CHATBOT INFORMASI AKADEMIK TEKNIK INFORMATIKA UIN MAULANA MALIK IBRAHIM MALANG BERBASIS *RETRIEVAL AUGMENTED GENERATION* (RAG)

SKRIPSI

Oleh :
WAFIY ANWARUL HIKAM
NIM. 220605110022

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2025

CHATBOT INFORMASI AKADEMIK TEKNIK INFORMATIKA UIN MAULANA MALIK IBRAHIM MALANG BERBASIS *RETRIEVAL AUGMENTED GENERATION* (RAG)

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
WAFIY ANWARUL HIKAM
NIM. 220605110022

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2025

HALAMAN PERSETUJUAN

CHATBOT INFORMASI AKADEMIK TEKNIK INFORMATIKA UIN MAULANA MALIK IBRAHIM MALANG BERBASIS *RETRIEVAL AUGMENTED GENERATION* (RAG)

SKRIPSI

Oleh :
WAFIY ANWARUL HIKAM
NIM. 220605110022

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 09 Desember 2025

Pembimbing I,	Pembimbing II,
Dr. Totok Chamidy, M.Kom	Dr. Muhammad Ainul Yaqin, M.Kom
NIP. 19691222 200604 1 001	NIP. 19761013 200604 1 004

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Supriyono, M.Kom
NIP. 19841010 201903 1012

HALAMAN PENGESAHAN

CHATBOT INFORMASI AKADEMIK TEKNIK INFORMATIKA UIN MAULANA MALIK IBRAHIMMALANG BERBASIS *RETRIEVAL AUGMENTED GENERATION* (RAG)

SKRIPSI

Oleh :
WAFIY ANWARUL HIKAM
NIM. 220605110022

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 16 Desember 2025

Susunan Dewan Penguji

Ketua Penguji	: Supriyono, M.Kom NIP. 19841010 201903 1012	( )
Anggota Penguji I	: Fajar Rohman Hariri, M.Kom NIP. 19890515 201801 1 001	( )
Anggota Penguji II	: Dr. Totok Chamidy, M.Kom NIP. 19691222 200604 1 001	( )
Anggota Penguji III	: Dr. Muhammad Ainul Yaqin, M.Kom NIP. 19761013 200604 1 004	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Supriyono, M.Kom
NIP. 19841010 201903 1012

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: Wafiy Anwarul Hikam
NIM	: 220605110022
Fakultas / Program Studi	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: Chatbot Informasi Akademik Teknik Informatika UIN Maulana Malik Ibrahim Malang Berbasis Rertrieval Augmented Generation (RAG)

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 25 Desember 2025
Yang membuat pernyataan,

Wafiy Anwarul Hikam
NIM.220605110022

MOTTO

"Jadilah Mata Air Bersih Yang Mengalir"

"Janganlah Menjadi Air Kotor Yang Menggenang"

HALAMAN PERSEMBAHAN

Dengan penuh rasa syukur, karya ini penulis persembahkan kepada:

Ayah dan Ibu tercinta dan adik penulis
Yang senantiasa mendukung, mendoakan, dan memberikan kasih sayang tanpa batas serta selalu menjadi motivasi terbesar penulis.

Keluarga besar
Atas dukungan dan doa yang selalu mengiringi langkah penulis.

Dosen Pembimbing dan Seluruh Pengajar
Yang telah membimbing serta memberikan ilmu yang bermanfaat.

Teman-teman Infinity Teknik Informatika 2022
Yang selalu menemani perjalanan belajar di bangku perkuliahan ini dengan penuh cerita, kebersamaan, semangat, dan optimisme.

Dan terakhir, untuk diri penulis sendiri
Yang telah berusaha dan bertahan sampai sejauh ini, semoga bahu ini semakin dikuatkan untuk terus melangkah maju tanpa ragu.

KATA PENGANTAR

Bismillahirrahmaanirrahiim, Assalamu'alaikum Warahmatullahi Wabarakatuh. Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT yang telah melimpahkan rahmat, nikmat, serta hidayah-Nya, sehingga pada kesempatan kali ini penulis mampu menyelesaikan skripsi yang berjudul "Chatbot Informasi Akademik Teknik Informatika UIN Maulana Malik Ibrahim Malang Berbasis *Retrieval Augmented Generation"* dengan baik dan lancar. Shalawat serta salam senantiasa penulis lafalkan kepada junjungan kekasih Allah yakni Nabi Agung Muhammad SAW., yang telah menjadi teladan mulia bagi seluruh umat manusia, membawa kita semua dari zaman jahiliyyah menuju zaman kebenaran, sempurnanya agama Islam, dan zaman yang penuh ilmu pengetahuan sehingga dapat kita nikmati manfaatnya hingga saat ini.

Skripsi ini penulis susun dalam rangka memenuhi salah satu syarat kelulusan untuk memperoleh gelar Sarjana Komputer pada Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, UIN Maulana Malik Ibrahim Malang. Selama proses penyusunan skripsi ini, penulis banyak sekali mendapatkan dukungan, bantuan, serta doa dari berbagai pihak. Maka dari itu, pada halaman ini penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Prof. Dr. Hj. Ilfi Nur Diana, M.Si., CAHRM., CRMP,, selaku Rektor Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang.
2. Prof. Dr. Sri Harini, M.Si., selaku Dekan Fakultas Sains dan Teknologi UIN Maulana Malik Ibrahim Malang.

3. Supriyono, M.Kom, selaku Ketua Program Studi Teknik Informatika UIN Maulana Malik Ibrahim Malang, sekaligus ketua penguji yang turut memberikan arahan dan kritik serta sarannya terhadap skripsi ini.
4. Dr. Totok Chamidy, M.Kom., selaku Dosen Pembimbing I, yang selalu dengan sabar membimbing dan mengarahkan penulis selama penyusunan skripsi ini.
5. Dr. Muhammad Ainul Yaqin, M.Kom., selaku Dosen Pembimbing II, yang juga senantiasa memberikan arahan dan masukan dalam pengerjaan skripsi ini.
6. Fajar Rohman Hariri, M.Kom., selaku Anggota Penguji 1 yang telah memberikan kritik, saran konstruktif, dan bimbingan untuk menyempurnakan skripsi ini.
7. Nia Faricha, S.Si., selaku admin Program Studi Teknik Informatika, yang turut membantu penulis dalam urusan administrasi dan selalu mengingatkan tentang kelengkapan berkas.
8. Bapak Wido Nugroho dan Ibu Faridatul Khuriyah, orang tua tercinta, serta Taufiq Luthfi Nurrohim, adik penulis, yang terus menjadi sumber kekuatan dengan doa, cinta, dan dukungan tanpa henti.
9. Teman-teman Boejank Abangan, Ridho, Faqih, Fahrizal yang telah menjadi tempat diskusi bersama berbagi cerita suka dan duka serta salah satu support system penulis.
10. Teman-teman Kopstud dan Sobat Lab Intelligent System, Uqie, Arif, Ridi, dan Radifan yang turut menjadi support system penulis juga selama

menghadapi berbagai kondisi mengerjakan skripsi serta tempat berbagi cerita dan motivasi.

11. Teman-teman IFL Malang, khususnya teman-teman Departemen CDSI Bintang, Keysha dan lainnya beserta teman-teman Divisi Sistem Informasi Zul, Tika, Akin, Anggi yang juga turut berperan menjadi penyemangat dan tempat penulis tumbuh dan belajar serta saling mendukung satu sama lain.
12. Teman-teman Sekamar Mabna Ibn-Rusyd Nomor 2 yang telah memberikan kebersamaan dan cerita suka duka di awal masa perkuliahan.
13. Teman-teman KKM Desa Sukomulyo yang turut memberikan pengalaman nyata dalam pengabdian kepada masyarakat dan ruang bagi penulis bisa mengabdikan diri.
14. Seluruh keluarga besar, teman, sahabat, dan kerabat penulis yang tidak dapat penulis sebutkan satu per satu, yang turut memberikan bantuan, semangat, dukungan, dan doa untuk penulis.
15. Terakhir tidak lupa terimakasih kepada diri penulis, yang sudah mau dan mampu melangkah hingga sejauh ini dengan tekad yang kuat sehingga setiap tantangan dan kesulitan dapat dilalui dengan baik pada perjalanan ini.

Penulis menyadari bahwa penelitian dalam tugas skripsi ini masih jauh dari kata sempurna dan terdapat banyak keterbatasan. Maka dari itu, dengan penuh kerendahan hati, penulis membuka diri untuk menerima kritik dan saran yang membangun dari para pembaca guna menjadi bahan evaluasi dan pengembangan di masa mendatang. Penelitian ini juga memiliki potensi untuk dilanjutkan dan dikembangkan lebih jauh dalam penelitian berikutnya, sehingga dapat melengkapi

kekurangan yang ada. Harapan penulis, karya ini di masa mendatang dapat memberikan kontribusi positif bagi masyarakat secara luas dan tidak berhenti sebatas memberikan manfaat bagi pembaca saja.

Wassalamu'alaikum Warahmatullahi Wabarakatuh. Assalamu'alaikum Warahmatullahi Wabarakatuh.

