konteks yang diberikan yaitu dengan memberikan definisi dari GME.

Pada pertanyaan *"Bagaimana CPMH memfasilitasi sivitas akademika Fakultas Psikologi UGM?"* terlihat ada dua entitas yang terkandung, yaitu CPMH dan sivitas akademika Fakultas Psikologi. Pertanyaan tersebut termasuk dalam `path_query` karena jawabannya seharusnya dapat diperoleh dengan melihat hubungan antar entitas yang terkandung di dalamnya. Untuk menjawab pertanyaan tersebut diperlukan dua *nodes*, yaitu CPMH dan Sivitas Akademika Fakultas Psikologi UGM seperti yang ada pada *golden nodes* beserta *nodes* lain yang berada di antara keduanya. Jawaban yang terdapat pada *golden answer* menjawab pertanyaan dengan memberikan hubungan antara CPMH dan sivitas akademika Fakultas Psikologi UGM, yaitu sebagai fasilitator, dilengkapi dengan contohnya.

4.4 Analisis RAG Berbasis *Knowledge Graph*

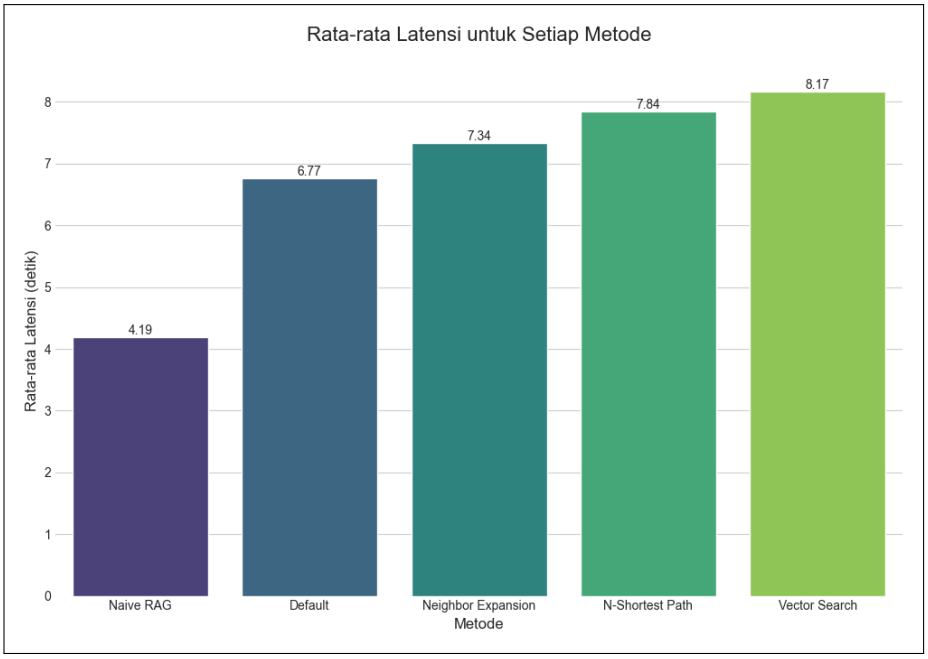
Evaluasi sistem dilakukan pada dua keluaran utama RAG berbasis *Knowledge Graph* yaitu konteks yang dikeluarkan oleh komponen *retrieval* dan jawaban final yang dihasilkan. Bagian *retriever* dilakukan evaluasi untuk mengukur sejauh mana sistem dapat mengambil konteks yang relevan dari *Knowledge Graph*. Jawaban final akan diukur kualitas keluaran LLM setelah diberikan konteks dari *retriever*. Selain itu, dilakukan juga analisis kesalahan untuk mengidentifikasi penyebab kegagalan sistem dalam menjawab pertanyaan.

Evaluasi dilakukan dengan konfigurasi perlakuan seperti pada Tabel 4.5 dengan mengatur parameter pada *body* API. Perlakuan dilakukan dengan mengatur parameter `search_method` dan `graph_traversal` untuk mengatur algoritma pencarian dan penelusuran graf yang digunakan oleh *retriever*. Setiap pertanyaan pada *dataset* evaluasi akan ditanyakan pada sistem dengan konfigurasi parameter menurut masing-masing perlakuan. Naive RAG atau RAG berbasis potongan dokumen digunakan sebagai *baseline* yang akan dibandingkan dengan RAG berbasis *Knowledge Graph*, terutama dalam hal kemampuannya menghasilkan jawaban final.

Tabel 4.5. Perlakuan yang digunakan untuk mengevaluasi RAG berbasis *Knowledge Graph* dengan *naive* RAG sebagai *baseline*

Perlakuan	**Keterangan**
Naive RAG (Baseline)	RAG dengan basis *document chunk* menggunakan pencarian vektor sebagai *retriever*. *Document chunk* berukuran 1.200 karakter dengan overlap 400 karakter. Perlakuan ini digunakan sebagai *baseline* evaluasi.
Default	*Search method*: auto, *graph traversal*: auto. Sistem otomatis memilih metode pencarian terbaik. Awalnya mencoba *full-text search*, jika tidak ditemukan hasil, baru menggunakan pencarian vektor. Algoritma penelusuran graf ditentukan oleh modul klasifikasi kueri secara otomatis.
Vector Search	*Search method*: vector, *graph traversal*: auto. Menggunakan pencarian node berbasis vektor (HNSW) dengan *cosine similarity*. Algoritma penelusuran graf ditentukan oleh modul klasifikasi kueri secara otomatis.
Neighbor Expansion	*Search method*: auto, *graph traversal*: neighbor_expansion. Sistem menelusuri graf dengan algoritma *one-hop neighbor expansion*. Algoritma pencarian ditentukan secara otomatis.
N-Shortest Path	*Search method*: auto, *graph traversal*: n-shortest_path. Sistem menelusuri graf menggunakan algoritma *n-shortest path*. Algoritma pencarian ditentukan secara otomatis.

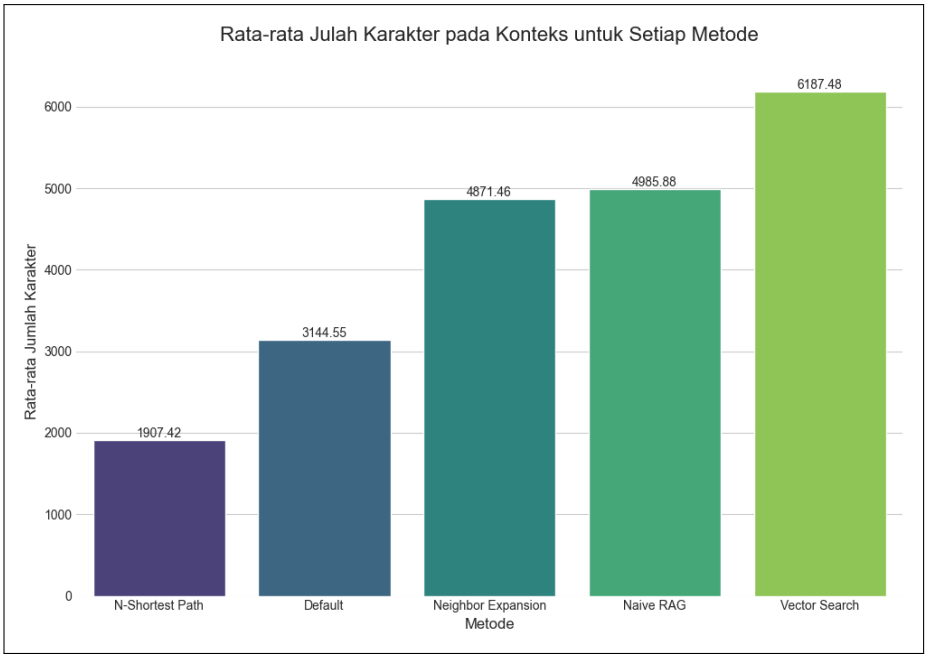
Hasil dari *request* pada sistem untuk setiap perlakuan pada Tabel 4.5 menghasilkan data yang disimpan untuk evaluasi lebih lanjut. Data hasil *request* pada sistem yang disimpan terdiri atas `knowledge`, `llm_answer`, `latency`, dan `graph_entities`. Komponen dari data tersebut yang kemudian akan dijadikan bahan evaluasi. Latensi dari masing-masing perlakuan dapat dilihat pada Gambar 4.13



Gambar 4.13. Rerata latensi respons sistem pada setiap perlakuan

Secara umum pendekatan Naive RAG memiliki rerata latensi yang lebih singkat, yaitu 4,19 detik. Angka ini lebih unggul dari RAG-KG yang memiliki latensi lebih dari 6 detik. Hal tersebut dapat dipahami karena pendekatan berbasis *Knowledge Graph* menggunakan pemanggilan LLM melalui API setidaknya sebanyak dua kali, yaitu pada proses ekstraksi dan klasifikasi pada kueri serta pada tahap *answer generation*. Sebaliknya, Naive RAG hanya memerlukan satu kali pemanggilan LLM. Selain itu, komponen *retriever* pada Naive RAG hanya perlu mencari dan mengambil potongan-potongan dokumen berdasarkan kemiripan teks, yang merupakan operasi yang lebih cepat dan tidak memerlukan proses penelusuran kompleks.

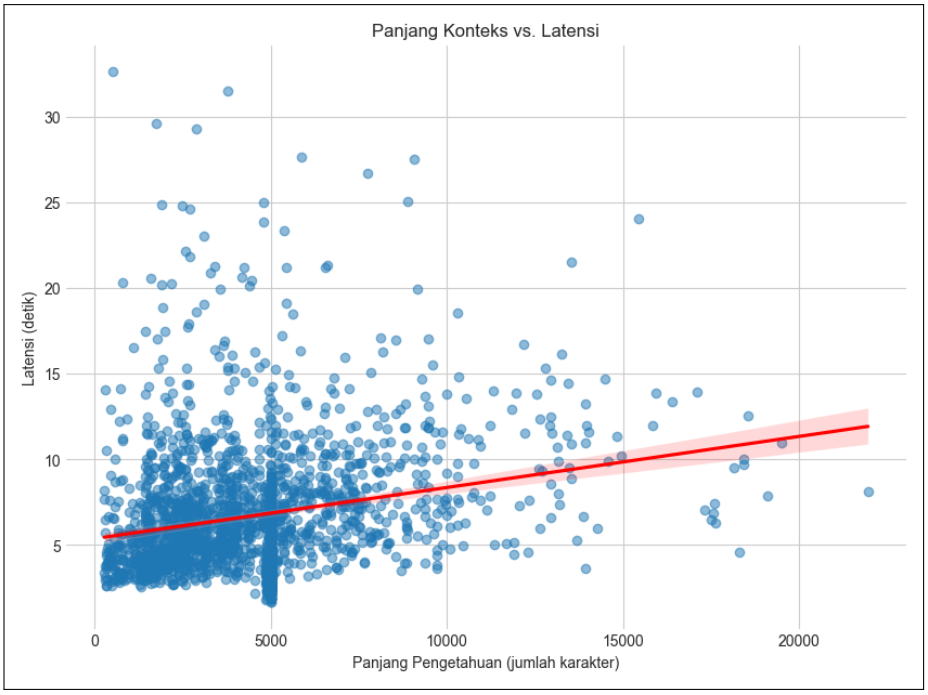
Knowledge atau konteks yang dihasilkan oleh komponen *retriever* akan dipakai sebagai konteks tambahan bagi LLM untuk menjawab pertanyaan. Rerata panjang konteks dari masing-masing perlakukan dapat dilihat pada Gambar 4.14.



