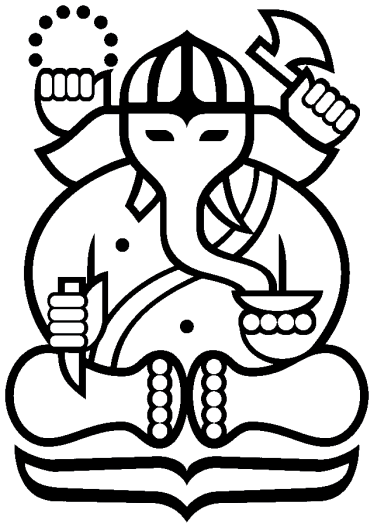
Laporan Tugas Proyek 2 IF4070
Graf Pengetahuan dan Sistem RAG untuk Klasifikasi Minuman Kopi



PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO & INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

Kelompok U:

13522011 - Dewantoro Triatmodjo

13522055 - Benardo

13522113 - William Glory Henderson

DAFTAR ISI

DAFTAR GAMBAR

BAB I
Graf Pengetahuan

1.1. Pemanfaatan Secara Garis Besar

Graf pengetahuan ini dirancang untuk memodelkan domain pengetahuan kopi secara eksplisit dan terstruktur. Pemanfaatan utamanya adalah sebagai sumber kebenaran (*source of truth*) bagi sistem *Retrieval-Augmented Generation* (RAG). Dengan merepresentasikan kopi dan atributnya sebagai simpul (*node*) yang saling terhubung, sistem dapat melakukan kueri kompleks yang melibatkan hubungan antar entitas (misalnya, mencari kopi apa saja yang berasal dari Italia) yang sulit dilakukan dengan pencarian teks biasa. Selain itu tujuannya juga untuk menghindari ambiguitas yang sering terjadi pada penggunaan teks tidak terstruktur.

Pemanfaatan utama graf ini adalah:

1. Representasi Relasional: Menghubungkan entitas kopi dengan atribut-atribut pembentuk / properti yang berkaitan secara langsung.
2. Sumber Kebenaran (*Ground Truth*): Menyediakan fakta yang dapat diverifikasi. Sistem tidak mengarang jawaban, melainkan mengambilnya dari *node* dan *relationship* yang ada.
3. Fleksibilitas Kueri: Memungkinkan pertanyaan multi-dimensi, seperti "Kopi apa yang berasal dari Italia tapi tidak menggunakan susu?" yang sulit dijawab oleh sistem pencarian biasa tanpa pemahaman konteks yang dalam.

1.2. Deskripsi Label dan Simpul yang Terlibat

Graf ini dibangun menggunakan skema yang ketat (*schema-strict*) dengan *constraints* untuk menjaga integritas data. Berikut adalah deskripsinya.

1. *Node Labels* (Entitas Utama)
    - *Coffee*: Representasi dari minuman kopi, *Properties*: *code* dan *name*
        - Espresso
        - Bica
        - Americano
        - Latte

    - Caffe Macchiato
    - Cappuccino
    - Flat White
    - Latte Macchiato
    - Kopi Tubruk
    - Greek Coffee
    - Cafe Mocha
- *Base*: Basis dari minuman kopi
    - *Espresso Base*
    - *Brewed Coffee Base*
- *MilkType*: Jenis atau tekstur susu yang digunakan
    - *No Milk*
    - *Steamed Milk*
    - *Foamed Milk*
    - *Steamed And Foamed Milk*
    - *Microfoam Milk*
    - *Cold Milk*
- *Additive*: Bahan tambahan perasa atau pelengkap
    - *None*
    - *Hot Water*
    - *Sugar*
    - *Chocolate*
- *PreparationMethod*: Teknik/metode pembuatan
    - *Pressure Method*
    - *Boiled Method*
    - *Diluted Method*
    - *Combined Method*
- *ServingStyle*: Wadah/cara penyajian
    - *Small Cup Serving*
    - *Tall Glass Serving*
    - *Large Cup Serving*

    - *Cup Serving*
    - *Demitasse Serving*
    - *Unfiltered Serving*
- Origin: Negara atau wilayah asal minuman
    - Italy *Origin*
    - Portugal *Origin*
    - Indonesia *Origin*
    - Greece *Origin*
    - Australia / New Zealand *Origin*