Malang, 22 Desember 2025

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Anwarul Hikam, Wafiy. 2025. Chatbot Informasi Akademik Teknik Informatika UIN Maulana Malik Ibrahim Malang Berbasis *Retrieval Augmented Generation* (RAG). Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Dr. Totok Chamidy, M.Kom (II) Dr. Muhammad Ainul Yaqin, M.Kom.

.

Kata kunci: chatbot, Retrieval-Augmented Generation, RAG, informasi akademik, TinyLlama, FAISS, BM25.

Aksesibilitas informasi akademik merupakan tantangan di institusi pendidikan tinggi karena mahasiswa kesulitan menemukan informasi terkini mengenai kurikulum, dosen, dan kebijakan akademik yang tersebar di berbagai platform. Penelitian ini bertujuan merancang dan mengimplementasikan sistem chatbot berbasis *Retrieval-Augmented Generation* (RAG) untuk layanan informasi akademik Prodi Teknik Informatika UIN Malang, serta menganalisis kinerjanya dalam memberikan respons akurat dan kontekstual. Metode penelitian menggunakan pendekatan *Research* and *Development* dengan tahapan: pengumpulan data melalui ekstraksi otomatis 350 halaman menghasilkan 384 potongan dokumen, pembangunan sistem pencarian gabungan FAISS dan BM25, implementasi pembangkit jawaban menggunakan TinyLlama 1.1B, serta evaluasi menggunakan 60 pertanyaan dengan ukuran kinerja pencarian dan pembangkitan jawaban. Hasil menunjukkan pencarian mencatat nilai kualitas peringkat 0.258 dan tingkat penemuan 1.028 dengan kinerja terbaik pada pertanyaan acak (0.68), sedangkan pembangkit jawaban mencatat kemiripan semantik 0.381 dan kesetiaan dokumen 0.549 dengan kinerja optimal pada pertanyaan 5+ kata (0.472). Pengujian konsistensi dengan mengulang pertanyaan sama 5 kali menghasilkan variasi nol, memvalidasi mekanisme pengaman berhasil mengatasi ketidakkonsistenan. Keterbatasan meliputi pengulangan kalimat dan kebocoran template yang dimitigasi melalui penyaringan pasca-pemrosesan. Sistem beroperasi mandiri dengan sumber daya minimal, membuktikan kelayakan penerapan di institusi pendidikan dengan anggaran terbatas.

ABSTRACT

Anwarul Hikam, Wafiy. 2025. Chatbot Informasi Akademik Teknik Informatika UIN Maulana Malik Ibrahim Malang Berbasis Retrieval Augmented Generation (RAG). Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Promotor: (I) Dr. Totok Chamidy, M.Kom (II) Dr. Muhammad Ainul Yaqin, M.Kom.

Academic information accessibility poses a challenge in higher education institutions as students struggle to find up-to-date information regarding curriculum, faculty, and academic policies scattered across various platforms. This research aims to design and implement a Retrieval-Augmented Generation (RAG) based chatbot system for academic information services at the Informatics Engineering Study Program UIN Malang, and to analyze its performance in delivering accurate and contextual responses. The research employs a Research and Development approach with stages: data collection through automatic extraction of 350 pages yielding 384 document chunks, construction of hybrid retrieval system using FAISS and BM25, generation implementation using TinyLlama 1.1B, and evaluation using 60 queries with retrieval and generation performance metrics. Results show retrieval achieved ranking quality of 0.258 and discovery rate of 1.028 with optimal performance on scrambled queries (0.68), while generation recorded semantic similarity of 0.381 and document faithfulness of 0.549 with optimal performance on 5+ word queries (0.472). Consistency testing with 5 repetitions of the same query yielded zero variation, validating that guardrail mechanisms successfully addressed inconsistencies. Limitations include sentence repetitions and template leakage mitigated through post-processing filtering. The system operates self-hosted with minimal resources, proving deployment feasibility in educational institutions with limited budgets.

Key words: chatbot, Retrieval-Augmented Generation, academic information, TinyLlama, FAISS, BM25.

البحث مستخلص

أنوار الحكم، وافي. ٢٠٢٥ .برنامج المحادثة الآلي للمعلومات الأكاديمية لقسم هندسة المعلوماتية بجامعة مولانا مالك إبراهيم الإسلامية الحكومية بمالانج القائم على التوليد المعزز بالاسترجاع .(RAG) البحث الجامعي .قسم هندسة المعلوماتية، كلية العلوم والتكنولوجيا، جامعة مولانا مالك إبراهيم الإسلامية الحكومية بمالانج. المشرف: (الأول) الدكتور توتوك تشاميدي الماجستير (الثاني) الدكتور محمد عين اليقين الماجستير.

الكلمات المفتاحية: برنامج المحادثة الآلي (Chatbot) ، التوليد المعزز بالاسترجاع (RAG) ، المعلومات الأكاديمية، TinyLlama ،FAISS ،BM25.

تمثل إمكانية الوصول إلى المعلومات الأكاديمية تحديًا في مؤسسات التعليم العالي، حيث يواجه الطلاب صعوبات في العثور على المعلومات الحديثة المتعلقة بالمناهج الدراسية وأعضاء هيئة التدريس والسياسات الأكاديمية المتناثرة عبر منصات مختلفة. يهدف (RAG) القائم على تقنية التوليد المعزز بالاسترجاع (Chatbot) "هذا البحث إلى تصميم وتنفيذ نظام "روبوت الدردشة لخدمات المعلومات الأكاديمية في قسم هندسة المعلوماتية بجامعة مولانا مالك إبراهيم بمالانج، بالإضافة إلى تحليل أدائه في تقديم عبر مراحل متعددة: جمع البيانات من (R&D) استجابات دقيقة وسياقية. تستخدم هذه الدراسة منهج البحث والتطوير و FAISS خلال الاستخراج التلقائي لـ 350 صفحة مما أنتج 384 جزءاً من الوثائق، وبناء نظام استرجاع هجين يجمع بين ، وأخيراً التقييم باستخدام 60 استعلاماً مع مقاييس أداء TinyLlama 1.1B ، وتنفيذ التوليد باستخدام نموذجBM25 بلغت 0.258 (Ranking Quality) الاسترجاع والتوليد. أظهرت النتائج أن عملية الاسترجاع سجلت جودة ترتيب بلغ 1.028 مع أداء أفضل في الاستعلامات العشوائية (0.68). بينما سجل (Discovery Rate) ومعدل اكتشاف (Document Faithfulness) بنسبة 0.381 وموثوقية الوثائق (Semantic Similarity) التوليد تشابهاً دلالياً بنسبة 0.549، مع أداء أمثل في الاستعلامات التي تزيد عن 5 كلمات (0.472). أثبت اختبار الاتساق بتكرار نفس السؤال 5 مرات عدم وجود أي تباين، مما يؤكد نجاح آليات الحماية في معالجة عدم الاتساق. وتشمل القيود تكرار الجمل وتسرب بموارد (Self-hosted) القوالب التي تم التخفيف من حدتها من خلال التصفية بعد المعالجة. يعمل النظام بشكل مستقل .قليلة، مما يثبت جدوى تطبيقه في المؤسسات التعليمية ذات الميزانيات المحدودة

BAB I

PENDAHULUAN

1.1 Latar Belakang

Era transformasi digital berlangsung sangat cepat dan telah mengubah paradigma akses informasi di lingkungan perguruan tinggi tanpa terkecuali mulai dari lingkup paling luas di universitas, fakultas hingga program studi. Program Studi Teknik Informatika UIN Maulana Malik Ibrahim Malang sebagai salah satu program studi unggul telah menyediakan website resmi (https://informatika.uin-malang.ac.id/) sebagai portal informasi akademik yang komprehensif. Website tersebut memuat berbagai informasi penting meliputi profil program studi, kurikulum, informasi dosen, kelompok keilmuan, prosedur akademik, hingga panduan skripsi, dan PKL. Meski demikian, berdasarkan observasi awal, masih terdapat gap antara ketersediaan informasi dengan tingkat pemanfaatannya oleh para pengguna khususnya mahasiswa.

Permasalahan yang sering muncul adalah mahasiswa cenderung mengalami kesulitan dalam menemukan informasi spesifik yang mereka butuhkan di website program studi. Hal ini disebabkan oleh beberapa faktor, pertama struktur navigasi website yang memerlukan eksplorasi manual, kedua informasi yang tersebar di berbagai halaman berbeda, terakhir belum adanya mekanisme interaktif untuk membantu mahasiswa menemukan informasi dengan cepat. Sehingga, mahasiswa lebih memilih untuk bertanya langsung baik kepada teman seangkatannya yang terkadang harus menjelaskan ulang maupun kepada admin program studi, yang tentunya memiliki keterbatasan waktu dan sumber daya dalam melayani pertanyaan

repetitif dari para mahasiswa. Kondisi ini menuntut sistem bantu yang interaktif, adaptif, dan kontekstual.