Gambar 4.14. Rerata panjang konteks dari respons sistem pada setiap perlakuan

Dari Gambar 4.14 dapat dilihat pendekatan berbasis *Knowledge Graph* secara umum memiliki panjang konteks yang lebih sedikit dari Naive RAG kecuali pada Vector Search. Pendekatan N-Shortest Path menghasilkan rerata panjang konteks paling sedikit sebesar 1907,42 kemudian diikuti oleh Default (3144,55), Neighbor Expansion (4871,46), Naive RAG (4985,88), dan Vector Search (6187,48). Hal tersebut dapat dipahami karena pendekatan dengan pencarian jalur menghasilkan lebih sedikit entitas dibandingkan dengan pendekatan ekspansi tetangga. Pendekatan Default terlihat sebagai titik tengah antara N-Shortest Path dan Neighbor Expansion karena secara dinamis menentukan algoritma penelusuran graf yang sesuai. Di sisi lain, pendekatan berbasis vektor seperti pada Naive RAG dan Vector Search cenderung menghasilkan konteks yang lebih panjang karena menggunakan k dokumen atau *node* teratas berdasarkan kemiripan (dengan $k = 5$ dalam kasus ini), yang secara inheren cenderung menarik lebih banyak teks dibandingkan pencarian dengan *full-text search*.

Konteks yang lebih panjang secara intuitif akan membuat LLM memerlukan waktu pemrosesan lebih lama yang akan memengaruhi latensi respons sistem secara keseluruhan. Hal tersebut dapat dilihat pada Gambar 4.15 yang menunjukkan adanya hubungan positif antara panjang konteks dengan latensi sistem. Semakin banyak konteks yang diberikan pada LLM akan menghasilkan latensi respons yang lebih lama.



Gambar 4.15. *Scatter plot* panjang konteks dengan latensi keseluruhan respons

4.4.1 Analisis Kinerja Komponen *Retriever*

Pengujian komponen *retriever* dilakukan untuk mengukur kualitas informasi atau konteks yang diambil dari *Knowledge Graph*. Informasi yang terdapat pada KG adalah kumpulan subgraf yang terdiri atas *nodes* yang dihubungkan oleh *edge* tertentu. Kumpulan subgraf tersebut yang kemudian disusun dalam bentuk tekstual yang memiliki makna semantik. Alih-alih mengevaluasi konteks yang sudah berupa teks panjang yang memerlukan alat yang dapat memahami bahasa alami seperti LLM yang kurang pasti dalam menguji, pengujian dilakukan pada *nodes* yang dihasilkan sehingga evaluasi lebih terkontrol dan akurat. Terdapat empat metrik uji untuk mengevaluasi kinerja komponen *retriever*, yaitu *precision*, *recall*, *Mean Reciprocal Rank* (MRR), dan *hit ratio*. Tabel 4.6 menunjukkan hasil pengujian empat metrik uji tersebut terhadap empat perlakukan RAG-KG pada Tabel 4.5.

Tabel 4.6. Hasil evaluasi *retriever* pada RAG berbasis *Knowledge Graph* untuk masing-masing perlakuan

Perlakuan	**Precision**	**Recall**	**MRR**	**Hit Ratio**
Default	**0,1530**	0,8203	**0,3758**	0,9053
Vector Search	0,0603	0,5929	0,1737	0,6744
Neighbor Expansion	0,1123	**0,8868**	0,3191	**0,9584**
N-Shortest Path	0,1245	0,5370	0,2788	0,6097

Berdasarkan hasil pada Tabel 4.6, terlihat bahwa nilai rerata *precision* tertinggi dicapai oleh perlakuan Default (0,1530), menunjukkan bahwa dari subgraf yang dihasilkan, metode ini memiliki relevansi paling tinggi. Meskipun demikian, nilai presisi 0,1530 masih tergolong rendah karena hanya sekitar 15.3% dari total *node* yang diambil benar-benar relevan. Dengan kata lain, mayoritas *node*, yaitu sekitar 85% dari konteks yang dihasilkan oleh metode ini merupakan *noise*. Metode Vector Search memiliki *precision* terendah (0,0603), mengindikasikan lebih banyak informasi yang tidak relevan diambil secara proporsional.

Pada metrik *Recall*, hasil yang sangat baik diperoleh oleh Neighbor Expansion (0,8868) dan Default (0,8203), yang berarti metode ini mampu mengambil hampir seluruh informasi relevan dari *Knowledge Graph*. Di sisi lain, metode N-Shortest Path dan Vector Search mencatat nilai *recall* yang relatif rendah (0,5370 dan 0,5929), menandakan adanya keterbatasan dalam menjangkau konteks relevan.

Metrik *Mean Reciprocal Rank* tertinggi dicapai oleh Default (0,3758) dan diikuti Neighbor Expansion (0,3191), mengindikasikan bahwa konteks yang relevan cenderung berada pada urutan atas hasil pencarian (urutan 2 hingga 3 secara rata-rata). Sebaliknya, Vector Search (0,1737) dan N-Shortest Path (0,2788) menempati posisi bawah yang mengindikasikan konteks relevan sering berada pada urutan bawah (urutan 5 sampai 6 secara rata-rata). Untuk *Hit Ratio*, nilai tertinggi diraih oleh Neighbor Expansion (0,9584), diikuti Default (0,9053), menandakan bahwa metode tersebut hampir selalu berhasil menemukan setidaknya satu konteks relevan dalam hasil pencarian. Sebaliknya, N-Shortest Path dan Vector Search mencatat *hit ratio* yang rendah 0,6097 dan 0,6744, menunjukkan bahwa masih ada kemungkinan yang cukup besar (30% hingga 40%) informasi gagal ditemukan.

Dari keempat metrik uji *retriever*, terlihat perlakuan Neighbor Expansion dan Default memperoleh masing-masing dua metrik dengan nilai tertinggi dalam pengujian. Metode Default unggul pada *precision* dan MRR, sedangkan Neighbor Expansion unggul pada *recall* dan *hit ratio*. Hal ini mengindikasikan bahwa Default lebih selektif dalam mengambil konteks, sedangkan Neighbor Expansion lebih agresif dan luas cakupannya, sehingga *recall*-nya tinggi tetapi precision-nya moderat. Neighbor Expansion cenderung menghasilkan data yang lebih lengkap tetapi mengandung lebih banyak noise dibandingkan dengan Default. Hal tersebut dapat dipahami karena metode ini mengambil *node* yang relevan dan *nodes* tetangganya yang jumlahnya cukup banyak. Di sisi lain, Default memilih algoritma penelusuran graf secara adaptif, sesuai dengan klasifikasi kueri, yang lebih sesuai dengan kebutuhan meskipun kalah dalam hal kelengkapan informasi. Vector Search cenderung kurang optimal di semua metrik, menunjukkan bahwa pendekatan berbasis vektor saja kurang efektif dalam melakukan pencarian dalam graf. Sementara itu, N-Shortest Path memberikan hasil yang kurang optimal untuk menangani sebagian

besar kueri. Secara keseluruhan, Neighbor Expansion dan Default menjadi opsi utama *retriever* dalam menghadapi mayoritas pertanyaan.

4.4.2 Analisis Respons Final

Pengujian respons final dilakukan untuk mengukur kualitas jawaban akhir yang dihasilkan oleh RAG berbasis *Knowledge Graph* setelah diberikan konteks tambahan oleh komponen *retriever*. Informasi berupa subgraf yang terdiri dari kumpulan *nodes* dan *edges* yang terhubung ditransformasikan ke dalam bentuk tekstual yang lebih bermakna secara semantik. Bentuk tekstual tersebut kemudian dijadikan sebagai konteks untuk memberikan informasi tambahan dalam menghasilkan jawaban. Jawaban akhir dievaluasi dengan memperhatikan empat parameter, yaitu pertanyaan, jawaban yang dihasilkan, konteks, dan jawaban seharusnya sebagai *ground truth*.

Evaluasi dilakukan dengan *framework* RAGAS (*Retrieval-Augmented Generation Assessment*), yang merupakan evaluator RAG dengan salah satu alat ujinya menggunakan LLM. Metrik uji untuk mengevaluasi jawaban adalah *similarity*, *correctness*, *relevancy*, dan *faithfulness* yang merupakan metrik bawaan RAGAS. Tabel 4.7 menunjukkan hasil evaluasi RAGAS pada masing-masing perlakukan, termasuk dengan *baseline* Naive RAG.

Tabel 4.7. Hasil evaluasi jawaban final pada RAG berbasis *Knowledge Graph* untuk masing-masing perlakuan dan dibandingkan dengan *baseline* Naive RAG

Perlakuan	**Similarity**	**Correctness**	**Relevancy**	**Faithfulness**
Naive RAG	0,9048	0,3391	0,6996	0,8993
Default	0,9242	0,4155	0,8624	**0,9435**
Vector Search	0,9115	0,3895	0,7387	0,8902
Neighbor Expansion	**0,9248**	**0,4246**	**0,8723**	0,9428
N-Shortest Path	0,9033	0,3936	0,7900	0,4614

Metrik *similarity* mengukur kesamaan semantik dengan membandingkan bentuk vektor (*embedding*) antara jawaban yang dihasilkan sistem dan jawaban *ground truth*. Hasil menunjukkan bahwa semua perlakuan, termasuk *baseline* Naive RAG, memiliki skor *similarity* yang relatif tinggi di atas 90%, yang menandakan bahwa sistem mampu menghasilkan jawaban yang secara umum memiliki kedekatan makna dengan jawaban seharusnya. Nilai tertinggi dicapai oleh perlakuan Neighbor Expansion (0,9248) diikuti dengan Default (0,9242), yang mengindikasikan bahwa perluasan konteks berbasis *Knowledge Graph* secara umum sama efektifnya dalam menjaga kesamaan makna dibandingkan dengan *baseline*.