2. *Relationships* (Hubungan)
    - (:Coffee)-[:HAS_BASE]→(:Base): Menunjukkan bahan dasar utama
    - (:Coffee)-[:HAS_MILK]→(:MilkType): Menentukan profil susu
    - (:Coffee)-[:HAS_ADDITIVE]→(:Additive): Menunjukkan tambahan standar
    - (:Coffee)-[:USES_PREPARATION]→(:PreparationMethod): Menjelaskan teknik preparasi
    - (:Coffee)-[:SERVED_IN]→(:ServingStyle): Standar penyajian
    - (:Coffee)-[:ORIGINATES_FROM]→(:Origin): Asal geografis/budaya

1.3. Keterkaitan antar Ontologi

Struktur graf ini menerapkan skema ontologi yang menstandarisasi definisi kelas dan properti relasional, dengan entitas *Coffee* sebagai pusatnya. Pendekatan ini mengubah data statis menjadi graf pengetahuan yang memungkinkan penalaran logis (*reasoning*).

1. Prinsip Pemodelan:
    - Konsep sebagai Label: Kelas entitas direpresentasikan sebagai *Node* Label (contoh: :Coffee, :MilkType, :Origin).
    - Instansi sebagai *Node*: Item spesifik direpresentasikan sebagai *Node* individu (contoh: *Node* Espresso, *Node* Steamed Milk).

2. Definisi Intensional (*Intensional Definition*):

- Semantik dari entitas *Coffee* tidak disimpan sebagai teks deskripsi statis, melainkan didefinisikan oleh koleksi relasinya.
- Contoh: *Node* Latte didefinisikan sebagai entitas yang memiliki relasi HAS_BASE ke Espresso dan HAS_MILK ke Steamed Milk.

3. Kemampuan Inferensi (*Graph-Based Reasoning*):
    - Sistem dapat mengelompokkan data secara dinamis tanpa pelabelan manual, cukup dengan menelusuri struktur graf.
    - Contoh *Query*: "Sebutkan varian berbasis Espresso?" Sistem mencari semua node yang memiliki relasi HAS_BASE ke node Base {code: 'espresso'}.
    - Analisis Kesamaan: Dua jenis kopi dapat dianggap "bersaudara" secara ontologis jika mereka berbagi shared neighbors (misalnya, terhubung ke tipe susu yang sama).

BAB 2
Sistem RAG

2.1. Arsitektur Sistem

Sistem dibangun menggunakan bahasa Python dengan arsitektur modular yang terdiri dari beberapa komponen kunci:

1. RAG *Engine* (Controller Utama):
    - Bertanggungjawab sebagai pengendali utama yang mengorkestrasi seluruh aliran data dari input pengguna hingga output jawaban.
    - Mengelola urutan pemanggilan komponen (terima pertanyaan → *generate cypher* → validasi → eksekusi → format jawaban).
    - Bertanggung jawab atas inisialisasi dan penutupan koneksi database (*connect*() dan *disconnect*())
    - Menangani *exception* yang terjadi di *low level* (seperti koneksi terputus atau API timeout) agar tidak menyebabkan *crash* pada aplikasi, serta mengembalikan respon *fallback* yang sesuai jika diperlukan.

2. Neo4j *Client* (*Database Interface*):
    - Menangani komunikasi dengan Neo4j menggunakan neo4j-driver (graf *database*).
    - Memiliki metode validate_query() yang menggunakan fitur EXPLAIN dari Neo4j untuk mengecek validitas sintaks Cypher sebelum dieksekusi, mencegah *runtime error* yang fatal dan injeksi kueri yang salah yang bisa membahayakan integritas sistem.
    - Mengonversi hasil mentah dari driver Neo4j (yang berupa objek *Record*) menjadi struktur data Python standar (*List of Dict*) agar mudah diproses oleh komponen lain.

3. Gemini *Client* dan *OpenRouter Client*(LLM *Interface*):
    - Abstraksi untuk model bahasa (mendukung *Google Gemini* dan *OpenRouter*).

- generate_cypher(): Menggunakan *prompt engineering* khusus (*system prompt*) yang berisi skema graf untuk memandu LLM menerjemahkan bahasa alami ke Cypher.
- format_results(): Mengubah hasil JSON mentah dari Neo4j menjadi narasi yang ramah pengguna.