Di sisi global, Labadze et al. (2024) melakukan kajian sistematis terhadap 67 penelitian internasional yang menunjukkan bahwa sistem interaktif chatbot AI membantu dalam akses cepat terhadap informasi akademik, personalisasi belajar, dan efisiensi interaksi dengan dosen. Di lain sisi, berdasarkan survei terhadap 5.894 mahasiswa universitas di Swedia, ditemukan bahwa lebih dari 55% memiliki sikap positif terhadap penggunaan chatbot AI, dan sekitar 35% sudah menggunakan ChatGPT secara rutin untuk kegiatan akademik (Stohr et al., 2024). Dan tren global saat ini menunjukkan adanya peningkatan signifikan penggunaan chatbot generatif di perguruan tinggi sejak 2023, menandakan pergeseran menuju sistem informasi akademik yang lebih interaktif (Mcgrath et al., 2025). Fakta-fakta tersebut memperkuat relevansi penelitian ini yang berfokus pada pengembangan chatbot akademik di era transformasi digital pendidikan tinggi.

Dalam perspektif Islam, akses terhadap ilmu yang mudah dan tertata adalah nilai yang dijunjung. Allah memerintahkan kepada kita di dalam Al-Qur'an untuk "membaca" sebagai jalan pembuka ilmu:

اِقْرَأْ بِاسْمِ رَبِّكَ الَّذِيْ خَلَقَۚ ١ اِقْرَأْ بِاسْمِ رَبِّكَ الَّذِيْ خَلَقَۚ ١ خَلَقَ الْاِنْسَانَ مِنْ عَلَقٍۚ ٢ اِقْرَأْ وَرَبُّكَ الْاَكْرَمُۙ ٣ الَّذِيْ عَلَّمَ بِالْقَلَمِۙ ٤ عَلَّمَ الْاِنْسَانَ مَا لَمْ يَعْلَمْۗ ٥

"*Bacalah dengan (menyebut) nama Tuhanmu yang menciptakan! Dia menciptakan manusia dari segumpal darah. Bacalah! Tuhanmulah Yang Maha Mulia, yang mengajar (manusia) dengan pena. Dia mengajarkan manusia apa yang tidak diketahuinya" (QS Al-'Alaq 96: 1 – 5)*.

Dari rangkaian ayat 1 sampai 5 pada Al-Qur'an surah Al-'Alaq tersebut Allah menegaskan bahwa "Allah mengajarkan kepada manusia apa yang tidak diketahui". Menurut tafsir *tahlili* NU Online juga dijelaskan bahwa Allah memberi kemampuan kepada manusia untuk bisa menggunakan alat tulis untuk menuliskan temuannya sehingga dapat dibaca oleh orang lain dan generasi berikutnya. Hal ini semakin memperkuat kondisi bahwa Allah sangat mendorong manusia untuk memaksimalkan sarana penyampaian ilmu demi kemajuan pengetahuan dan sekaligus membuktikan bahwa integrasi yang terdapat pada ayat tersebut relevan dengan konteks penelitian ini yang mencoba menghadirkan layanan teknologi informasi berupa chatbot untuk memudahkan para mahasiswa dalam mengakses pengetahuan akademik secara tepat dan cepat.

Al-Qur'an juga menegaskan pentingnya medium bahasa yang dapat dipahami manusia. Pada Surah Yusuf ayat 2, Allah berfirman:

إِنَّا أَنْزَلْنَاهُ قُرْآنًا عَرَبِيًّا لَعَلَّكُمْ تَعْقِلُونَ

"Sesungguhnya Kami menurunkannya berupa Al-Qur'an berbahasa Arab agar kamu memahaminya." (QS. Yūsuf 12: 2).

Dikutip dari Tafsir Tahlili NU Online, ayat ini menekankan bahwa ilmu pengetahuan disampaikan dengan bahasa yang dapat dimengerti. Penelitian ini mengambil spirit tersebut, dimana sistem harus dapat "memahami" pertanyaan manusia lalu menyajikan jawaban yang bisa dipahami kembali oleh manusia. Dari sisi teknis, prinsip tersebut diwujudkan melalui proses NLP (*Natural Language Processing*) yaitu teknik yang memetakan bahasa alami ke bentuk terstruktur

(*embedding*) sehingga komputer dapat memprosesnya, dan kemudian mengembalikan informasi yang jelas, terarah, dan sesuai konteks institusional.

Di samping itu, Islam juga menekankan pentingnya menjaga kebenaran informasi. Allah sendiri berfirman di dalam Al-Qur'an seperti berikut:

يٰٓاَيُّهَا الَّذِيْنَ اٰمَنُوْٓا اِنْ جَاۤءَكُمْ فَاسِقٌۢ بِنَبَاٍ فَتَبَيَّنُوْٓا اَنْ تُصِيْبُوْا قَوْمًاۢ بِجَهَالَةٍ فَتُصْبِحُوْا عَلٰى مَا فَعَلْتُمْ نٰدِمِيْنَ

"Wahai orang-orang yang beriman, jika seorang fasik datang kepadamu membawa berita penting, maka telitilah kebenarannya agar kamu tidak mencelakakan suatu kaum karena ketidaktahuan(-mu) yang berakibat kamu menyesali perbuatanmu itu". (QS Al-Ḥujurāt 49: 6).

Dikutip dari tafsir *tahlili* NU Online juga dijelaskan bahwa pada ayat ini Allah memberikan pedoman bagi kaum mukmin untuk senantiasa berhati-hati dalam menerima berita dan juga mengecek kebenarannya terlebih dulu sebelum sebelum meyakini berita yang disampaikan. Prinsip *tabayyun* inilah yang menjadi landasan normatif untuk memastikan jawaban yang disajikan bersumber dari sumber yang benar, nyata, dan tidak menyesatkan. Pada konteks chatbot, prinsip *tabayyun* ini diterapkan dengan melandaskan setiap jawaban yang dihasilkan tetap pada koridor dan konteks sumber institusional yang sah serta terverifikasi. Dua dasar yaitu akses terhadap ilmu dan verifikasi kebenaran inilah yang menjadi pijakan integrasi Islam dalam penelitian ini.

Untuk dapat memenuhi kebutuhan penyediaan informasi akademik dengan mekanisme interaktif dan tetap berpijak pada kedua dasar utama prinsip Islam (menyampaikan ilmu dan tabayyun), maka diperlukan teknologi yang cerdas. Di sinilah peran teknologi *Artificial Intelligence* (AI) yang saat ini berkembang pesat, khususnya *Natural Language Processing* (NLP) hadir. Dimana dengan kehadiran

teknologi cerdas tersebut nantinya akan memudahkan potensi terciptanya sistem interaktif tadi yaitu dengan mengimplementasikan pengembangan chatbot cerdas.

Chatbot sendiri banyak jenisnya, salah satunya adalah chatbot konvensional *rule-based*. Chatbot jenis ini memang dapat digunakan, akan tetapi memiliki keterbatasan dalam memahami konteks dan menghasilkan respons yang natural. Sementara itu, model bahasa besar (*Large Language Model*/LLM) seperti GPT, Claude, dan Gemini mampu memberikan pemahaman bahasa yang lebih baik, meski demikian LLM ini masih menghadapi tantangan berupa halusinasi, yakni menghasilkan jawaban yang terlihat sangat meyakinkan padahal tidak akurat atau tidak sesuai dengan konteks kebutuhan institusi. Disinilah letak tantangannya. Bagaimana kemungkinan timbulnya halusinasi dari jawaban yang dihasilkan chatbot tersebut dapat diminimalisir bahkan dicegah, sehingga respon yang dihasilkan akan tetap bersandar pada dua nilai islam yang sudah dibahas pada paragraf sebelumnya dan akurat sesuai informasi resmi dari institusi yang menggunakan.

Maka dari itu, *Retrieval-Augmented Generation* (RAG) di sini dihadirkan. RAG hadir sebagai solusi yang dapat menggabungkan kekuatan LLM dengan akurasi informasi yang bersumber dari *knowledge base* lokal. Teknologi RAG memungkinkan chatbot untuk mengambil informasi relevan dari basis data vektor yang berisi *embedding* dari konten website prodi, kemudian menggunakan informasi tersebut sebagai konteks untuk menghasilkan respons yang akurat dan sesuai dengan kondisi spesifik di Program Studi Teknik Informatika UIN Malang tanpa keluar dari konteks *knowledge base* yang sudah diberikan sebelumnya.

Pendekatan ini juga telah terbukti efektif dalam berbagai konteks pendidikan, pengembangan chatbot sebagai dukungan mahasiswa menunjukkan bahwa RAG mengurangi halusinasi dan meningkatkan keterkaitan jawaban dengan sumber domain-spesifik, sehingga lebih andal untuk layanan informasi akademik (Oreški & Vlahek, 2024). Pada penelitian Soliman et al. (2025) terbukti bahwa prototipe RAG untuk dukungan pembelajaran di perguruan tinggi meraih tingkat jawaban benar hingga 87% sehingga dinilai cukup berhasil dalam meningkatkan ketepatan informasi. Di sisi lain implementasi di chatbot berbasis RAG di sekolah menengah juga menunjukkan akurasi yang sangat tinggi sebesar 100% pada dua model yang digunakan yaitu (LlaMA-3-8B-Instruct dan Mistral-7B-Instruct-v0.3) dengan jumlah pertanyaan sebanyak 30 (Elysia et al., 2024). Sementara, studi layanan akademik calon mahasiswa di UCIC melaporkan kualitas respons yang baik (ROUGE-1 0,50; ROUGE-L 0,48) dengan arsitektur Python–LangChain–FAISS–GPT (Sugiarto et al., 2025).