Metrik *correctness* menilai kebenaran faktual dari jawaban yang dihasilkan. Hasil menunjukkan peningkatan hingga sebesar 8,55% pada perlakuan yang memanfaatkan *Knowledge Graph* khususnya pada perlakuan Neighbor Expansion (0,4246) dan Default (0,4155) dibanding *baseline* Naive RAG (0,3391). Pada metrik ini, semua metode RAG-KG menunjukkan adanya peningkatan akurasi faktual dibandingkan dengan Naive RAG. Peningkatan ini secara langsung mengatribusikan kemampuan RAG-KG dalam menyajikan konteks faktual dari KG sebagai faktor yang berhasil meningkatkan akurasi kebenaran dari jawaban akhir.

Relevansi menunjukkan sejauh mana jawaban yang dihasilkan sistem tetap fokus dan sesuai dengan pertanyaan yang diajukan. Neighbor Expansion (0,8723) dan Default (0,8624) menunjukkan peningkatan sebesar 17% dari Naive RAG (0,6996) pada metrik ini. Peningkatan tersebut menunjukkan bahwa konteks yang dihasilkan dari *Knowledge Graph* tidak hanya memberikan jawaban yang lebih faktual, tetapi juga lebih tepat sasaran. Konteks yang lebih tepat sasaran akan membuat LLM lebih mudah dalam menghasilkan jawaban yang lebih relevan dengan pertanyaan.

Faithfulness mengukur kesesuaian informasi dalam jawaban terhadap konteks yang diberikan oleh *retriever*. Hampir semua metode menghasilkan skor di atas 0,89, kecuali pada metode N-Shortest Path yang memiliki skor sangat rendah (0,4614), menandakan bahwa metode ini lebih sering menghasilkan jawaban yang tidak konsisten atau tidak sepenuhnya didukung oleh konteks. Nilai tinggi pada Default (0,9435) dan Neighbor Expansion (0,9428) menunjukkan bahwa strategi ini paling konsisten dalam menggunakan informasi yang tersedia. RAG-KG dengan strategi tersebut tetap lebih tinggi sekitar 4% dibandingkan dengan *baseline* Naive RAG menghasilkan skor (0,8993). Ini adalah bukti bahwa konteks yang diperkaya oleh *Knowledge Graph* dengan pemilihan metode yang tepat secara efektif dapat mengurangi LLM dari halusinasi atau mengarang informasi.

Hasil evaluasi jawaban final secara umum menunjukkan bahwa penggunaan *Knowledge Graph* yang tepat dapat meningkatkan kualitas jawaban akhir secara signifikan. Peningkatan tertinggi dari *baseline* dapat diperoleh sistem RAG-KG yang mengimplementasikan metode Default dan Neighbor Expansion hingga 8% secara rata-rata. Penggunaan metode Vector Search secara umum menunjukkan peningkatan dibandingkan dengan Naive RAG meskipun tertinggal dari RAG-KG dengan metode pencarian *full-text* (Default dan Neighbor Expansion) dalam hal relevansi dan *correctness*. Sementara itu, penggunaan N-Shortest Path dalam melakukan penelusuran graf secara umum berpotensi menghasilkan jawaban yang tidak akurat ditandai dengan skor *faithfulness* yang sangat rendah, bahkan lebih rendah dibandingkan dengan *baseline*.

4.4.3 Analisis Kesalahan dan Studi Kasus

Analisis terakhir adalah meninjau secara langsung jawaban yang dihasilkan oleh masing-masing metode. Pada studi kasus kueri *"Apa yang dimaksud dengan gejala psikotik?"*, terlihat perbedaan kualitas jawaban yang dihasilkan oleh berbagai metode RAG berbasis *Knowledge Graph*. Jawaban yang diharapkan (*golden answer*) memuat penjelasan singkat bahwa gejala psikotik meliputi halusinasi atau delusi seperti mendengar suara, yang dapat memengaruhi pola pikir dan berisiko terkait pikiran bunuh diri. Hasil jawaban secara lengkap dapat dilihat pada Tabel 4.8.

Tabel 4.8. Komparasi kueri dan jawaban oleh masing-masing metode

Kueri	Apa yang dimaksud dengan gejala psikotik?
Golden Answer	Gejala psikotik meliputi halusinasi atau delusi seperti mendengar suara yang dapat memengaruhi cara seseorang berpikir dan berisiko terkait pikiran bunuh diri. (**Terdapat Konteks**)
Naive RAG	Mohon maaf, konteks yang diberikan tidak menjelaskan apa yang dimaksud dengan gejala psikotik. Konteks tersebut hanya menyebutkan tentang individu risiko tinggi psikotik dan pencegahan transisi menjadi gangguan psikotik. (**Terdapat Konteks**)
Default	Gejala psikotik adalah gejala yang mungkin dialami oleh Orang dengan Kebutuhan Disabilitas (OKBD) yang bisa memengaruhi keinginan bunuh diri, contohnya adalah mendengar suara-suara. (**Terdapat Konteks**)
Vector Search	Saya mohon maaf, saya tidak menemukan informasi spesifik mengenai "gejala psikotik" dalam pengetahuan yang saya miliki. Pengetahuan yang tersedia lebih berfokus pada keluhan fisik, gejala somatik, faktor biologis, dan tanda bahaya kecenderungan bunuh diri. (**Terdapat Konteks**)
Neighbor Expansion	Gejala psikotik adalah gejala yang mungkin dialami oleh Orang dengan Kondisi Biologis dan Diagnosa (OKBD) yang bisa memengaruhi keinginan bunuh diri, seperti mendengar suara-suara. Gejala ini juga dapat memperbesar risiko bunuh diri. (**Terdapat Konteks**)

N-Shortest Path	Gejala psikotik adalah serangkaian pengalaman yang menunjukkan hilangnya kontak dengan realitas. Ini sering kali melibatkan perubahan dalam pikiran, persepsi, dan perilaku seseorang. Gejala psikotik utama meliputi: 1. **Halusinasi:** Persepsi sensorik tanpa rangsangan eksternal. 2. **Delusi:** Keyakinan salah yang bertahan meskipun ada bukti bertentangan. 3. **Pemikiran dan Bicara Tidak Terorganisir:** Kesulitan mengatur pikiran sehingga bicara menjadi kacau. 4. **Perilaku Tidak Terorganisir atau Katatonik:** Perilaku aneh, agitasi, atau tidak bergerak sama sekali. (**Tanpa Konteks**)

Metode Naive RAG gagal memberikan jawaban yang diharapkan, yaitu definisi gejala psikotik, hanya menyampaikan bahwa konteks tidak menjelaskan gejala psikotik dan sekadar menyinggung individu berisiko tinggi psikotik serta pencegahan transisi menjadi gangguan psikotik. Jawaban tersebut menunjukkan bahwa *retriever* gagal dalam memberikan konteks yang relevan dan sesuai untuk menjawab pertanyaan. Hal serupa juga dapat dilihat pada metode Vector Search, yang menunjukkan jawaban yang mengaku tidak menemukan informasi spesifik dan malah memaparkan topik terkait keluhan fisik, faktor biologis, dan tanda bahaya kecenderungan bunuh diri, yang kurang relevan dengan pertanyaan. Hal ini membuktikan bahwa untuk kueri yang menuntut definisi istilah khusus, pencarian semantik berbasis vektor masih memiliki potensi kegagalan dalam mengambil konteks.

Sebaliknya, Default dan Neighbor Expansion berhasil mendekati topik yang benar, menyebutkan bahwa gejala psikotik dapat memengaruhi keinginan bunuh diri dan memberikan contoh seperti mendengar suara-suara. Munculnya istilah "OKBD" yang tidak sepenuhnya relevan dapat dilihat sebagai efek samping dari penelusuran graf, di mana sistem mengambil node tetangga yang terhubung namun berada di luar konteks jawaban yang ideal. Meskipun demikian, hal ini menunjukkan pendekatan yang jauh lebih efektif daripada pencarian berbasis vektor seperti pada Naive RAG dan Vector Search.

Di sisi lain metode N-Shortest Path menghasilkan jawaban yang secara definisi umum sangat lengkap dan mencakup empat gejala utama (halusinasi, delusi, pemikiran/bicara tidak terorganisir, dan perilaku tidak terorganisir atau katatonik). Meskipun tampak informatif, jawaban ini merupakan bentuk halusinasi karena tidak bersumber da-

ri konteks yang diambil oleh *retriever*. *Retriever* gagal untuk mengambil konteks dari *Knowledge Graph* yang menyebabkan LLM memanfaatkan pengetahuan internalnya untuk mengisi kekosongan informasi. Kegagalan tersebut dapat dipicu oleh algoritma *shortest path* di mana tidak ditemukannya jalur di antara *nodes* yang terdeteksi. Hal tersebut dapat terjadi apabila memaksakan pertanyaan yang termasuk ke dalam *entity query* dalam kasus ini *"Apa yang dimaksud dengan gejala psikotik?"* yang menanyakan definisi akan suatu hal.