2.2. Fungsionalitas Penting dari Sistem RAG yang Dikembangkan

Sistem RAG yang dikembangkan memiliki beberapa fungsionalitas penting sebagai berikut:

1. *Mekanisme Self-Correction & Retry Loop*: Sistem tidak langsung menyerah saat LLM menghasilkan Cypher query yang salah sintaksnya. RAG *Engine* memiliki *loop* percobaan otomatis (hingga 3x). Jika validasi Neo4jClient menemukan *error*, pesan *error* tersebut dikirimkan kembali ke LLM agar LLM dapat memperbaiki kuerinya sendiri secara iteratif sebelum akhirnya dieksekusi atau dinyatakan gagal.
2. *Text-to-Cypher Translation*: Mengubah pertanyaan bahasa alami (contoh: "Apa kopi buatan Itali?") menjadi kueri *database* graf (*Cypher*) yang valid. *System prompt* dirancang dengan aturan ketat ("*Generate ONLY valid Cypher*", "*Use MATCH patterns*"). Ini meminimalkan kueri yang tidak valid.
3. Validasi Kueri (Sebelum eksekusi): Sebelum kueri dikirim ke *database*, kueri divalidasi menggunakan fitur EXPLAIN dari Neo4j. Jika sintaks salah (hal yang umum terjadi pada LLM), sistem menangkapnya lebih awal. Hal ini memastikan sintaks Cypher yang dihasilkan valid sebelum dieksekusi untuk mencegah error database.
4. Eksekusi *Graph Query* dan *Result Formatting*: Menjalankan kueri terhadap Neo4j untuk mendapatkan fakta terstruktur dan mengubah hasil data tersebut (JSON/Dictionary) kembali menjadi jawaban bahasa alami yang mudah dipahami pengguna.
5. *Fallback Mechanism*: Sebuah fitur ketahanan (*robustness*) di mana jika *database* tidak dapat diakses atau gagal menghasilkan kueri yang valid (hasil kuerinya kosong), sistem secara otomatis tidak membiarkan LLM mengarang jawaban. Sistem secara eksplisit mengembalikan pesan "*I can only help with coffee questions. Try asking about coffee types, ingredients, or origins.*", memastikan pengguna tahu batasan pengetahuan sistem. Sistem dirancang untuk menolak pertanyaan di luar domain kopi. Jika pengguna bertanya tentang hal lain (misal: “Siapa nama presiden Indonesia?”), LLM dilatih untuk merespon

dengan token khusus OUT_OF_SCOPE. Sistem kemudian menangkap token ini dan memberikan respons standar yang sopan tanpa membuang sumber daya *database*.

6. *Error Suppression*: Pengguna akhir tidak pernah melihat pesan *error* teknis (contoh: "*Cypher Syntax Error*"). Semua *error* ditangani di *backend*, dan pengguna mendapat pesan yang sopan mengenai ketiadaan data.

2.3. Pemanfaatan Graf Pengetahuan

Dalam sistem ini, graf pengetahuan bertindak bukan hanya sebagai *database* tetapi juga sebagai konteks eksternal yang presisi. Berbeda dengan RAG berbasis vektor yang mencari potongan teks mirip, sistem ini bertanya langsung ke database untuk mendapatkan fakta yang *exact match*. Alur pemanfaatannya adalah:

1. Permintaan Pengguna (*User Query*): Pengguna mengajukan pertanyaan, misalnya: "Apa perbedaan antara Latte dan Cappuccino?"

2. Penalaran & Validasi LLM (*Reasoning & Validation*): LLM menerjemahkan intensi pengguna menjadi logika kueri graf (Cypher). Sebelum dieksekusi, sistem memvalidasi kueri tersebut melalui fungsi validate_query(). Jika terdeteksi sintaks yang tidak valid, sistem mendorong LLM untuk melakukan perbaikan mandiri (*self-correction*).

3. Pengambilan Data Graf (*Graph Retrieval*): Setelah kueri tervalidasi, sistem mengeksekusinya untuk mengambil data faktual. Contoh Kueri: MATCH (c:Coffee) WHERE c.code IN ['latte', 'cappuccino'] RETURN c.name, c.milk

4. Injeksi Konteks Bersyarat (*Conditional Context Injection*):
    - Jika Data Ditemukan: Fakta mentah (misal: Latte = Steamed Milk, Cappuccino = Steamed + Foamed) diinjeksikan ke dalam prompt LLM untuk dinarasikan menjadi jawaban natural.
    - Jika Data Kosong: Alur ke LLM diputus (*short-circuited*). Sistem secara otoritatif menyatakan bahwa informasi tidak tersedia. Mekanisme ini mencegah LLM melakukan halusinasi (mengarang jawaban fiktif).