Berdasarkan temuan-temuan tersebut, penelitian ini diarahkan untuk dapat menjembatani kesenjangan yang ada dengan mengimplementasikan chatbot berbasis RAG yang terintegrasi dengan informasi website Prodi Teknik Informatika UIN Malang. Sistem ini diharapkan dapat berfungsi menjadi asisten virtual yang dapat membantu mahasiswa memperoleh informasi akademik secara cepat, akurat, dan interaktif, sekaligus meningkatkan efisiensi layanan serta memperkaya pengalaman pengguna dalam mengakses informasi resmi program studi. Selain memberikan manfaat praktis bagi mahasiswa dan pengelola program studi,

penelitian ini juga diharapkan dapat menjadi kontribusi akademik dalam penerapan teknologi kecerdasan buatan pada bidang pendidikan tinggi di Indonesia.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam penelitian ini adalah:

a. Bagaimana merancang dan mengimplementasikan sistem chatbot berbasis Retrieval-Augmented Generation (RAG) untuk menyediakan layanan informasi akademik Prodi Teknik Informatika UIN Malang?
b. Bagaimana performa chatbot berbasis RAG dalam menjawab pertanyaan akademik mahasiswa berdasarkan konteks informasi dari website prodi?

1.3 Batasan Masalah

Lingkup data yang digunakan pada penelitian ini hanya mencakup konten publik dari website prodi (profil, kurikulum, dosen, laboratorium, fasilitas, akreditasi, beasiswa, dan lainnya), tidak termasuk data internal pada siakad.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah:

a. Merancang dan mengimplementasikan sistem chatbot berbasis *Retrieval-Augmented Generation* (RAG) yang dapat memberikan layanan informasi akademik Prodi Teknik Informatika UIN Malang.
b. Menganalisis performa chatbot dalam memberikan respons yang akurat, relevan, dan kontekstual berdasarkan informasi dari website prodi.

1.5 Manfaat Penelitian

Dengan hadirnya penelitian, diharapkan dapat memberikan kontribusi nyata dalam pengembangan ilmu pengetahuan di bidang Sistem Informasi, khususnya pada penerapan metode *Retrieval-Augmented Generation* (RAG) dalam pengembangan chatbot berbasis data akademik. Memudahkan mahasiswa mengakses informasi akademik secara interaktif, cepat, dan akurat melalui chatbot Informasi Akademik Prodi Teknik Informatika UIN Malang yang selama ini hanya dapat diakses secara pasif. Sementara, pengelola program studi juga akan terbantu dalam mengurangi beban pertanyaan berulang, serta pihak akademisi dan peneliti juga dapat menjadikan penelitian ini sebagai referensi untuk penelitian lanjutan di bidang sistem informasi berbasis kecerdasan buatan. Dengan demikian, penelitian ini layak untuk dilakukan karena memiliki nilai kebermanfaatan baik secara akademik maupun praktis.

BAB II

STUDI PUSTAKA

2.1 Penelitian Terkait

Penelitian Elysia et al. (2024) mengembangkan chatbot berbasis RAG untuk layanan informasi sekolah menengah, dengan memanfaatkan model LlaMA/Mistral/Zephyr. Fokus penelitian terletak pada aksesibilitas informasi pendidikan bagi siswa dan orang tua. Pendekatan ini menekankan kemudahan pengguna dalam memperoleh informasi akademik, dan berhasil menunjukkan bahwa RAG dapat diadaptasi di domain pendidikan non-kampus secara efektif.

Kemudian Pujiono et al. (2024) dalam Jurnal JITK Nusa Mandiri membahas penerapan RAG yang dikombinasikan dengan basis data vektor untuk chatbot. Mereka menilai performa sistem melalui metrik retrieval yang terukur, sehingga arsitektur RAG + *vector* DB dapat dipertanggungjawabkan secara kuantitatif. Kontribusinya memperlihatkan validitas pendekatan RAG sebagai solusi yang lebih unggul daripada retrieval klasik seperti BM25.

Putro et al. (2025) di Infomedia PNL membangun chatbot RAG untuk layanan aduan pelanggan PLN. Penelitian ini menekankan penggunaan LLM lokal berbahasa Indonesia dan menghasilkan tingkat kebermanfaatan jawaban sebesar 92%. Studi ini menunjukkan bahwa RAG tidak hanya relevan di domain akademik, tetapi juga di sektor pelayanan public dengan hasil yang cukup meyakinkan.

Sugiarto et al., (2025) menyajikan implementasi chatbot berbasis RAG untuk melayani informasi akademik bagi calon mahasiswa dalam proses penerimaan mahasiswa baru (PMB). Konteksnya yang dekat dengan kebutuhan

akademik Indonesia membuat penelitian ini sangat relevan, terutama karena menunjukkan bagaimana RAG dapat mempercepat akses informasi kampus.

Penelitian oleh Nur'aini (2024) merancang chatbot informasi kesehatan mental berbasis RAG dengan memanfaatkan LlaMA3 dan *create_retrieval_chain*. Penelitian ini memperlihatkan implementasi RAG *end-to-end* dalam platform web, serta menekankan pentingnya akses informasi yang sensitif melalui sistem yang responsif dan faktual.

Sementara Prasetyo (2024) menyoroti pengembangan chatbot untuk informasi pembangunan Kota Semarang. diterapkan untuk memperkuat distribusi informasi publik, dan penelitian ini memberi gambaran tentang ruang lingkup aplikasi RAG di tingkat pemerintahan daerah. Nilai tambahnya ada pada pola evaluasi yang digunakan untuk mengukur keberhasilan sistem.

Abudrrohman R (2024) memadukan RAG dengan GPT-4 yang diperkaya oleh *knowledge graph*. Variasi basis pengetahuan ini memberikan perspektif berbeda dari sekadar *vector database*. Dengan pendekatan graph, chatbot dapat memahami hubungan antarentitas dengan lebih baik, sekaligus menjadi rujukan alternatif dalam membahas desain *knowledge base*.

Septri (2025) mengusulkan chatbot RAG untuk layanan akademik seperti informasi skripsi dan PMB. Penelitian ini menggunakan metrik evaluasi modern, yaitu *BERTScore* dan *UniEval*, untuk mengukur kualitas jawaban. Hal ini sangat relevan bagi penelitian saya karena memberikan pembanding konkret dalam aspek pengujian dan evaluasi chatbot akademik.

Sementara itu Salsabila (2025) menyinggung pemanfaatan FAISS sebagai *knowledge base* dalam sistem chatbot berbasis RAG. Studi ini penting karena memperlihatkan penggunaan FAISS di konteks Indonesia, sehingga memperkuat argumen pemilihan FAISS sebagai *vector database* yang tepat untuk implementasi RAG.

Penelitian oleh Samudra et al. (2025) ini membangun sistem RAG dengan penekanan pada evaluasi *faithfulness* dan *semantic similarity*. Metode yang digunakan adalah kombinasi *dense retriever* (*embedding*) dan *reranker* untuk memastikan jawaban konsisten dengan dokumen sumber. Kebaruannya ada pada metrik evaluasi yang tidak hanya berbasis akurasi tetapi juga kesesuaian semantik. Hasil penelitian menunjukkan peningkatan relevansi jawaban chatbot hingga 15% dibanding baseline retrieval murni. Hasilnya RAG mampu meningkatkan kualitas jawaban chatbot dalam domain terbatas dengan evaluasi yang lebih kaya.

Kemudian Lewis et al. (2020) pada penelitiannya yang berjudul *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks* memperkenalkan arsitektur RAG pertama kali, menggabungkan *retriever* berbasis *Dense Passage Retrieval* (DPR) dengan generator *seq2seq* (BART). Kebaruan penelitian ini adalah integrasi *retrieval* langsung ke proses *decoding*, sehingga model dapat menghasilkan jawaban lebih faktual. Hasilnya menunjukkan RAG mengungguli baseline *open-domain* QA pada dataset *Natural Questions*. Penelitian ini membuktikan bahwa RAG merupakan fondasi kuat untuk aplikasi chatbot berbasis *knowledge retrieval*.

2.2 Landasan Teori

Bagian ini membahas teori-teori yang menjadi dasar pengembangan sistem, mencakup konsep chatbot, dasar NLP, arsitektur transformer dan LLM, hingga mekanisme RAG dan pengindeksan vektor yang mendukung proses pencarian konteks secara efisien.

2.2.1 Chatbot dan Klasifikasinya

Chatbot pada dasarnya adalah sistem perangkat lunak yang mampu menjawab pertanyaan manusia menggunakan teks atau suara secara otomatis, menyerupai percakapan manusia. Sistem ini sering diintegrasikan ke dalam layanan digital untuk memfasilitasi interaksi pengguna tanpa memerlukan manusia sebagai perantara (Adamopoulou & Moussiades, 2023).

Dalam literatur, klasifikasi chatbot dikembangkan berdasarkan berbagai kriteria. Salah satu pembagian yang sering dipakai adalah berdasarkan metode generasi respons: *rule-based*, *retrieval-based*, dan *generative-based* (Adamopoulou & Moussiades, 2023).