Dari kasus ini dapat disimpulkan bahwa kelengkapan jawaban tidak selalu berarti kualitas RAG yang baik. Metode yang terlihat "pintar" seperti N-Shortest Path memiliki kemungkinan untuk berhalusinasi jika gagal mengambil konteks yang relevan. Di sisi lain, metode Default dan Neighbor Expansion berhasil mengambil konteks relevan sehingga mampu memanfaatkannya untuk menghasilkan jawaban yang akurat dan terpercaya. Kedua metode tersebut menunjukkan kejujuran sistem, berlandaskan konteks, dan cenderung lebih *faithful* dan dapat dipercaya.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, metode RAG berbasis *Knowledge Graph* secara umum mengungguli Naive RAG yang berbasis potongan dokumen pada mayoritas metrik uji. Hasil evaluasi terhadap beberapa metode pengambilan konteks dari *Knowledge Graph* serta analisis kualitas jawaban akhir menghasilkan beberapa kesimpulan utama sebagai berikut:

1. Arsitektur RAG Berbasis *Knowledge Graph* terbukti mampu meningkatkan akurasi dan konsistensi respons. Penelitian ini berhasil merancang dan mengimplementasikan arsitektur *chatbot* yang mengintegrasikan RAG dengan KG. Hasil evaluasi menunjukkan bahwa arsitektur ini secara konsisten mengungguli pendekatan Naive RAG, yang dibuktikan dengan peningkatan correctness sebesar 8%, *relevancy* sebesar 17%, dan *faithfulness* sebesar 4%. Paningkatan ini menegaskan bahwa integrasi struktur graf mampu menghasilkan respons yang lebih akurat secara faktual, relevan, dan konsisten dengan basis pengetahuannya, sehingga meminimalkan masalah fundamental terkait risiko misinformasi.
2. Metode *Knowledge Extraction* yang dirancang berhasil mengonstruksi sebuah *Knowledge Graph* yang koheren dan kaya, terdiri dari 1.119 entitas unik dan 1.078 relasi yang diekstraksi dari korpus literatur. Efektivitas proses KE dalam memetakan hubungan kompleks dari literatur tervalidasi secara fungsional, yang dibuktikan oleh nilai *recall* (0,8868) dan *hit ratio* (0,9584) yang tinggi pada proses retrieval. Hal ini secara tidak langsung mengindikasikan bahwa struktur relasi yang diekstraksi cukup logis dan terorganisir dengan baik untuk memungkinkan penelusuran informasi yang mendalam dan menyeluruh.
3. Metode *retrieval* Default dan *Neighbor Expansion* menunjukkan performa paling konsisten dan unggul di antara seluruh strategi RAG-KG yang diuji. Keduanya terbukti mampu menemukan informasi relevan dari *Knowledge Graph* dan menghasilkan kualitas jawaban yang relatif lebih baik. Oleh karena itu, penelitian ini merekomendasikan kedua pendekatan tersebut sebagai pilihan utama untuk *Knowledge Retrieval* dalam arsitektur RAG berbasis KG.

Secara keseluruhan, penelitian ini menegaskan bahwa penggunaan *Knowledge Graph* sebagai fondasi pengetahuan dalam RAG dapat meningkatkan relevansi, akurasi, dan faktualitas respons *chatbot*. Dengan memanfaatkan sumber data berupa buku dan artikel terkait kesehatan mental, sistem yang dikembangkan mampu menyajikan jawaban

yang lebih informatif dan faktual, sehingga berpotensi meminimalkan risiko halusinasi saat berinteraksi dengan pengguna.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, berikut adalah beberapa saran untuk penelitian dan pengembangan selanjutnya:

1. Pada penelitian ini *dataset* evaluasi masih diperoleh berupa data sintetis hasil *prompting* menggunakan LLM. Penelitian selanjutnya disarankan menggunakan *dataset* pertanyaan dan jawaban yang telah divalidasi oleh ahli atau bersumber langsung dari sumber terpercaya agar evaluasi mencerminkan performa saat digunakan dalam kasus yang sebenarnya.
2. Mengingat belum adanya standar baku untuk struktur *Knowledge Graph* dan cara mengevaluasinya, penelitian mendatang dapat berfokus pada evaluasi dampak dari berbagai desain KG terhadap kualitas jawaban akhir. Evaluasi ini akan melengkapi fokus penelitian saat ini yang lebih terpusat pada metrik *retrieval* dan kualitas jawaban.
3. Perlu dilakukan eksplorasi lebih lanjut terhadap *tuning* parameter, seperti nilai top-k dan *max hop* pada proses *retrieval*, untuk menemukan konfigurasi optimal yang dapat menyeimbangkan antara kelengkapan dan relevansi konteks yang diambil.
4. Meskipun berkinerja kurang baik, konsep pencarian jalur seperti pada *N-Shortest Path* tetap memiliki potensi. Penelitian lebih lanjut dapat difokuskan pada penerapan mekanisme penyaringan atau pembobotan untuk memastikan hanya jalur yang paling relevan secara semantik dan kontekstual yang dipertimbangkan, sehingga dapat mengurangi risiko halusinasi.

DAFTAR PUSTAKA

[1] K. Enevoldsen *et al.*, "Mmteb: Massive multilingual text embedding benchmark," *arXiv preprint arXiv:2502.13595*, 2025. [Online]. Available: https://arxiv.org/abs/2502.13595

[2] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, and J. Larson, "From local to global: A graph rag approach to query-focused summarization," 2025. [Online]. Available: https://arxiv.org/abs/2404.16130

[3] G. . M. D. Collaborators, "Global, regional, and national burden of 12 mental disorders in 204 countries and territories, 1990–2019: a systematic analysis for the global burden of disease study 2019," *The Lancet Psychiatry*, vol. 9, no. 2, pp. 137–150, 2022.

[4] Center for Reproductive Health, University of Queensland, and Johns Hopkins Bloomberg School of Public Health, "Indonesia - national adolescent mental health survey (i-namhs): Laporan penelitian," Pusat Kesehatan Reproduksi, Yogyakarta, Indonesia, Tech. Rep., 2022, edisi pertama, Oktober 2022.

[5] World Health Organization, *World mental health report: transforming mental health for all*. Geneva: World Health Organization, 2022, licence: CC BY-NC-SA 3.0 IGO. Available: https://www.who.int/publications/i/item/9789240049338.

[6] S. Kelly, S.-A. Kaye, and O. Oviedo-Trespalacios, "A multi-industry analysis of the future use of ai chatbots," *Human Behavior and Emerging Technologies*, vol. 2022, no. 1, p. 2552099, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2022/2552099

[7] K. K. Fitzpatrick, A. Darcy, and M. Vierhile, "Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (woebot): A randomized controlled trial," *JMIR Ment Health*, vol. 4, no. 2, p. e19, Jun 2017. [Online]. Available: http://mental.jmir.org/2017/2/e19/

[8] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021. [Online]. Available: https://arxiv.org/abs/2005.11401

[9] X. Chen, S. Jia, and Y. Xiang, "A review: Knowledge reasoning over knowledge graph," *Expert systems with applications*, vol. 141, p. 112948, 2020.

[10] S. Choi and Y. Jung, "Knowledge graph construction: Extraction, learning, and evaluation," *Applied Sciences*, vol. 15, no. 7, 2025. [Online]. Available: https://www.mdpi.com/2076-3417/15/7/3727

[11] J. Weizenbaum, "Eliza—a computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.

[12] UGM, "Layanan chatbot "lintang"," 2023. [Online]. Available: https://hpu.ugm.ac.id/layanan-chatbot-lintang/

[13] H. Yu and S. McGuinness, "An experimental study of integrating fine-tuned llms and prompts for enhancing mental health support chatbot system," *Journal of Medical Artificial Intelligence*, pp. 1–16, 2024.

[14] B. Chen, Z. Zhang, N. Langrené, and S. Zhu, "Unleashing the potential of prompt engineering for large language models," 2025. [Online]. Available: https://arxiv.org/abs/2310.14735

[15] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy, "Challenges and applications of large language models," 2023. [Online]. Available: https://arxiv.org/abs/2307.10169

[16] T. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[17] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, T. Liu, B. Chang, X. Sun, L. Li, and Z. Sui, "A survey on in-context learning," 2024. [Online]. Available: https://arxiv.org/abs/2301.00234

[18] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, and J. Taylor, "Industry-scale knowledge graphs: Lessons and challenges: Five diverse technology companies show how it's done," *Queue*, vol. 17, no. 2, pp. 48–75, Apr. 2019. [Online]. Available: https://doi.org/10.1145/3329781.3332266

[19] R. Angles, "A comparison of current graph database models," in *2012 IEEE 28th International Conference on Data Engineering Workshops*, 2012, pp. 171–177.

[20] V. A. Traag, L. Waltman, and N. J. van Eck, "From louvain to leiden: guaranteeing well-connected communities," *Scientific Reports*, vol. 9, no. 1, p. 5233, 2019. [Online]. Available: https://doi.org/10.1038/s41598-019-41695-z

[21] E. D. Liddy, "Natural language processing," 2001.

[22] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: state of the art, current trends and challenges," *Multimedia Tools and Applications*, vol. 82, no. 3, pp. 3713–3744, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1007/s11042-022-13428-4

[23] D. A. Scherbakov, N. C. Hubig, L. A. Lenert, A. V. Alekseyenko, and J. S. Obeid, "Natural language processing and social determinants of health in mental health research: Ai-assisted scoping review," *JMIR Ment Health*, vol. 12, p. e67192, Jan 2025. [Online]. Available: https://mental.jmir.org/2025/1/e67192

[24] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to machine learning, neural networks, and deep learning," *Translational vision science & technology*, vol. 9, no. 2, pp. 14–14, 2020.

[25] I. Goodfellow, "Deep learning," 2016.

[26] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang *et al.*, "A survey on evaluation of large language models," *ACM transactions on intelligent systems and technology*, vol. 15, no. 3, pp. 1–45, 2024.

[27] H. R. Saeidnia, "Welcome to the gemini era: Google deepmind and the information industry," *Library Hi Tech News*, no. ahead-of-print, 2023.

[28] G. Comanici, E. Bieber, M. Schaekermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. Rosen *et al.*, "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities," *arXiv preprint arXiv:2507.06261*, 2025.

[29] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford *et al.*, "Gpt-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.

[30] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, 3rd ed., 2025, online manuscript released January 12, 2025. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[31] K. Bostrom and G. Durrett, "Byte pair encoding is suboptimal for language model pretraining," *arXiv preprint arXiv:2004.03720*, 2020.

[32] G. Strang, *Introduction to linear algebra*. Wellesley - Cambridge Press, 2009.

[33] P. Xia, L. Zhang, and F. Li, "Learning similarity with cosine similarity ensemble," *Information sciences*, vol. 307, pp. 39–52, 2015.

[34] P. N. Singh, S. Talasila, and S. V. Banakar, "Analyzing embedding models for embedding vectors in vector databases," in *2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG)*. IEEE, 2023, pp. 1–7.

[35] J. Beall, "The weaknesses of full-text searching," *The Journal of Academic Librarianship*, vol. 34, no. 5, pp. 438–444, 2008.

[36] MongoDB. (2025) Full-text search: What is it and how it works. Accessed: 2025-09-03. [Online]. Available: https://www.mongodb.com/resources/basics/full-text-search#what-is-fulltext-search

[37] Y. Han, C. Liu, and P. Wang, "A comprehensive survey on vector database: Storage and retrieval technique, challenge," *arXiv preprint arXiv:2310.11703*, 2023.

[38] S. Paul, A. Mitra, and C. Koner, "A review on graph database and its representation," in *2019 International Conference on Recent Advances in Energy-efficient Computing and Communication (ICRAECC)*, 2019, pp. 1–5.