5. Jawaban Akhir (*Final Answer*): Respon akhir kepada pengguna dijamin bersumber langsung dari simpul graf atau berupa pernyataan eksplisit bahwa data tidak ditemukan.

BAB 3

Pembahasan

3.1. Hasil dari Sistem RAG

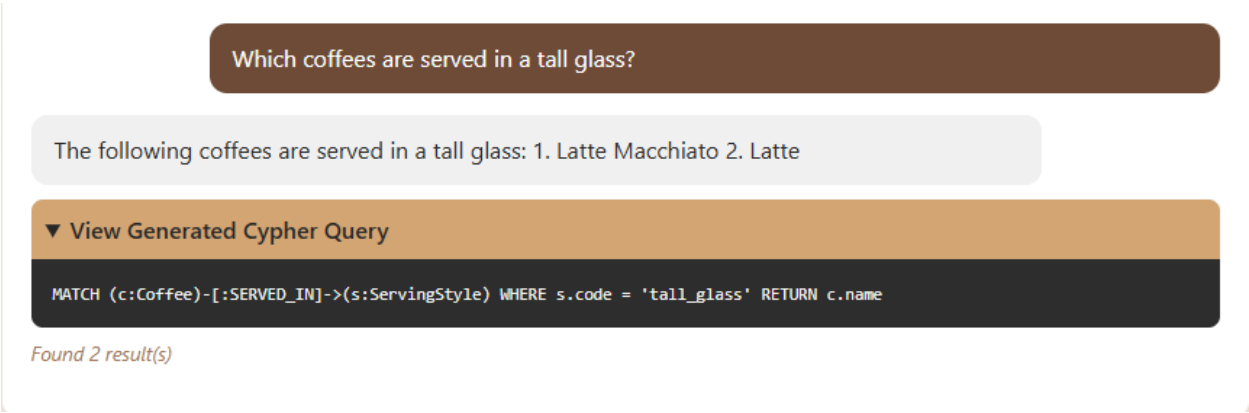
Testing 1

Status: Berhasil

Gambar 1 - Hasil dari Sistem RAG Testing 1

Testing 2

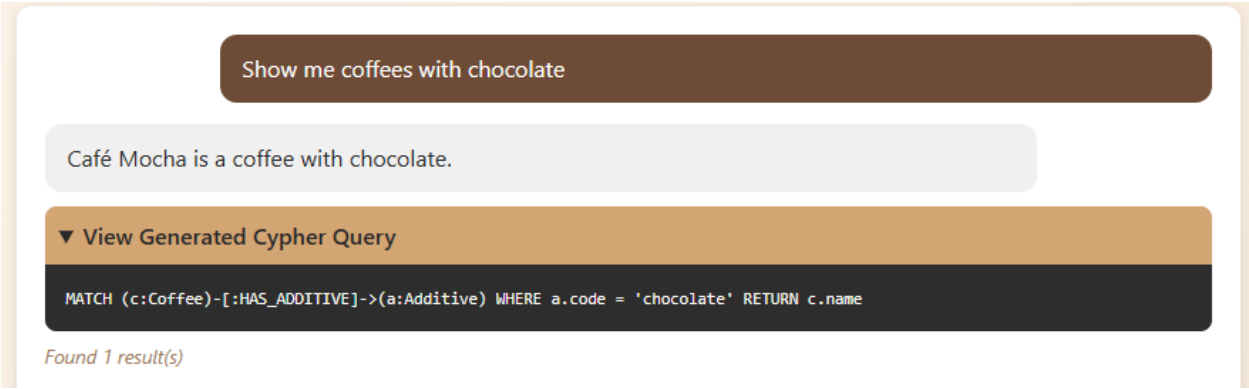
Status: Berhasil



Gambar 2 - Hasil dari Sistem RAG Testing 2

Testing 3

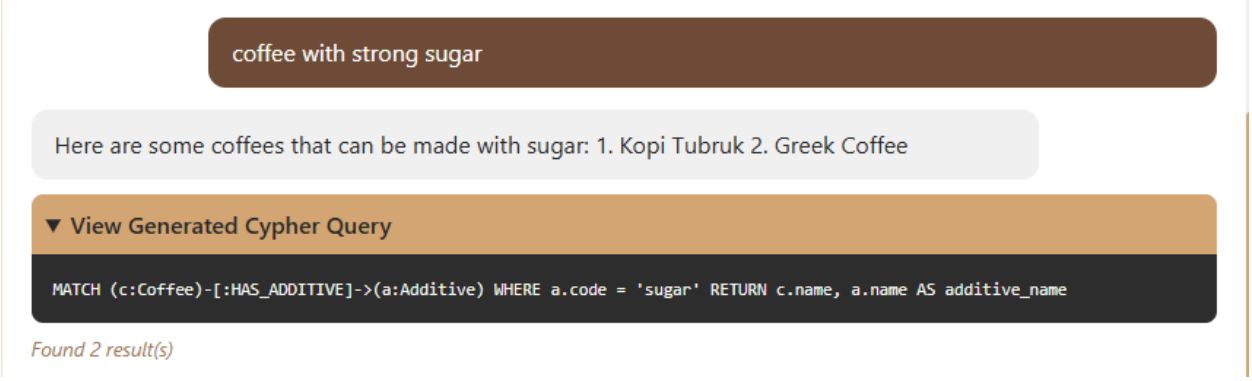
Status: Berhasil



Gambar 3 - Hasil dari Sistem RAG Testing 3

Testing 4

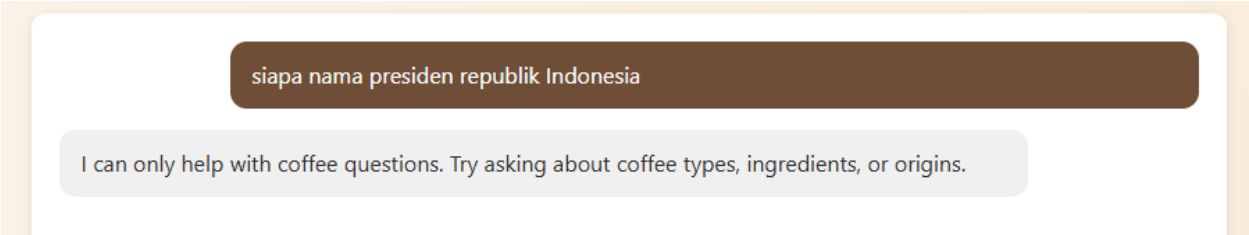
Status: Berhasil



Gambar 4 - Hasil dari Sistem RAG Testing 4

Testing 5

Status: Berhasil (Diluar Context)



Gambar 5 - Hasil dari Sistem RAG Testing 5

3.2. Perbandingan antara Sistem RAG dan Sistem Berbasis Pengetahuan

Dalam pengerjaan tugas proyek ini, kami mengeksplorasi dua paradigma berbeda dalam pengelolaan pengetahuan domain kopi. Pendekatan pertama pada tugas 1 adalah sistem berbasis pengetahuan tradisional yang memanfaatkan Prolog dan Ontologi, sedangkan pendekatan kedua pada tugas 2 adalah sistem modern *Retrieval-Augmented Generation* (RAG) yang mengintegrasikan *knowledge graph* Neo4j dengan *Large Language Model* (LLM). Perbedaan mendasar antara kedua sistem ini terletak pada cara representasi pengetahuan, metode interaksi, dan mekanisme penalaran yang digunakan.

Sistem berbasis pengetahuan tradisional bersifat deterministik dan sangat formal. Pengetahuan dikodekan secara eksplisit dalam bentuk aturan logika dan hierarki kelas yang ketat. Keunggulan utama dari pendekatan ini adalah presisi yang mutlak dan kemampuan untuk melakukan penalaran logis (*reasoning*) secara otomatis untuk memvalidasi konsistensi data. Namun, kekakuan ini menjadi kelemahan tersendiri karena pengguna diwajibkan memahami sintaks kueri formal untuk berinteraksi dengan sistem. Hal ini membuat sistem kurang fleksibel dan memiliki kurva pembelajaran yang tinggi, sehingga lebih cocok digunakan untuk keperluan verifikasi akademis atau oleh pengguna ahli.