Chatbot *rule-based* bekerja berdasarkan aturan eksplisit (*pattern matching*, *decision trees*), cocok untuk domain terbatas, namun sangat rentan jika pertanyaan keluar dari aturan yang didefinisikan. Chatbot *retrieval-based* memilih respons dari kumpulan respons yang sudah ada, dengan pencocokan semantik atau leksikal (misalnya *cosine similarity*, BM25). Terakhir, chatbot *generative-based* menghasilkan respons baru lewat model pembelajaran mendalam (*deep learning*), misalnya dengan model *seq2seq* atau model bahasa besar (LLM). Ada juga chatbot dengan pendekatan *hybrid* yang menggabungkan *retrieval* dan *generative*, RAG

(*Retrieval-Augmented Generation*) sendiri adalah contoh *hybrid* popular dimana sistem bisa menarik konteks relevan dari basis pengetahuan dan sekaligus menghasilkan respons yang fleksibel.

Keunggulan (RAG) dibanding metode konvensional adalah kemampuannya menghasilkan jawaban yang lebih factual dan *grounding* ke sumber eksternal, sekaligus mengurangi risiko *hallucination* yang umum pada model generatif murni (Klesel & Wittmann, 2025).

Sejalan dengan perkembangan teknologi, konsep interaksi manusia dengan sistem digital (dalam konteks ini adalah chatbot) dapat dianalogikan dengan gambaran interaksi yang Allah abadikan di dalam Al-Qur'an. Misalnya, pada kisah Siti Maryam saat berinteraksi dengan malaikat Jibril yang menampakkan diri kepadanya:

فَاتَّخَذَتْ مِنْ دُوْنِهِمْ حِجَابًاۖ فَاَرْسَلْنَآ اِلَيْهَا رُوْحَنَا فَتَمَثَّلَ لَهَا بَشَرًا سَوِيًّا ﴿١٧﴾ قَالَتْ اِنِّيْٓ اَعُوْذُ بِالرَّحْمٰنِ مِنْكَ اِنْ كُنْتَ تَقِيًّا ﴿١٨﴾ قَالَ اِنَّمَآ اَنَا۠ رَسُوْلُ رَبِّكِۖ لِاَهَبَ لَكِ غُلٰمًا زَكِيًّا ﴿١٩﴾

"Maka ia mengadakan tabir (yang melindunginya) dari mereka; lalu Kami mengutus roh Kami kepadanya, maka ia menjelma di hadapannya dalam bentuk manusia yang sempurna. Maryam berkata: 'Sesungguhnya aku berlindung dari padamu kepada Tuhan Yang Maha Pemurah, jika kamu seorang yang bertakwa.' Ia (Jibril) berkata: 'Sesungguhnya aku ini hanyalah seorang utusan Tuhanmu, untuk memberimu seorang anak laki-laki yang suci.'" (QS Maryam: 17–19).

Dikutip dari tafsir tahlili NU Online bahwa pada ayat ini tergambar jelas terdapat komunikasi yang terjadi dua arah antara Siti Maryam dan malaikat Jibril, di mana menyampaikan Siti Maryam menyampaikan respon saat melihat malaikat Jibril, lalu malaikat Jibril memberikan jawaban yang menenangkan Siti Maryam

dan bersifat informatif. Hal ini memberi gambaran bahwa interaksi bukanlah sesuatu yang asing, melainkan bagian dari *sunnatullah* dalam menyampaikan pesan. Inilah bentuk interaksi yang bisa dianalogikan dengan mekanisme tanya-jawab pada chatbot, di mana pertanyaan dari pengguna itu dapat dibalas dengan jawaban yang relevan oleh sistem chatbot.

Demikian pula ketika Allah berdialog dengan para malaikat tentang penciptaan manusia pertama, Nabi Adam:

وَاِذْ قَالَ رَبُّكَ لِلْمَلٰۤىِٕكَةِ اِنِّيْ جَاعِلٌ فِى الْاَرْضِ خَلِيْفَةً ۗ قَالُوْٓا اَتَجْعَلُ فِيْهَا مَنْ يُّفْسِدُ فِيْهَا وَيَسْفِكُ الدِّمَاۤءَ ۚ وَنَحْنُ نُسَبِّحُ بِحَمْدِكَ وَنُقَدِّسُ لَكَ ۗ قَالَ اِنِّيْٓ اَعْلَمُ مَا لَا تَعْلَمُوْنَ ۝٣٠

"Ingatlah ketika Tuhanmu berfirman kepada para malaikat: 'Sesungguhnya Aku hendak menjadikan seorang khalifah di muka bumi.' Mereka berkata: 'Mengapa Engkau hendak menjadikan di bumi itu orang yang akan membuat kerusakan padanya dan menumpahkan darah, padahal kami senantiasa bertasbih dengan memuji Engkau dan mensucikan Engkau?' Tuhan berfirman: 'Sesungguhnya Aku mengetahui apa yang tidak kamu ketahui.'" (QS Al-Baqarah: 30)."

Dialog ini menunjukkan bagaimana malaikat bertanya tentang alasan Allah menurunkan khalifah di muka bumi kemudian Allah menjawab dengan argumentasi yang menegaskan hikmah di balik keputusan-Nya bahwa Allah lebih mengetahui segalanya daripada sekedar yang makhluk-Nya ketahui. Adanya interaksi tanya-jawab antara Allah dan malaikat tersebut memperlihatkan terjadi pola komunikasi yang dapat menjadi analogi juga dalam memahami fungsi chatbot yaitu adanya pengguna yang menyampaikan pertanyaan, dan sistem memberikan jawaban yang relevan.

Dengan demikian, keberadaan chatbot dalam layanan akademik dapat dipahami sebagai bentuk rekayasa teknologi yang meniru pola komunikasi

interaktif, yaitu adanya pertukaran informasi antara penanya dan pemberi jawaban. Analogi ini menekankan bahwa chatbot bukan sekadar alat pasif, melainkan menjadi suatu sarana interaksi yang aktif dan dinamis, sebagaimana interaksi antara manusia dengan malaikat digambarkan dalam Al-Qur'an.

Sebagai catatan terminologi, kata "chatbot" sendiri tidak selalu memiliki padanan spesifik dalam KBBI, tapi bisa digolongkan sebagai *bot* interaktif di domain percakapan digital.

2.2.2 Dasar NLP (*Natural Languange Processing*)

Sebelum teks bisa dipahami dalam model komputasi, langkah awal yang mutlak adalah tokenisasi memecah teks menjadi unit lebih kecil yang disebut "token". Token bisa berupa kata, *subword*, atau karakter, tergantung metode yang dipakai. Tokenisasi memfasilitasi pemrosesan teks agar tidak diperlakukan sebagai string panjang tak terstruktur (Airbyte, 2024).

Setelah tokenisasi, teks umumnya melalui tahap normalisasi, meliputi, konversi ke huruf kecil (*lowercasing*), penghapusan karakter non-alfanumerik (simbol, tanda baca yang tidak penting), penghapusan spasi ekstra atau karakter yang tidak bermakna. Tujuan normalisasi adalah menyatukan format agar "Rumah," "rumah" dan "rumah!" tidak dianggap entitas berbeda. Beberapa penelitian memperlihatkan bahwa meskipun *pipeline embedding* modern sangat kuat, tahap praproses tetap memberikan kontribusi terhadap akurasi (Siino et al., 2024).

Setelah itu, token-token tersebut diwujudkan ke bentuk numerik melalui representasi vektor atau *embedding* inti dari NLP modern. Sebelumnya, metode

representasi seperti *one-hot encoding* atau TF-IDF sering digunakan, tetapi memiliki keterbatasan dalam menangkap makna. *Embedding* (representasi padat) memungkinkan penyajian teks dengan dimensi tetap yang menyimpan konteks semantik dan hubungan antar kata. Sebuah survei menyebut bahwa bidang NLP telah bergeser dari representasi eksplisit dan statistik ke representasi kontekstual yang dipelajari via model *neural* (Patil et al., 2023).

2.2.3 Arsitektur Transformer

Arsitektur Transformer diperkenalkan oleh Vaswani et al. (2017) melalui makalah *Attention is All You Need*. Model ini didesain untuk memproses data sekuensial (seperti teks) tanpa ketergantungan berurutan seperti pada RNN atau LSTM. Komponen inti dari transformer adalah *self-attention*, yaitu mekanisme yang memungkinkan setiap token dalam sebuah kalimat memperhatikan token lain untuk membangun representasi kontekstual yang lebih kaya.

Secara matematis, *self-attention* dihitung dengan tiga proyeksi vektor: *Query* (Q), *Key* (K), dan *Value* (V). Formula perhatian adalah:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

Keterangan:
Q= *Query*,
K= *Key*,
V = *Value*,
d_k = dimensi *vector key*.

Persamaan ini menjelaskan bagaimana bobot relevansi antar token dihitung untuk menghasilkan representasi baru. Nilai QK^T menghasilkan skor kesesuaian antar token, kemudian hasilnya dinormalisasi dengan *softmax* sehingga menjadi bobot

perhatian. Bobot ini kemudian dikalikan dengan V untuk menghasilkan representasi baru yang sudah memperhatikan relasi semantik antar token.