[39] A. Singhal. (2012) Introducing the knowledge graph: things, not strings. 2020-11-13. [Online]. Available: https://www.blog.google/products/search/introducing-knowledge-graph-things-not/

[40] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic web*, vol. 8, no. 3, pp. 489–508, 2016.

[41] Y. Yao, Y. Zeng, N. Zhong, and X. Huang, "Knowledge retrieval (kr)," in *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*, 2007, pp. 729–735.

[42] D. M. W. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2020. [Online]. Available: https://arxiv.org/abs/2010.16061

[43] N. Craswell, "Mean reciprocal rank," in *Encyclopedia of database systems*. Springer, 2016, pp. 1–1.

[44] S. Es, J. James, L. E. Anke, and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, 2024, pp. 150–158.

LAMPIRAN

L.1 Spesifikasi API Sistem RAG

L.1.1 API: Process File Uploaded

Tabel L.1. Spesifikasi API Process File Uploaded

Properti	**Keterangan**
HTTP Method	`POST`
URL Path	`/api/v1/extraction/extract`
Deskripsi	Mengekstrak konten dari file dokumen (docx, html, pdf) untuk diproses dan disimpan ke dalam basis data graf.

Tabel L.2. Spesifikasi Parameter API Process File Uploaded

Parameter	**Tipe**	**Deskripsi**
`file_path`	`string`	Path menuju file yang akan diproses.
`file_type`	`string`	Pilihan tipe file: `'docx'`, `'html'`, atau `'pdf'`.
`split`	`boolean`	Jika `true`, dokumen akan dipecah menjadi beberapa bagian.

L.1.2 API: Get Answer from Knowledge Graph

Tabel L.3. Spesifikasi API Get Answer from KG

Properti	**Keterangan**
HTTP Method	`POST`
URL Path	`/api/v1/retrieval/get-answer`
Deskripsi	Menerima kueri pengguna dan mengambil jawaban berdasarkan penelusuran pada *Knowledge Graph*.

Tabel L.4. Spesifikasi Parameter API Get Answer from KG

Parameter	**Tipe**	**Deskripsi**
`query`	`string`	Pertanyaan dari pengguna.

Tabel L.4. Spesifikasi Parameter API Get Answer from KG (Lanjutan)

Parameter	Tipe	Deskripsi
`graph_traversal`	`string`	Metode penelusuran graf. Pilihan: `'neighbor_expansion'`, `'n-shortest_path'`, `'auto'`.
`search_method`	`string`	Metode pencarian entitas awal. Pilihan: `'vector'`, `'auto'`.
`max_hop`	`integer`	(Opsional) Lompatan maksimal untuk pencarian berbasis jalur. Default: 10.
`limit`	`integer`	(Opsional) Batas total entitas yang diambil. Default: 50.
`top_k`	`integer`	(Opsional) Jumlah entitas teratas yang dicari. Default: 3.

L.1.3 API: Generate Evaluation Dataset

Tabel L.5. Spesifikasi API Generate Evaluation Dataset

Properti	Keterangan
HTTP Method	`POST`
URL Path	`/api/v1/evaluation/generate-evaluation-dataset`
Deskripsi	Menghasilkan pasangan tanya-jawab (dataset evaluasi) dari potongan teks (*chunks*) yang spesifik.

Tabel L.6. Spesifikasi Parameter API Generate Evaluation Dataset

Parameter	Tipe	Deskripsi
`chunk_ids`	`array`	Daftar ID unik dari potongan teks yang akan dijadikan dasar pembuatan dataset.

L.1.4 API: Add Document to Vector DB

Tabel L.7. Spesifikasi API Add Document to Vector DB

Properti	**Keterangan**
HTTP Method	`POST`
URL Path	`/api/v1/extraction/extract-rag-vector`
Deskripsi	Memproses file dan menyimpannya sebagai *embedding* ke dalam *vector database* (untuk Naive RAG).

Tabel L.8. Spesifikasi Parameter API Add Document to Vector DB

Parameter	**Tipe**	**Deskripsi**
`file_path`	`string`	Path menuju file yang akan diproses.
`file_type`	`string`	Pilihan tipe file: `'docx'`, `'html'`, atau `'pdf'`.

L.1.5 API: Get Answer from Vector DB (Naive RAG)

Tabel L.9. Spesifikasi API Get Answer from Vector DB

Properti	**Keterangan**
HTTP Method	`POST`
URL Path	`/api/v1/retrieval/get-answer-vector-rag`
Deskripsi	Menerima kueri dan mengambil jawaban menggunakan metode Naive RAG (pencarian vektor).

Tabel L.10. Spesifikasi Parameter API Get Answer from Vector DB

Parameter	**Tipe**	**Deskripsi**
`query`	`string`	Pertanyaan dari pengguna.
`top_k`	`integer`	(Opsional) Jumlah potongan dokumen teratas yang relevan untuk diambil sebagai konteks. Default: 5.

L.2 Prompt

L.2.1 Prompt untuk Ekstraksi Entitas dan Relasi

```
"""
INSTRUKSI:
Anda adalah sebuah pipeline ekstraksi Knowledge Graph untuk
   domain kesehatan mental. Dari DOKUMEN yang diberikan, ekstrak
    semua entitas penting dan relasi di antara mereka. Patuhi
   struktur JSON dan daftar tipe yang telah ditentukan dengan
   ketat.

LANGKAH-LANGKAH:
1.  **Ekstrak Entitas**: Identifikasi entitas, klasifikasikan
   tipenya, dan berikan deskripsi yang dapat berupa potongan
   dari teks asli. Deskripsi dapat meliuti nomor telepon, alamat
   , atau informasi penting dari entitas.
2.  **Ekstrak Relasi**: Identifikasi hubungan langsung antar
   entitas tersebut. `name` relasi harus berupa frasa kerja dari
    teks, dan `type` adalah kategorisasi formalnya.

---

### **Sistem Tipe (Ontologi)**

**A. Tipe Entitas yang Diizinkan:**
* **Gangguan & Kondisi**: `Gangguan_Mental`, `Gejala`, `
   Kondisi_Medis_Terkait`
* **Intervensi & Perawatan**: `Terapi_Psikologis`, `Obat_Medis`,
    `Layanan_Kesehatan`, `Aktivitas_Kesejahteraan`, `
   Penyedia_Layanan`
* **Aktor & Pemangku Kepentingan**: `Profesional_Kesehatan`, `
   Organisasi`, `Individu`
* **Faktor & Konteks**: `Faktor_Risiko`, `Faktor_Pelindung`, `
   Dokumen_Hukum_Kebijakan`, `Lokasi`
* **Data & Konsep**: `Data_Statistik`, `Konsep_Abstrak`
* **Jaring Pengaman**: `Lainnya` (Gunakan untuk entitas penting
   yang tidak cocok dengan kategori lain)

**B. Tipe Relasi yang Diizinkan:**
* **Sebab-Akibat**: `Menyebabkan`, `Berkontribusi_Pada`, `
   Mengurangi_Risiko`, `Merupakan_Gejala_Dari`
* **Intervensi**: `Mendiagnosis`, `Menangani`, `Meresepkan`, `
```

```
   Menawarkan_Layanan`
* **Hierarki & Keanggotaan**: `Adalah_Jenis_Dari`, `Bekerja_Di`,
    `Berlokasi_Di`, `Bagian_Dari`
* **Deskriptif**: `Memiliki_Atribut`, `Didasarkan_Pada`, `
   Direkomendasikan_Untuk`
* **Jaring Pengaman**: `Terkait_Dengan` (Gunakan untuk hubungan
   yang jelas tapi tidak cocok kategori lain)

---

### **Struktur Output JSON**
{
"entities": [
    {
    "name": "nama_entitas",
    "type": "tipe_entitas_dari_daftar_di_atas",
    "description": "penjelasan berisi informasi penting dari
   entitas, dapat berupa kutipan asli dari dokumen atau
   ringkasannya"
    }
],
"relations": [
    {
    "source_entity": "nama_entitas_sumber",
    "target_entity": "nama_entitas_target",
    "name": "frasa_kerja_dari_dokumen",
    "type": "tipe_relasi_dari_daftar_di_atas"
    }
]
}

---

### **CONTOH EKSEKUSI**

**DOKUMEN:**
Kesehatan Jiwa adalah kondisi dimana seorang individu dapat
   berkembang secara fisik, mental, spiritual, dan sosial. K
arakteristik gangguan jiwa secara umum yaitu kombinasi pikiran
   yang abnormal, emosi, dan persepsi.
Faktor psikologis seperti trauma yang mengakibatkan stres dan
```

```
   faktor biologis seperti genetik merupakan faktor yang
   berkontribusi terhadap terjadinya gangguan jiwa.
Sebesar 50% gangguan jiwa berawal pada usia 14 tahun.

Layanan Psikologi UGM dan Fakultas

Klinik GMC (Mental Health Support)
Psikolog: +62 813-2620-0342
GMC: +62 851-0047-3123

Layanan Psikologi Fakultas (hanya untuk sivitas fakultas yang
   bersangkutan)
Biologi: form registrasi
Ekonomika dan Bisnis: Telp: +62 811-2843-884 atau isi Form
   Registrasi melalui Portal SINTESIS di menu Layanan Konsultasi
   .

**JSON_OUTPUT:**
{
"entities": [
    {
    "name": "Kesehatan Jiwa",
    "type": "Konsep_Abstrak",
    "description": "kondisi dimana seorang individu dapat
   berkembang secara fisik, mental, spiritual, dan sosial."
    },
    {
    "name": "Gangguan Jiwa",
    "type": "Gangguan_Mental",
    "description": "Sebuah kondisi yang memiliki karakteristik
   seperti kombinasi pikiran abnormal, dan dipengaruhi oleh
   faktor psikologis serta biologis."
    },
    {
    "name": "Karakteristik Gangguan Jiwa",
    "type": "Karakteristik",
    "description": "kombinasi pikiran yang abnormal, emosi, dan
   persepsi."
    },
    {
    "name": "Faktor Psikologis",
```

```
 "type": "Faktor_Risiko",
 "description": "Faktor yang dapat mengakibatkan stres,
contohnya trauma."
 },
 {
 "name": "Faktor Biologis",
 "type": "Faktor_Risiko",
 "description": "Faktor yang berkontribusi pada gangguan jiwa
, contohnya genetik."
 },
 {
 "name": "Stres",
 "type": "Gejala",
 "description": "Sebuah kondisi yang diakibatkan oleh faktor
psikologis seperti trauma."
 },
 {
 "name": "Statistik Onset Gangguan Jiwa",
 "type": "Data_Statistik",
 "description": "Sebesar 50% gangguan jiwa berawal pada usia
14 tahun."
 },
 {
 "name": "Layanan Kesehatan Mental Rumah Sakit Akademik UGM",
 "type": "Penyedia_Layanan",
 "description": "Layanan kesehatan mental yang disediakan
oleh Rumah Sakit Akademik (RSA) UGM. Kontak dapat dilakukan
melalui telepon di +62 811-2548-118."
 },
 {
 "name": "Klinik GMC - Mental Health Support",
 "type": "Penyedia_Layanan",
 "description": "Layanan dukungan kesehatan mental dari
Klinik GMC. Kontak dapat ditujukan ke Psikolog di +62 813-262
0-0342 atau ke nomor umum GMC di +62 851-0047-3123."
 },
 {
 "name": "Layanan Psikologi Fakultas Biologi",
 "type": "Penyedia_Layanan",
 "description": "Layanan psikologi khusus untuk sivitas
Fakultas Biologi UGM. Pendaftaran dilakukan dengan mengisi
```

```
   form registrasi yang tersedia."
    },
    {
    "name": "Layanan Konsultasi Fakultas Ekonomika dan Bisnis",
    "type": "Penyedia_Layanan",
    "description": "Layanan konsultasi khusus untuk sivitas
   Fakultas Ekonomika dan Bisnis UGM. Pendaftaran dapat
   dilakukan melalui telepon di +62 811-2843-884 atau dengan
   mengisi Form Registrasi via Portal SINTESIS."
    }
],
"relations": [
    {
    "source_entity": "Gangguan Jiwa",
    "target_entity": "Karakteristik Gangguan Jiwa",
    "name": "memiliki karakteristik",
    "type": "Memiliki_Atribut"
    },
    {
    "source_entity": "Faktor Psikologis",
    "target_entity": "Stres",
    "name": "mengakibatkan",
    "type": "Menyebabkan"
    },
    {
    "source_entity": "Faktor Psikologis",
    "target_entity": "Gangguan Jiwa",
    "name": "berkontribusi terhadap",
    "type": "Berkontribusi_Pada"
    },
    {
    "source_entity": "Faktor Biologis",
    "target_entity": "Gangguan Jiwa",
    "name": "berkontribusi terhadap",
    "type": "Berkontribusi_Pada"
    },
    {
    "source_entity": "Gangguan Jiwa",
    "target_entity": "Statistik Onset Gangguan Jiwa",
    "name": "memiliki data onset",
    "type": "Memiliki_Atribut"
```

```
    },
    {
    "sumber": "Klinik GMC - Mental Health Support",
    "target": "Layanan Kesehatan Mental Universitas Gadjah Mada"
   ,
    "tipe": "Bagian_Dari"
    },
    {
    "sumber": "Layanan Psikologi Fakultas Biologi",
    "target": "Layanan Kesehatan Mental Universitas Gadjah Mada"
   ,
    "tipe": "Bagian_Dari"
    },
    {
    "sumber": "Layanan Konsultasi Fakultas Ekonomika dan Bisnis"
   ,
    "target": "Layanan Kesehatan Mental Universitas Gadjah Mada"
   ,
    "tipe": "Bagian_Dari"
    },
]
}
"""
```