Sebaliknya, sistem RAG menawarkan pendekatan yang jauh lebih fleksibel dan adaptif. Dengan menggabungkan struktur data graf dan kemampuan pemrosesan bahasa alami dari LLM, sistem ini memungkinkan pengguna untuk bertanya menggunakan bahasa sehari-hari tanpa perlu memahami struktur *database*. LLM bertugas menerjemahkan pertanyaan pengguna menjadi kueri Cypher yang valid. Meskipun sistem ini tidak memiliki mekanisme validasi logika sekuat Prolog dan bersifat probabilistik (hasil bergantung pada kualitas interpretasi LLM), sistem RAG jauh lebih unggul dalam hal pengalaman pengguna (*user experience*) dan skalabilitas.

Secara keseluruhan, kedua pendekatan memiliki kegunaannya masing-masing. Sistem berbasis Prolog unggul dalam akurasi dan validasi logika yang ketat, sementara sistem RAG unggul dalam kemudahan penggunaan dan interaksi yang intuitif. Dalam konteks pengembangan aplikasi yang ditujukan untuk pengguna umum, pendekatan RAG terbukti lebih relevan karena

mampu menghubungkan kompleksitas data menjadi informasi yang mudah diakses melalui percakapan natural.

3.3. Keterbatasan Graf Pengetahuan dan RAG

Meskipun implementasi sistem RAG dengan *knowledge graph* Neo4j memperlihatkan berbagai keunggulan interaktif, terdapat sejumlah keterbatasan signifikan yang perlu diperhatikan. Keterbatasan ini mencakup aspek struktur graf pengetahuan itu sendiri, pendekatan RAG secara umum, serta spesifikasi teknis dari implementasi yang telah dibangun.

Pertama, dari sisi Graf Pengetahuan (Neo4j), keterbatasan utama terletak pada skala dan kompleksitas data yang masih minim. Domain pengetahuan saat ini hanya mencakup entitas kopi dalam jumlah terbatas dengan properti yang sederhana, tanpa menyertakan detail mendalam seperti konteks sejarah, varietas biji, atau metode penyeduhan yang kompleks. Selain itu, karena Neo4j berfungsi sebagai *database* dan bukan *reasoner* logika murni, sistem tidak memiliki kemampuan inferensi otomatis atau validasi konsistensi semantik layaknya sistem berbasis ontologi. Hal ini membuat sistem sangat bergantung pada kueri eksplisit dan struktur data yang kaku, serta menghadapi tantangan skalabilitas jika skema perlu diubah di masa depan.

Kedua, keterbatasan juga ditemukan pada pendekatan RAG yang sangat bergantung pada *Large Language Model* (LLM). Sifat LLM yang *non-deterministik* dan probabilistik menimbulkan risiko inkonsistensi jawaban serta potensi halusinasi informasi, di mana model dapat mengarang fakta yang tidak ada dalam *database*. Sistem juga belum mendukung pencarian semantik berbasis vektor, sehingga kurang toleran terhadap kesalahan penulisan atau istilah yang ambigu. Selain itu, terdapat isu latensi dan ketergantungan pada kualitas model eksternal, di mana kegagalan dalam menerjemahkan bahasa alami ke sintaks database akan menyebabkan kegagalan sistem secara keseluruhan.

Ketiga, secara spesifik pada implementasi sistem kami, terdapat kekakuan pada skema data yang sulit untuk dikembangkan ke domain baru tanpa refaktorisasi ulang. Sistem saat ini hanya mendukung bahasa Inggris dan bersifat satu arah (*read-only*), sehingga tidak memungkinkan pengguna untuk berkontribusi menambahkan data baru. Aspek teknis seperti validasi data yang

mendalam, penanganan kesalahan yang komprehensif, keamanan API, serta pemantauan performa (*monitoring*) juga belum sepenuhnya terintegrasi, menjadikan status sistem ini masih berupa purwarupa pengembangan dan belum siap untuk lingkungan produksi.