Untuk meningkatkan kapasitas model, transformer menggunakan *Multi-Head Attention* (MHA), di mana beberapa *self-attention* berjalan paralel. Hasil dari tiap "*head*" digabungkan, sehingga model dapat menangkap beragam jenis hubungan semantik sekaligus. Agar urutan kata tetap dikenali, Transformer menambahkan *positional encoding* ke *embedding input*.

1. Encoder

Encoder terdiri dari beberapa lapisan yang masing-masing memiliki *Multi-Head Self-Attention* dan *feed-forward network*. Encoder bertugas mengubah input teks menjadi representasi vektor yang padat dan informatif. Setiap lapisan encoder saling menumpuk, sehingga menghasilkan embedding yang makin kaya makna. Contoh model *encoder-only* adalah BERT dan E5, yang efektif digunakan untuk klasifikasi, ekstraksi fitur, dan pembuatan *embedding* untuk *retrieval*.

2. Decoder

Decoder juga terdiri dari beberapa lapisan, tetapi selain memiliki *self-attention*, decoder menambahkan *masked self-attention* agar prediksi token berikutnya hanya bergantung pada token sebelumnya (autoregresif). Selain itu, decoder memiliki *cross-attention*, yang memungkinkan decoder memperhatikan hasil dari encoder. Di antara contoh model *decoder-only* adalah GPT, LLaMA, TinyLLaMA, yang digunakan untuk menghasilkan teks baru dengan memprediksi token berikutnya secara bertahap.

3. Encoder-Decoder

Arsitektur penuh encoder–decoder menggabungkan kedua peran: encoder membangun representasi dari *input*, dan decoder menghasilkan *output* berdasarkan representasi tersebut. Arsitektur ini umum dipakai untuk tugas *machine translation* (misalnya pada model T5, mBART).

Dalam penelitian ini, kedua peran arsitektur Transformer dimanfaatkan secara terpisah sesuai kebutuhan:

a. *Encoder-only* (Sentence-Transformer/E5) digunakan untuk menghasilkan embedding dokumen dan query yang padat serta bermakna. Embedding inilah yang disimpan pada basis data vektor (FAISS) untuk mendukung retrieval.
b. *Decoder-only* (TinyLLaMA 1.1B) digunakan untuk menyusun jawaban dengan cara memanfaatkan konteks hasil retrieval. Model generatif ini dipilih karena lebih ringan secara komputasi dibandingkan LLM besar (misalnya LLaMA 7B/8B), tetapi tetap efektif jika dikombinasikan dengan RAG yang memastikan jawaban berbasis konteks domain spesifik.

Dengan kombinasi tersebut, sistem dapat memanfaatkan keunggulan encoder dalam memahami teks dan keunggulan decoder dalam menghasilkan jawaban, tanpa harus menggunakan model raksasa yang membebani perangkat.

2.2.4 *Large Language Model* (LLM)

Large Language Model (LLM) merupakan implementasi lanjutan dari arsitektur transformer. LLM dilatih pada korpus teks berukuran sangat besar, sehingga mampu mempelajari pola bahasa, relasi semantik, serta pengetahuan dunia secara luas. Dalam konteks penelitian ini, peran LLM dan encoder dipisahkan

sesuai fungsinya. Proses *embedding* teks dilakukan menggunakan sentence-transformer (E5). Sentence-Transformer *(E5)*, yang termasuk kategori *encoder-only model*, untuk menghasilkan representasi vektor yang padat dan bermakna dari dokumen maupun *query*. Sementara itu, *LLM decoder-only* seperti TinyLLaMA 1.1B digunakan pada tahap *generation* untuk menyusun jawaban berdasarkan hasil *retrieval*. Pemisahan peran ini bertujuan menjaga efisiensi komputasi sekaligus mempertahankan kualitas hasil keluaran sistem. Untuk proses e*mbedding* sendiri akan memetakan kalimat atau dokumen ke dalam ruang vektor berdimensi tinggi, sehingga memungkinkan perhitungan kesamaan semantik dengan operasi matematis sederhana seperti *cosine similarity*:

$$\cos(\theta) = \frac{q \cdot d}{|q|\,|d|} \tag{2.2}$$

dengan q sebagai vektor *query* dan d sebagai vektor dokumen.

Untuk memastikan model LLM yang nantinya digunakan tidak memberatkan perangkat keras (*hardware*), maka seperti yang sudah disebutkan pada poin sebelumnya di penelitian ini mengadopsi TinyLlama-1.1B dan tidak menggunakan model LLM besar seperti LLaMA-3 8B. TinyLlama-1.1B sendiri adalah sebuah model bahasa ringan berparameter 1,1 miliar (sesuai dengan nama detailnya) (Zhang et al., 2024). TinyLlama ini dirancang sebagai versi kompak dari arsitektur LLaMA 2, dilatih pada sekitar 1 triliun token selama 3 epoch dan memanfaatkan optimasi seperti *FlashAttention* untuk efisiensi komputasi ekstra. Kelebihan utama TinyLlama ialah performa lebih baik dibandingkan model *open-source* yang seukuran pada banyak tugas *downstream*, meskipun ukurannya kecil.

Dalam penelitian ini, TinyLlama berfungsi sebagai model generatif (*decoder-only*) yang akan menerima konteks hasil *retrieval* dari basis pengetahuan prodi dan akan menghasilkan jawaban yang relevan. Karena konteks sudah "dipersempit" melalui *retrieval*, maka beban komputasi model generatif menjadi lebih ringan. Hal ini lebih memungkinkan bagi TinyLlama meskipun ukurannya jauh lebih kecil dibandingkan dengan LLaMA 8B dan tetap mampu memberikan jawaban yang memadai dalam domain akademik lokal.

Namun, tetap perlu dicatat bahwa tentu terdapat keterbatasan, dimana model kecil umumnya hanya memiliki kapasitas memori internal yang lebih terbatas, sehingga kemungkinan penurunan kualitas *reasoning* kompleks atau "lupa konteks panjang" lebih tinggi dibanding dengan model besar. Maka dari itu, strategi *prompting*, pemotongan konteks, dan pengaturan *top-k* harus dioptimalkan agar hasil (*output*) tetap konsisten dan tidak melantur.

Dengan mempertimbangkan kemampuan perangkat (komputer/Colab), TinyLlama-1.1B menjadi pilihan paling realistis dibanding model besar seperti LLaMA 3.1 8B atau Mistral 7B. Jadi, seluruh skripsi ini akan mengacu pada *pipeline* RAG yang memadukan encoder (E5 / Sentence-Transformer) untuk *embedding* dan TinyLlama sebagai model generatif ringan, daripada menggunakan model generatif raksasa yang tidak *feasible* di lingkungan penelitian ini.

2.2.5 *Retrieval-Augmented Generation* (RAG)

Retrieval-Augmented Generation (RAG) adalah paradigma baru dalam pemrosesan bahasa alami yang menggabungkan dua komponen utama: *retriever*

dan *generator* (Oreški & Vlahek, 2024). *Retriever* bertugas mencari dokumen relevan dari basis pengetahuan eksternal (misalnya vektor *database*), sedangkan *generator* (biasanya LLM) menggunakan dokumen tersebut sebagai konteks tambahan untuk menghasilkan jawaban. Dengan cara ini, RAG meminimalkan risiko *hallucination* yang sering muncul pada model generatif murni karena jawaban selalu didasarkan pada informasi faktual yang diambil dari *knowledge base* (Lewis et al., 2020). Di samping itu, dengan adanya RAG maka akan memastikan jawaban yang dihasilkan tetap pada pada konteks *knowledge base* dan tidak keluar dari konteks tersebut.

Model bahasa besar (LLM) seperti GPT atau LLaMA dapat menghasilkan teks yang fasih, tetapi sering kali berisiko mengarang fakta karena keterbatasan data latih. Di sisi lain, pendekatan *retrieval* tradisional seperti BM25 tetap memiliki peran penting dalam proses *information retrieval*, khususnya pada tahap *reranking* untuk memastikan hasil pencarian paling relevan berada di posisi teratas. RAG menggabungkan kekuatan keduanya dengan menghasilkan jawaban faktual yang didukung dokumen relevan, sembari mempertahankan gaya bahasa alami dari model generatif. Arsitektur ini memungkinkan pemisahan yang efisien antara komponen *retriever* dan *generator*, sehingga pembaruan basis pengetahuan cukup dilakukan di sisi *retrieval/vector database* tanpa perlu melatih ulang model bahasa besar.

RAG telah diuji pada berbagai domain mulai dari pendidikan Elysia et al. (2024) membuktikan RAG dapat mempermudah akses informasi sekolah. Kemudian di bidang pelayanan publik Putro et al. (2025) menunjukkan efektivitas

RAG di sektor aduan pelanggan PLN dengan akurasi tinggi, dan tentu di bidang akademik Sugiarto et al. (2025) menyoroti potensi RAG untuk layanan penerimaan mahasiswa baru. Studi-studi ini memperkuat bahwa RAG relevan untuk domain pendidikan tinggi, termasuk untuk chatbot informasi akademik program studi.