L.2.2 Prompt untuk Ekstraksi Entitas dan Klasifikasi Kueri

```
"""
Tugas: Klasifikasikan kueri pengguna ke dalam kategori '
   entity_query' atau 'path_query', dan ekstrak semua entitas
   yang relevan.

Definisi:
'entity_query' (Query Atribut/Node): Pertanyaan yang fokus untuk
    mendeskripsikan atau mengambil properti/atribut dari sebuah
   konsep sentral. Contoh: "Apa itu X?", "Apa saja gejala dari X
   ?".
'path_query' (Query Relasi/Edge): Pertanyaan tentang hubungan
   antara dua atau lebih entitas, atau mencari alur/proses.
   Contoh: "Bagaimana hubungan A dan B?", "Cari A yang terkait
   dengan B?".

Output Format (JSON):
```

```
{{
"category": "entity_query" | "path_query",
"entities": ["Entity 1", "Entity 2", ..., "Entity N"]
}}

Contoh:

Query: "Apa itu depresi dan bagaimana gejalanya?"
Output: {{"category": "entity_query", "entities": ["depresi"]}}

Query: "Apa saja jenis obat antidepresan yang umum diresepkan?"
Output: {{"category": "entity_query", "entities": ["obat
   antidepresan"]}}

Query: "Bagaimana cara kerja terapi kognitif-perilaku (CBT)
   untuk kecemasan?"
Output: {{"category": "path_query", "entities": ["terapi
   kognitif-perilaku (CBT)", "kecemasan"]}}

Query: "Jelaskan peran serotonin dalam gangguan mood."
Output: {{"category": "path_query", "entities": ["serotonin", "
   gangguan mood"]}}

Query: "Bisakah olahraga membantu mengurangi stres dan
   meningkatkan kesehatan mental?"
Output: {{"category": "path_query", "entities": ["olahraga", "
   stres", "kesehatan mental"]}}

Query: "Layanan dukungan kesehatan mental apa saja yang tersedia
    untuk remaja di Jakarta?"
Output: {{"category": "path_query", "entities": ["layanan
   dukungan kesehatan mental", "remaja", "Jakarta"]}}

Query: "Siapa psikolog yang spesialisasi di terapi trauma?"
Output: {{"category": "path_query", "entities": ["psikolog", "
   terapi trauma"]}}

---
Klasifikasikan dan ekstrak entitas untuk kueri berikut:

Query:
```

```
{query}
"""
```

L.2.3 Prompt untuk Membuat Dataset Evaluasi

```
  """
Anda adalah seorang ahli pembuatan data, ditugaskan untuk
   membuat dataset evaluasi "Silver Standard" untuk chatbot
   kesehatan mental.
Respons Anda HARUS berupa daftar kamus (list of dictionaries)
   dalam format JSON yang valid. Jangan tambahkan teks pengantar
    atau penjelasan di luar struktur JSON.

**KONTEKS & ATURAN:**
1.  **Sumber Kebenaran:** Anda HANYA boleh menggunakan informasi
    yang disediakan di bagian 'KONTEKS DOKUMEN' dan 'NODE
   KNOWLEDGE GRAPH' di bawah ini. Jangan gunakan pengetahuan
   eksternal.
2.  **Node Knowledge Graph:** Bagian 'NODE KNOWLEDGE GRAPH'
   menyediakan daftar node yang tersedia beserta nama dan
   deskripsinya. Untuk 'golden_nodes', Anda HANYA boleh
   menggunakan 'name' dari node-node tersebut. Gunakan '
   description' untuk memahami arti setiap node dan membuat
   pilihan yang lebih baik.
3.  **Tugas:** Hasilkan minimal {minimum} atau lebih contoh
   evaluasi yang beragam dalam format JSON yang ditentukan.
4.  **Keragaman Query:** Buat campuran `entity_query` (apa itu X
   ?) dan `path_query` (bagaimana/mengapa X terkait dengan Y?).
5.  **Gaya Kueri:** 'query' harus mencerminkan pertanyaan
   pengguna yang ingin mengetahui tentang suatu hal. Buatlah
   sealami mungkin.
5.  **Gaya Jawaban:** 'golden_answer' harus empatik, aman, dan
   sangat berdasarkan pada KONTEKS yang disediakan. Jawaban
   tidak boleh memberikan nasihat medis.

---
**KONTEKS DOKUMEN:**
{doc}
---
**NODE KNOWLEDGE GRAPH:**
{nodes_str}
```

```
---

**CONTOH FEW-SHOT (Ikuti format dan gaya ini):**
{few_shot_examples}
---

**TUGAS ANDA:**
Sekarang, hasilkan minimal {minimum} contoh baru yang unik
   berdasarkan semua aturan dan konteks yang diberikan. Pastikan
    output Anda adalah sebuah list JSON tunggal.
"""
```