Untuk mengatasi berbagai keterbatasan tersebut, pengembangan ke depan disarankan berfokus pada perluasan domain pengetahuan agar lebih kaya dan mendetail, serta penerapan pendekatan pencarian *hybrid* yang menggabungkan kueri graf dengan pencarian vektor semantik. Integrasi fitur-fitur lanjutan seperti dukungan multi-bahasa, mekanisme *caching* untuk mempercepat respons, serta penerapan standar teknis industri (seperti *CI/CD* dan kontainerisasi) juga sangat diperlukan. Langkah-langkah ini diharapkan dapat meningkatkan *robustness*, akurasi, dan skalabilitas sistem agar dapat diandalkan.

BAB 4
Kesimpulan

Berdasarkan perancangan, implementasi, dan pengujian yang telah dilakukan pada sistem *Retrieval-Augmented Generation* (RAG) berbasis Graf Pengetahuan untuk domain minuman kopi, dapat ditarik beberapa kesimpulan yaitu:

1. Efektivitas Graf Pengetahuan sebagai *Source of Truth.* Penerapan struktur ontologi yang ketat (*schema-strict*) pada Neo4j berhasil memodelkan domain kopi secara eksplisit. Definisi intensional yang diterapkan, di mana entitas seperti "Latte" didefinisikan berdasarkan relasinya terhadap basis espresso dan susu, memungkinkan sistem melakukan penalaran berbasis graf (*graph-based reasoning*). Hal ini membuktikan bahwa graf pengetahuan mampu menangani kueri multi-dimensi yang kompleks (seperti hubungan bahan, asal, dan metode) yang sulit dijawab oleh sistem pencarian teks konvensional.

2. Keandalan Arsitektur RAG dan Mekanisme Validasi. Sistem RAG yang dikembangkan berhasil menghubungkan kesenjangan antara bahasa alami manusia dan bahasa kueri *database* (Cypher). Fitur-fitur ketahanan sistem, khususnya mekanisme *Self-Correction* & *Retry Loop* serta validasi sintaks menggunakan EXPLAIN, terbukti efektif dalam meminimalkan kegagalan eksekusi akibat kesalahan interpretasi LLM. Selain itu, mekanisme *Conditional Context Injection* dan *Short-circuiting* berhasil mencegah halusinasi AI, memastikan bahwa setiap jawaban yang diberikan kepada pengguna memiliki dasar fakta yang dapat diverifikasi atau secara jujur menyatakan ketidaktahuan jika data tidak tersedia.

3. Perbandingan antara Sistem Berbasis Pengetahuan dan Sistem RAG. Eksplorasi terhadap dua pendekatan pengelolaan pengetahuan menunjukkan perbedaan fundamental. Sistem berbasis logika tradisional (Prolog) unggul dalam validasi formal dan kepastian deterministik, namun memiliki hambatan interaksi yang tinggi bagi pengguna umum. Sebaliknya, sistem RAG menawarkan fleksibilitas interaksi yang jauh lebih baik melalui antarmuka bahasa alami, meskipun bersifat probabilistik. Untuk aplikasi yang berorientasi pada pengalaman pengguna

akhir (*end-user*), pendekatan RAG terbukti lebih relevan dibandingkan pendekatan berbasis aturan kaku (sistem berbasis pengetahuan).

4. Keterbatasan dan Arah Pengembangan. Meskipun sistem ini berfungsi dengan baik, sistem masih memiliki keterbatasan pada skala data yang kecil dan ketergantungan penuh pada kemampuan penerjemahan LLM tanpa dukungan pencarian vektor (semantik). Pengembangan selanjutnya perlu difokuskan pada penerapan metode *hybrid search* yang menggabungkan presisi graf dengan fleksibilitas vektor, serta perluasan skema ontologi untuk mencakup domain pengetahuan yang lebih luas dan mendalam.

Secara keseluruhan, integrasi antara *Large Language Model* (LLM) dan *Knowledge Graph* dapat menciptakan sistem tanya-jawab yang tidak hanya cerdas dan interaktif, tetapi juga dapat dipercaya dan mudah digunakan oleh pengguna umum.

BAB VI

Lampiran: Matriks Kontribusi

Nama - NIM	Pembagian Tugas
Dewantoro Triatmodjo - 13522011	Mengerjakan semua tugas secara bersama
Benardo - 13522055	Mengerjakan semua tugas secara bersama
William Glory Henderson - 13522113	Mengerjakan semua tugas secara bersama

Link Repository: https://github.com/wegeh/Tubes-IF4070-RPP/releases/tag/v1.0