Alasan RAG masih relevan dan penting dalam penelitian ini diantaranya yaitu dari sisi program studi sendiri data akademik tentu akan mengalami perubahan yang dinamis baik (kurikulum, dosen, prestasi, beasiswa, dan informasi terkait akademik lainnya), sehingga *retrieval* menjadi solusi fleksibel. Kemudian dari sisi faktualitas, dengan *retrieval*, jawaban chatbot tidak berdiri pada data latih lama, melainkan selalu didukung *knowledge base* terbaru. Dan dari sisi fleksibilitas, sistem tetap bisa menggunakan LLM *open-source* (misalnya LLaMA), namun *knowledge base* bisa diperbarui hanya di *retriever*.

Pada Tabel 2.1 berikut disajikan alur umum RAG. Alur ini secara berurutan dimulai dari tahap *query encoding,* kemudian dilakukan proses *retrieval* pada korpus data, setelah itu hasil top-k pada proses itu akan dipilih dimana dalam hal ini sudah memasuki tahap augmentation. Terakhir masuk ke tahap generation jawaban oleh LLM berdasarkan hasil proses *augmentation*.

Tabel 2.1 Alur Umum Retrieval-Augmented Generation RAG

Tahap	Deskripsi
Query Encoding	pertanyaan pengguna diubah menjadi vektor
Retrieval	vektor *query* dicocokkan dengan koleksi dokumen (*vector DB*, misalnya FAISS atau *ElasticSearch*).
Augmentation	dokumen dengan skor kesesuaian tertinggi (Top-k) dipilih sebagai konteks tambahan.
Generation	model generatif memproduksi jawaban berdasarkan *query* dan konteks tersebut.

Penelitian yang dilakukan oleh Prastowo et al. (2025) turut menegaskan efektivitas RAG dalam meningkatkan kualitas *customer service*. Hasil studinya menunjukkan bahwa kombinasi mekanisme *retrieval* dan *generation* mampu menghasilkan respons yang lebih kontekstual dan konsisten dibandingkan model *retrieval-based* murni. Temuan ini memperlihatkan kemampuan RAG untuk menggabungkan relevansi informasi dengan gaya respons alami, bahkan dalam domain dengan data dinamis.

2.2.6 *Text Embedding / Encoder*

Tahap inti dalam *pipeline* RAG adalah representasi teks ke dalam bentuk vektor yang bisa dihitung secara matematis. Proses ini dilakukan oleh *encoder*. *Encoder* memetakan kalimat atau dokumen ke *embedding* berdimensi tetap, misalnya 768 dimensi pada model E5. *Embedding* ini menyimpan representasi semantik sehingga dua teks dengan makna mirip akan berada berdekatan di ruang vektor (Reimers & Gurevych, 2019).

Salah satu model populer adalah Sentence-Transformers, misalnya keluarga E5 (*Enhanced Embeddings for Information Retrieval*). Model E5 dilatih dengan pendekatan *contrastive learning* untuk memaksimalkan kedekatan embedding antar pasangan teks yang relevan (Wang et al., 2024).

Secara matematis, *embedding* hasil *encoder* biasanya dinormalisasi dengan *L2 normalization:*

$$\hat{E}(t) = \frac{E(t)}{|E(t)|_2} \tag{2.3}$$

Sehingga, kesamaan antara *query* q dan dokumen d dapat dihitung dengan *cosine similarity:*

$$\cos(q, d) = \hat{E}(q) \cdot \hat{E}(d) \tag{2.4}$$

Contoh sederhana penerapan *embedding* dapat dilihat pada proses ketika pengguna mengajukan pertanyaan ke chatbot. Misalnya, pengguna menanyakan "Kapan waktu pelaksanaan PKL untuk mahasiswa Informatika?". Kalimat pertanyaan ini kemudian diubah menjadi representasi vektor, misalnya [0.45, 0.28, –0.12, …]. Di sisi lain, sistem juga telah menyimpan vektor-vektor dari dokumen atau teks sumber, seperti "PKL dilaksanakan pada semester 6 selama minimal 1 bulan", yang mungkin dipetakan ke vektor [0.47, 0.31, –0.10, …]. Karena jarak kosinus antar kedua vektor tersebut sangat kecil (misalnya 0.94), sistem menilai bahwa dokumen tersebut relevan dengan pertanyaan pengguna.

Dengan demikian, proses *embedding* tidak hanya memetakan kalimat sederhana seperti "Mahasiswa Teknik Informatika" ke vektor [0.12, 0.34, –0.07, …], tetapi juga memungkinkan sistem mengenali kedekatan makna antara pertanyaan dan jawaban yang berbeda secara redaksi namun serupa secara semantik. Implementasi *text embedding* ini menjadi fondasi penting bagi proses *indexing* pada FAISS serta perhitungan skor relevansi di tahap *retrieval*.

2.2.7 *Vector Search* dan FAISS

Vector search adalah teknik pencarian dokumen berbasis kesamaan semantik, bukan sekadar pencocokan kata kunci. Prinsip dasarnya yaitu setiap dokumen d dan *query* q diproyeksikan ke ruang vektor berdimensi tetap oleh

encoder. Kemudian, skor kesamaan dihitung (misalnya *cosine similarity* atau *inner product*), dan dokumen dengan skor tertinggi dianggap paling relevan.

Untuk mengatasi skalabilitas, digunakan *Approximate Nearest Neighbor* (ANN) *indexing*. ANN memungkinkan pencarian *Top-k* dokumen relevan dari jutaan entri dengan efisiensi tinggi, tanpa harus menghitung jarak ke seluruh dokumen. Salah satu 25emanti ANN 25emanti adalah FAISS (*Facebook AI Similarity Search*) (Douze et al., 2025). FAISS mendukung berbagai skema indexing, misalnya: *Flat Index*: *brute-force search*; akurat tetapi lambat. IVF (*Inverted File Index*) membagi ruang vektor ke dalam *centroids* sehingga pencarian hanya dilakukan pada subset. HNSW (*Hierarchical Navigable Small World*) struktur graf yang mempermudah pencarian tetangga terdekat (Malkov & Yashunin, 2018).

Efektivitas FAISS dalam konteks bahasa Indonesia juga telah diuji pada penelitian Ramadhan et al. (2024) yang mengimplementasikan *Passage Retrieval* untuk sistem *Question Answering* (QA). Dengan memanfaatkan *embedding* BERT dan FAISS sebagai *vector index*, penelitian tersebut berhasil mencapai akurasi hingga 72,5% setelah proses *fine-tuning*, sekaligus mempertahankan waktu eksekusi rata-rata hanya 0,23 detik per pertanyaan (Ramadhan et al., 2024). Hasil ini menunjukkan bahwa FAISS tidak hanya unggul secara teoritis dalam mendukung skema ANN, tetapi juga terbukti praktis dalam aplikasi pencarian semantik berbasis bahasa Indonesia.

Penggunaan *vector database* dalam penelitian ini dimaksudkan untuk mengelola dan menata informasi sehingga lebih mudah ditemukan kembali secara

efisien. Konsep pencatatan dan penyimpanan informasi ini memiliki kesesuaian dengan prinsip yang digambarkan dalam Al-Qur'an. Allah berfirman:

مَا يَلْفِظُ مِنْ قَوْلٍ اِلَّا لَدَيْهِ رَقِيْبٌ عَتِيْدٌ ﴿١٨﴾

"Tidak ada suatu kata yang diucapkannya melainkan di sisinya ada malaikat pengawas yang selalu hadir." (QS Qāf: 18).

Ayat ini menunjukkan bahwa setiap ucapan manusia dicatat oleh malaikat Raqīb dan 'Atīd secara rapi dan terstruktur. Bahkan, Al-Qur'an sendiri ditegaskan tersimpan di *Lauh Mahfuzh* sebagai bentuk pencatatan ilahi yang sempurna.

بَلْ هُوَ قُرْاٰنٌ مَّجِيْدٌ ﴿٢١﴾ فِيْ لَوْحٍ مَّحْفُوْظٍ ﴿٢٢﴾

"Bahkan, (yang didustakan itu) Al-Qur'an yang mulia, yang (tersimpan) dalam (tempat) yang terjaga (Lauh Mahfuz)." (QS Al-Burūj: 22).

Hal ini dapat dianalogikan dengan peran *vector database* yang berfungsi bukan sekadar menyimpan data, tetapi juga menata dan mengindeksnya sehingga informasi dapat dengan mudah ditemukan kembali saat dibutuhkan. Dengan demikian, penerapan FAISS dan metode *indexing* dalam penelitian ini sejalan dengan prinsip pencatatan yang sistematis sebagaimana digambarkan dalam Al-Qur'an.

Alur kerja FAISS sendiri secara garis besar meliputi *build time* yaitu semua *embedding* dokumen disimpan dalam index (misalnya IVF), setelah itu masuk ke tahap *query time* dimana *embedding query* dihitung kemudian dicocokkan dengan *index*, terakhir dokumen relevan dengan nilai Top-k akan dikembalikan. Contoh

sederhana: misalnya ada 3 dokumen (d1, d2, d3) dengan *embedding* tersimpan di FAISS. Saat *query q* masuk, FAISS mencari *embedding* dengan jarak kosinus terdekat dan mengembalikan dokumen [d2, d1] sebagai Top-2.