L.3 Contoh Response

Beriku merupakan contoh response untuk kueri *"Apa itu Gangguan Mental Emosional?"*:


```
{
"message": "Sucessfull",
"data": {
  "latency": 5.0608,
  "graph_traversal": "neighbor_expansion",
  "query_class": {
    "category": "entity_query",
    "entities": [
      "Gangguan Mental Emosional"
      ]
  },
  "answer": "Gangguan Mental Emosional (GME) adalah suatu
   kondisi yang mengindikasikan seseorang mengalami perubahan
   psikologis. Perubahan ini bisa merupakan kondisi normal,
   namun dapat juga merupakan kondisi patologis.\n\nPenting
   untuk diketahui bahwa Gangguan Mental Emosional **bukanlah
   diagnosis gangguan jiwa**. Meskipun demikian, kondisi ini
   merupakan masalah kesehatan jiwa yang banyak dijumpai di
   Indonesia, diperkirakan diderita oleh 19 juta orang.\n\nGME
   dapat memengaruhi aktivitas sehari-hari seseorang, yang pada
   akhirnya berdampak pada menurunnya produktivitas dan kualitas
    hidup. Istilah ini juga digunakan dalam \"Petunjuk Teknis
   Pencegahan dan Pengendalian Gangguan Mental Emosional\"
   sebagai subjek utama panduan tersebut.",
  "knowledge": "Berikut adalah informasi yang ditemukan dari
   knowledge graph:
    ### Entitas: Gangguan Mental Emosional (Tipe: ['Entity', '
   Gangguan_Mental'])
    - Deskripsi: Bukan diagnosis gangguan jiwa. Dapat
   memengaruhi aktifitas sehari-hari yang berdampak pada
   menurunnya produktifitas dan kualitas hidup. Kondisi atau
   gangguan yang menjadi subjek utama dari 'Petunjuk Teknis
   Pencegahan dan Pengendalian Gangguan Mental Emosional'.
   Masalah kesehatan jiwa yang banyak dijumpai di Indonesia,
   diperkirakan diderita oleh 19 juta orang. Masalah kesehatan
   jiwa yang banyak terjadi di masyarakat. Suatu kondisi yang
   mengindikasikan seseorang mengalami perubahan psikologis yang
    mungkin merupakan sebuah kondisi normal, tetapi dapat juga
```

```
merupakan kondisi patologis.
 - Skor Relevansi Awal: 5.07
 - Hubungan Terkait:
   - [OUTGOING] --[merupakan|Tipe: Adalah_Jenis_Dari]-->
Masalah Kesehatan Jiwa (Tipe: ['Entity', 'Gangguan_Mental'] |
 Deskripsi: "Kondisi yang dapat dicegah melalui upaya
promotif dan preventif.")
   - [OUTGOING] --[merupakan|Tipe: Adalah_Jenis_Dari]-->
Masalah Kesehatan Jiwa (Tipe: ['Entity', 'Konsep_Abstrak'] |
Deskripsi: "Istilah umum untuk kondisi yang mencakup tekanan
emosional atau gangguan mental emosional. Kondisi kesehatan
jiwa umum yang mencakup gangguan mental emosional.")
     - [OUTGOING] --[bukanlah diagnosis|Tipe:
Adalah_Jenis_Dari]--> Gangguan Jiwa (Tipe: ['Entity', '
Gangguan_Mental'] | Deskripsi: "Berbagai macam gangguan
dengan gejala yang berbeda-beda, memiliki karakteristik
kombinasi pikiran yang abnormal, emosi, persepsi, perilaku
dan hubungan dengan orang lain. Kondisi kesehatan mental yang
 menjadi objek analisa epidemiologi. Kondisi yang berbeda
dengan gangguan mental emosional, dan perlu dicegah
terjadinya. Kondisi yang dapat dicegah dengan deteksi dan
intervensi dini gangguan mental emosional.")
     - [OUTGOING] --[dapat merupakan|Tipe: Adalah_Jenis_Dari
]--> Kondisi Patologis (Tipe: ['Entity', '
Kondisi_Medis_Terkait'] | Deskripsi: "Kondisi yang mungkin
terjadi pada seseorang yang mengalami perubahan psikologis.")
     - [INCOMING] <--[mencegah|Tipe: Mengurangi_Risiko]--
Upaya Promosi Kesehatan (Tipe: ['Entity', '
Aktivitas_Kesejahteraan'] | Deskripsi: "Kegiatan terintegrasi
 untuk mencegah dan mengendalikan GME, termasuk promosi
kesehatan sesuai alur layanan GME dan promosi kesehatan yang
lebih luas. Upaya yang diperlukan untuk mencegah gangguan
mental emosional.")
     - ...

 ### Entitas: Deteksi Gangguan Mental Emosional (Tipe: ['
Entity', 'Layanan_Kesehatan'])
 - Deskripsi: Proses identifikasi gangguan mental emosional
yang diharapkan mampu dilakukan oleh fasilitas pelayanan
kesehatan primer.
 - Skor Relevansi Awal: 4.52
```

```
  - Hubungan Terkait:
    - [INCOMING] <--[mengurangi|Tipe: Mengurangi_Risiko]--
  Keluhan fisik (Tipe: ['Entity', 'Gejala'] | Deskripsi: "Jenis
   keluhan yang sering dibawa penderita gangguan mental
  emosional ke fasilitas pelayanan kesehatan, menyebabkan
  ketidakdeteksian kondisi.")
    - [INCOMING] <--[mampu melakukan|Tipe: Menawarkan_Layanan
  ]-- Fasilitas pelayanan kesehatan primer (Tipe: ['Entity', '
  Penyedia_Layanan'] | Deskripsi: "Lini terdepan yang
  diharapkan mampu melakukan deteksi gangguan mental emosional
  dan intervensi dini.")
    - [OUTGOING] --[dapat segera diintervensi|Tipe:
  Direkomendasikan_Untuk]--> Intervensi (Tipe: ['Entity', '
  Layanan_Kesehatan'] | Deskripsi: "Tindakan yang dapat
  dilakukan segera setelah deteksi gangguan mental emosional
  untuk mencegah terjadinya gangguan jiwa.")

   ### Entitas: Penderita Gangguan Mental Emosional (Tipe: ['
  Entity', 'Individu'])
   - Deskripsi: Sebagian besar individu yang mengalami gangguan
   mental emosional, sering datang ke fasilitas pelayanan
  kesehatan dengan keluhan fisik dan tidak terdeteksi.
   - Skor Relevansi Awal: 4.52
   - Hubungan Terkait:
    - [OUTGOING] --[datang ke|Tipe: Terkait_Dengan]-->
  Fasilitas pelayanan kesehatan (Tipe: ['Entity', '
  Penyedia_Layanan'] | Deskripsi: "Tempat penderita gangguan
  mental emosional sering datang dengan keluhan-keluhan fisik
  sehingga tidak terdeteksi.")
    - [OUTGOING] --[dengan|Tipe: Memiliki_Atribut]--> Keluhan
  fisik (Tipe: ['Entity', 'Gejala'] | Deskripsi: "Jenis keluhan
   yang sering dibawa penderita gangguan mental emosional ke
  fasilitas pelayanan kesehatan, menyebabkan ketidakdeteksian
  kondisi.")"
"graph": [
  {
    "central_id": "070d75c6-3ae8-41d1-a4ca-c2c09ffab9a5",
    "central_name": "Gangguan Mental Emosional",
    "central_type": [
      "Entity",
      "Gangguan_Mental"
```

```
      ],
    "central_description": "Bukan diagnosis gangguan jiwa.
Dapat memengaruhi aktifitas sehari-hari yang berdampak pada
menurunnya produktifitas dan kualitas hidup. Kondisi atau
gangguan yang menjadi subjek utama dari 'Petunjuk Teknis
Pencegahan dan Pengendalian Gangguan Mental Emosional'.
Masalah kesehatan jiwa yang banyak dijumpai di Indonesia,
diperkirakan diderita oleh 19 juta orang. Masalah kesehatan
jiwa yang banyak terjadi di masyarakat. Suatu kondisi yang
mengindikasikan seseorang mengalami perubahan psikologis yang
 mungkin merupakan sebuah kondisi normal, tetapi dapat juga
merupakan kondisi patologis.",
    "score": 5.068589210510254,
    "neighborhood": [
      {
        "relation": {
          "direction": "OUTGOING",
          "name": "merupakan",
          "type": "Adalah_Jenis_Dari"
        },
        "neighbor": {
          "id": "0271758d-dd70-4cbe-bc5f-f12acdebf6dc",
          "description": "Kondisi yang dapat dicegah melalui
upaya promotif dan preventif.",
          "name": "Masalah Kesehatan Jiwa",
          "type": [
            "Entity",
            "Gangguan_Mental"
            ]
          }
        },
        {
        "relation": {
          "direction": "OUTGOING",
          "name": "merupakan",
          "type": "Adalah_Jenis_Dari"
        },
        "neighbor": {
          "id": "de2a5ef9-e0a8-4692-8773-2979257c826b",
          "description": "Istilah umum untuk kondisi yang
mencakup tekanan emosional atau gangguan mental emosional.
```

```
    Kondisi kesehatan jiwa umum yang mencakup gangguan mental
    emosional.",
              "name": "Masalah Kesehatan Jiwa",
              "type": [
                "Entity",
                "Konsep_Abstrak"
                ]
              }
            }, ...
        ]
      }, ...
  ]
}
}
```

L.4 Kode

Repositori kode dapat diakses pada https://github.com/gigahidjrikaaa/UGM-AICare.git.

L.4.1 Fungsi Ekstraksi Entitas dan Relasi dari Dokumen

```
from fastapi import APIRouter, HTTPException, Depends, Body
from pydantic import BaseModel
from datetime import datetime
import logging
import time
import asyncio
import os

from src.service.data__ingestion import DataIngestion
from src.service.llm import LLMService
from src.service.graph import GraphService
from src.service.vector_db_service import VectorDBService,
   Document
from src.model.schema import EntityRelationResponse, Entity,
   Relation
from src.utils.extraction import remove_duplicates_by_keys,
   resolve_entities_with_merged_descriptions, remap_relations


router = APIRouter()
logger = logging.getLogger(__name__)
async def get_data_ingestion_service():
  return DataIngestion()

async def get_llm_service():
  return LLMService()

async def get_graph_service():
  return GraphService(llm_service = await get_llm_service())

async def get_vector_service():
  return VectorDBService(llm_service = await get_llm_service())

class ExtractEntitiesBody(BaseModel):
  file_path: str
  file_type: str
  split: bool = True
```

```
@router.post("/extract")
async def extract_entities_relations(
  body: ExtractEntitiesBody = Body(...),
  data_ingestion_service: DataIngestion = Depends(
   get_data_ingestion_service),
  llm_service: LLMService = Depends(get_llm_service),
  graph_service: GraphService = Depends(get_graph_service)
):
  """Extract entities from text"""
  start_time = time.time()

  try:
    file_path = body.file_path
    file_type = body.file_type
    split = body.split
    logger.info(f"{file_path}, {file_type}")
    logger.info("Entity  Relation Extraction Started")

    texts = data_ingestion_service.extract_text_from_files(
   file_path=file_path, type=file_type, split=split)



    chunk_data = []
    for index, text in enumerate(texts):
        file_base = os.path.splitext(os.path.basename(file_path)
   )[0]
        file_name = f"{file_base}_{datetime.now().strftime('%Y%m
   %d_%H%M%S')}_chunk{index+1}.txt"

        with open(f"./data/{file_name}", "w") as f:
            f.write(text)

        chunk_data.append((text, file_name))

    extraction_tasks = [llm_service.
   extract_entities_and_relations(text=text, chunk_id=file_name)
    for text, file_name in chunk_data]
    chunk_results: list[EntityRelationResponse] = await asyncio.
   gather(*extraction_tasks, return_exceptions=True)
```

```
 all_results = EntityRelationResponse(entities=[], relations
=[])

 for res in chunk_results:
   if isinstance(res, Exception):
     logger.error(f"An error occurred during chunk extraction
: {res}")
     continue
   if res:
     all_results.entities.extend(res.entities)
     all_results.relations.extend(res.relations)

 deduped_entities, entity_map =
resolve_entities_with_merged_descriptions(entities=
all_results.entities)
 remapped_relations = remap_relations(relations=all_results.
relations, entity_name_map=entity_map)
 deduped_relations: list[Relation] =
remove_duplicates_by_keys(remapped_relations, keys=["
source_entity", "target_entity", "name", "type"])

 logger.info(f"Extracted {len(deduped_entities)} entities and
 {len(deduped_relations)} relations.")

 source_embedding = [
    f'{entity.name}, {entity.type}, {entity.description}'
    for entity in deduped_entities
    ]

 embeddings = await llm_service.get_embeddings(input=
source_embedding, task="RETRIEVAL_DOCUMENT")

 for entity, embedding in zip(deduped_entities, embeddings):
   entity.embedding = embedding

 insert_entity = await graph_service.insert_entities(entities
=deduped_entities)
 insert_relation = await graph_service.insert_relations(
relations=deduped_relations)
```

```
    processing_time = time.time() - start_time
    return {
      "message": "successful",
      "processing_time": processing_time,
      "duplicate_entity": len(entity_map) - len(deduped_entities
   ),
      "total_entities": len(deduped_entities),
      "total_relation": len(deduped_relations),
      "data": {
        "entities": deduped_entities,
        "relations": deduped_relations
        }
        }

  except Exception as e:
    logger.error(f"Fail to Extract Entities and Relations from
   Documen {body.file_path} : {str(e)}")
    raise HTTPException(status_code=500, detail=str(e))

class RAGExtractBody(BaseModel):
  file_path: str
  file_type: str
  chunk_size: int = 1000
  chunk_overlap: int = 200

class AddDocumentsResponse(BaseModel):
    success: bool
    message: str
    chunks_added: int
    documents_processed: int



@router.post("/extract-rag-vector")
async def add_document(
  body: RAGExtractBody = Body(...),
  data_ingestion_service: DataIngestion = Depends(
   get_data_ingestion_service),
  vector_service: VectorDBService = Depends(get_vector_service)
):
  try:
```

```
    file_path      = body.file_path
    file_type      = body.file_type
    chunk_size     = body.chunk_size
    chunk_overlap = body.chunk_overlap
    logger.info(f"{file_path}, {file_type}")
    logger.info("Add Document to Vector DB Started")

    input_docs = data_ingestion_service.extract_text_from_files(
  file_path=file_path, type=file_type, split=False)
    metadata = {
      "file_name": file_path
    }

    documents = []
    for doc_input in input_docs:
      doc = Document(
        content=doc_input,
        metadata=metadata,
        doc_id=file_path
        )
      documents.append(doc)

    result = await vector_service.add_documents(documents)

    return AddDocumentsResponse(**result)
  except Exception as e:
    logger.error(f"Fail to add document to RAG database: {e}")
    raise HTTPException(status_code=500, detail=str(e))
```

L.4.2 Fungsi Full-Text Search

```
async def _find_by_fulltext(self, name: str, top_k: int = 5) ->
   list[dict]:
    try:
        query = """
        CALL db.index.fulltext.queryNodes('entityNameIndex',
   $name)
        YIELD node, score
        RETURN node.name AS name, node.id AS id, node.type AS
   type, node.description AS description, score
        ORDER BY score DESC
```

```
        LIMIT $top_k
        """
        result = await neo4j_conn.execute_query(
            query=query,
            parameters={
                "name": name,
                "top_k": top_k
                }
        )
        logger.info(f"found {len(result)} entities using
   fulltext search")
        return [dict(row) for row in result] if result else []
     except Exception as e:
       logger.warning(f"Cannot find Entity by fulltext {e}")
         return []
```

L.4.3 Fungsi Vector Search

```
async def _find_by_vector(self, name: str, query: str, top_k:
   int = 3) -> list[dict]:
  try:
    embedding = await self.llm_service.get_embeddings(input=[f"{
   name}, {query}"], task="RETRIEVAL_QUERY")
    if not embedding or not embedding[0]:
        return None

    query = """
    CALL db.index.vector.queryNodes('entityIndex', $top_k,
   $embedding)
    YIELD node, score
    RETURN node.name AS name, node.id AS id, node.type AS type,
   node.description AS description, score
    """
    result = await neo4j_conn.execute_query(
        query=query,
        parameters={
            "embedding" : embedding[0],
            "top_k"     : top_k
            }
    )
    logger.info(f"found {len(result)} entities using vector
   search")
```

```
    return [dict(row) for row in result] if result else []
except Exception as e:
    logger.warning(f"Cannot find Entity by vector {e}")
    return []
```

L.4.4 Fungsi Neighbor Expansion Graph Traversal

```
async def NeighborExpansion(
    self,
    query: str,
    candidate_entities: list[str],
    search_method: Literal['vector', 'auto'] = 'auto',
    top_k: int = 3,
    limit: int = 50,
) -> list[dict]:
    if not isinstance(candidate_entities, list) or not
   candidate_entities:
        raise ValueError("candidate_entities must be a non-empty
    list")

    try:
        logger.info(
            f"Starting entity-based neighbor search for {len(
   candidate_entities)} "
            f"candidates using search method '{search_method}'"
        )

        # Entity extraction
        if search_method == 'vector':
            entities = await self.find_entity(query=query, top_k
   =top_k, method='vector')
        else:
            entities_nested = await asyncio.gather(*[
                self.find_entity(entity=e, query=query, top_k=
   top_k, method='auto')
                for e in candidate_entities
            ])
            entities = [
                item
                for sublist in entities_nested if sublist
                for item in sublist
            ]
```

```
    # Deduplicate entities
    unique_entities_map = {item['id']: item for item in
entities}
    unique_entities = list(unique_entities_map.values())

    if not unique_entities:
        logger.warning("No valid entities found from
candidates")
        return []

    logger.info(f"Found {len(unique_entities)} valid
entities in total from candidates")

    # Cypher query
    query_cypher = """
    UNWIND $entities AS entity
    MATCH (central:Entity)
    WHERE central.id = entity.id

    CALL (central) {
        OPTIONAL MATCH (central)-[r]-(neighbor)
        RETURN COLLECT({
            neighbor: {
                id: neighbor.id,
                name: neighbor.name,
                type: labels(neighbor),
                description: neighbor.description
            },
            relation: {
                name: r.name,
                type: type(r),
                direction: CASE
                    WHEN startNode(r).id = central.id
                    THEN 'OUTGOING'
                    ELSE 'INCOMING'
                END
            }
        })[..$neighbor_limit] AS limited_neighborhood
    }
```

```
        RETURN
            central.id AS central_id,
            central.name AS central_name,
            labels(central) AS central_type,
            central.description AS central_description,
            entity.score AS score,
            limited_neighborhood AS neighborhood
        LIMIT $limit
        """

        parameters = {
            "entities": unique_entities,
            "limit": limit,
            "neighbor_limit": 20,
        }

        # Execute query
        result = await neo4j_conn.execute_query(
            query=query_cypher,
            parameters=parameters
        )

        records = result.record if hasattr(result, "record")
    else result
        logger.info(
            f"Entity-based neighbor search completed, returned {
    len(records)} results"
        )

        return [dict(record) for record in records]

     except Exception as e:
        logger.error(f"Entity-based neighbor search failed: {e}"
    )
        return []
```

L.4.5 Fungsi N-Shortest Path Graph Traversal

```
async def N_ShortestPath(
    self,
    query: str,
    search_method: Literal['vector', 'auto'] = 'auto',
```

```
    candidate_entities: list = [],
    max_hop: int = 10,
    paths_per_group: int = 5,
    top_k: int = 2,
) -> list[dict]:
    try:
        logger.info(
            f"Starting entity-based neighbor search for {len(
  candidate_entities)} candidates"
        )

        # Entity extraction
        if search_method == 'vector':
            entities = await self.find_entity(query=query, top_k
  =top_k, method='vector')
            entities_nested = [entities]
        else:
            entities_nested = await asyncio.gather(*[
                self.find_entity(entity=e, query=query, top_k=
  top_k, method='auto')
                for e in candidate_entities
            ])
            if len(entities_nested) == 1:
                entities_nested = [[entity] for entity in
  entities_nested[0]]

            entities = [
                item
                for sublist in entities_nested if sublist
                for item in sublist
            ]

        # Deduplicate entities
        unique_entities_map = {item['id']: item for item in
  entities}
        unique_entities = list(unique_entities_map.values())

        if not unique_entities:
            logger.warning("No valid entities found from
  candidates")
            return []
```

```
        logger.info(f"Found {len(unique_entities)} valid
entities in total from candidates")

        # Group entities
        entity_groups = [
            [entity['id'] for entity in group]
            for group in entities_nested
            if group
        ]

        # Cypher query for shortest paths
        query_cypher = f"""
        UNWIND range(0, size($entity_groups)-1) AS i
        UNWIND range(i+1, size($entity_groups)-1) AS j
        WITH i, j, $entity_groups[i] AS source_candidates,
$entity_groups[j] AS target_candidates

        UNWIND source_candidates AS source_id
        UNWIND target_candidates AS target_id
        WITH i, j, source_id, target_id
        WHERE source_id <> target_id

        MATCH (source:Entity {{id: source_id}})
        MATCH (target:Entity {{id: target_id}})
        MATCH p = allShortestPaths((source)-[*1..{max_hop}]-(
target))
        WHERE ALL(n IN nodes(p) WHERE n.id IS NOT NULL)

        WITH i AS source_group_index,
             p, source, target,
             nodes(p) AS path_nodes_raw,
             relationships(p) AS path_rels_raw,
             length(p) AS hops

        WITH source_group_index,
             collect({{
                 p: p,
                 source: source,
                 target: target,
                 path_nodes_raw: path_nodes_raw,
```

```
                path_rels_raw: path_rels_raw,
                hops: hops
            }}) AS group_paths

        WITH source_group_index,
             [path IN group_paths | path][0..coalesce(
$paths_per_group, 10)] AS limited_paths

        UNWIND limited_paths AS path_data

        RETURN DISTINCT
            path_data.source.id AS source_id,
            path_data.target.id AS target_id,
            source_group_index,
            [i IN range(0, size(path_data.path_nodes_raw)-1) |
{{
                id: path_data.path_nodes_raw[i].id,
                name: path_data.path_nodes_raw[i].name,
                type: path_data.path_nodes_raw[i].type,
                description: path_data.path_nodes_raw[i].
description,
                position: i
            }}] AS path_nodes,
            [i IN range(0, size(path_data.path_rels_raw)-1) | {{
                type: type(path_data.path_rels_raw[i]),
                name: path_data.path_rels_raw[i].name,
                direction: CASE
                    WHEN startNode(path_data.path_rels_raw[i]).
id = path_data.path_nodes_raw[i].id
                    THEN 'OUTGOING'
                    ELSE 'INCOMING'
                END
            }}] AS path_rels,
            path_data.hops AS hops
        ORDER BY source_group_index, path_data.hops ASC
        """

        # Execute query
        result = await neo4j_conn.execute_query(
            query=query_cypher,
            parameters={
```

```
                "entity_groups": entity_groups,
                "paths_per_group": paths_per_group,
            }
        )

        records = result.record if hasattr(result, "record")
else result
        return [dict(record) for record in records]

    except Exception as e:
        logger.error(f"Entity Based All Shortest Path search
failed: {e}")
        return []
```