2.2.8 Pengujian *Retrieval* dan *Generation*

Evaluasi merupakan aspek penting dalam penelitian RAG, karena kualitas sistem tidak hanya ditentukan oleh kemampuannya mengambil dokumen (*retrieval*), tetapi juga bagaimana jawaban akhir dibangkitkan (*generation*). Oleh sebab itu, pengujian dilakukan dua level yaitu *Retrieval* (IR) dan *Generation* (NLG) *Natural Language Generation*).

2.2.8.1 Pengujian *Retrieval*

Pada level retrieval, yang dinilai adalah sejauh mana sistem berhasil menemukan dokumen relevan dari koleksi yang tersedia (Guo et al., 2020). Dua metrik utama adalah *Recall@k* dan *Mean Reciprocal Rank* (MRR).

Recall@k menilai berapa banyak dokumen relevan yang berhasil muncul dalam Top-k hasil pencarian. Misalnya, jika ada 5 dokumen relevan dan sistem mengembalikan 3 di antaranya pada *Top*-5, maka *recall@5* = 3/5 = 0,6. Metrik ini penting karena pengguna biasanya hanya melihat hasil teratas, semakin tinggi skor *recall@k*, maka semakin baik sistem memastikan konteks relevan tidak terlewat (Shiri, 2004).

Mean Reciprocal Rank (MRR) mengukur seberapa cepat dokumen relevan pertama muncul pada daftar hasil. Jika sebuah *query* memiliki dokumen relevan di posisi pertama, skor *reciprocal rank* = 1; jika di posisi ketiga, maka skor = 1/3.

MRR menghitung rata-rata *reciprocal rank* di seluruh *query*. Dengan demikian, MRR memberi gambaran seberapa cepat sistem menghadirkan jawaban yang tepat di depan mata pengguna.

2.2.8.2 Pengujian Generasi Jawaban

Setelah *retrieval*, tahap berikutnya adalah generasi jawaban oleh LLM. Evaluasi di tahap ini menilai aspek kualitas, koherensi, dan kesesuaian jawaban.

a. *Faithfulness* menilai apakah klaim dalam jawaban sesuai dengan dokumen sumber yang di-*retrieve* (Samudra et al., 2025). Jika 4 dari 5 klaim sesuai dokumen, maka *faithfulness* = 0,8. Metrik ini dinyatakan sebagai berikut.

$$\text{Faithfulness} = \frac{\text{jumlah klaim benar}}{\text{total klaim}} \tag{2.6}$$

   Metrik ini krusial untuk RAG karena tujuan utamanya adalah mengurangi halusinasi.

b. *Semantic Similarity* dihitung dengan *cosine similarity* antar *embedding* jawaban sistem dan jawaban referensi. Metrik ini membantu mengukur kedekatan makna tanpa harus membandingkan token satu per satu.

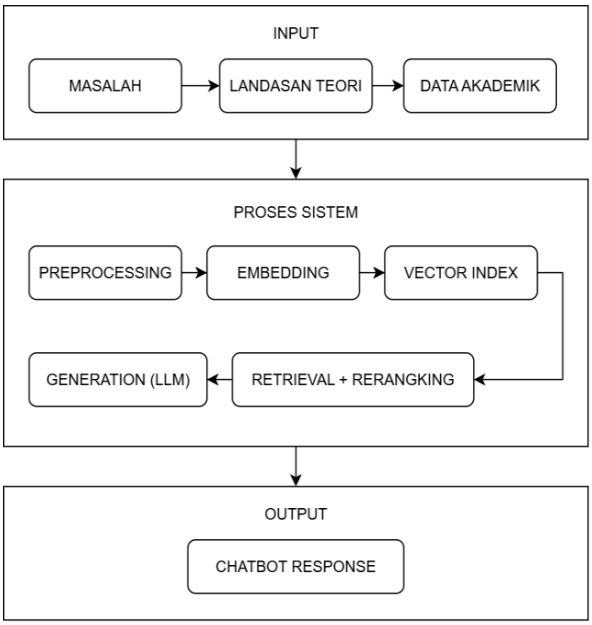
Untuk memudahkan, evaluasi dapat diilustrasikan sebagai berikut. Misalnya, untuk *Query* Q1 dengan dokumen relevan {D2, D5}, sistem mengembalikan Top-3 {D2, D4, D1}. Maka *recall@3* = 1/2 = 0,5.

2.3 Kerangka Pemikiran

Pada penelitian ini, permasalahan utama adalah bagaimana menyediakan layanan chatbot akademik yang mampu memberikan jawaban relevan berdasarkan

data internal Program Studi Teknik Informatika UIN Malang. Untuk menjawab masalah tersebut digunakan pendekatan *Retrieval-Augmented Generation (RAG)* yang memadukan dua komponen utama, yaitu *retrieval* dan *generation*.

Dari Gambar 2.1 berikut dapat dipahami, tahap pertama, data akademik dikumpulkan melalui proses *web scraping* dari situs resmi program studi. Data mentah tersebut kemudian melalui tahap *preprocessing* untuk dibersihkan dari elemen yang tidak relevan (seperti tag HTML, format yang berlebihan, maupun *stopwords*). Setelah itu, dilakukan proses *text embedding* menggunakan model *Sentence-Transformers* (E5) yang mengubah teks ke dalam representasi vektor numerik. Representasi ini kemudian disimpan dalam basis data vektor (FAISS) agar dapat dilakukan pencarian berbasis kesamaan.



Gambar 0.1 Kerangka Pemikiran Penelitian

Pada saat pengguna memberikan pertanyaan (*query*), sistem akan melakukan tahap retrieval untuk mengambil potongan dokumen paling relevan

berdasarkan ukuran kesamaan kosinus (*cosine similarity*). Hasil dari tahap *retrieval* dapat ditingkatkan kualitasnya melalui proses *reranking* menggunakan algoritma BM25 untuk mengutamakan konteks dengan tingkat kemiripan tertinggi. Selanjutnya, dokumen hasil *reranking* dipadukan dengan pertanyaan pengguna dan diproses oleh model bahasa besar (LLM) untuk menghasilkan jawaban yang lebih koheren dan kontekstual.

Kerangka pemikiran ini menegaskan bahwa penggunaan RAG memungkinkan chatbot tidak hanya menghasilkan jawaban berbasis *memorization* model, tetapi juga memanfaatkan basis pengetahuan spesifik prodi. Hal ini diharapkan meningkatkan akurasi, kontekstualitas, dan kredibilitas jawaban chatbot. Dengan demikian, penelitian ini secara konseptual membangun hubungan antara teori NLP, teknik *information retrieval*, serta kebutuhan praktis layanan informasi akademik.

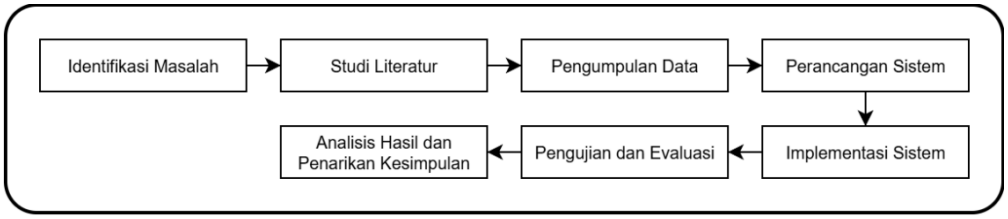
BAB III

DESAIN DAN IMPLEMENTASI

3.1 Desain Penelitian

Penelitian ini menggunakan pendekatan penelitian terapan (*applied research*). Tujuannya adalah untuk merancang, mengimplementasikan, dan mengevaluasi sebuah teknologi, yaitu sistem chatbot informasi akademik berbasis *Retrieval-Augmented Generation* (RAG), guna menyelesaikan permasalahan praktis yang dihadapi mahasiswa dalam mengakses informasi di website Prodi Teknik Informatika UIN Malang. Dengan adanya desain penelitian yang jelas maka akan membantu penulis melakukan penelitian secara terarah dan tepat.

Secara garis besar, alur penelitian akan mengikuti *flowchart* yang digambarkan pada Gambar 3.1.



Gambar 3.1 Desain Penelitian

Kerangka kerja metodologis penelitian ini dirancang secara sistematis untuk memastikan setiap tahapan dilakukan secara terstruktur dan terukur. Secara garis besar, tahapan penelitian ini dimulai dari identifikasi masalah untuk menentukan kebutuhan penelitian, dilanjutkan dengan studi literatur untuk mengumpulkan referensi dan landasan teori yang relevan. Setelah itu, proses pengumpulan data dilakukan untuk mendapatkan informasi yang diperlukan sebagai dasar dari tahap perancangan sistem. Selanjutnya adalah tahap perancangan sistem dan

implementasi sistem dilakukan. Kemudian sistem yang sudah selesai dibuat tersebut memasuki tahap pengujian, dan evaluasi. Tahap ini dilakukan dengan tujuan untuk mengukur kinerja dan ketepatan dari solusi yang dihasilkan sistem. Tahap terakhir adalah analisis hasil dan penarikan kesimpulan berdasarkan data yang diperoleh dari seluruh proses penelitian yang sudah dilakukan (Pratama & Sisephaputra, 2024). Pendekatan ini memungkinkan pengembangan teknologi yang tidak hanya berfungsi tetapi juga teruji dan dievaluasi secara komprehensif, memberikan solusi praktis terhadap permasalahan yang diidentifikasi.

3.2 Pengumpulan Data

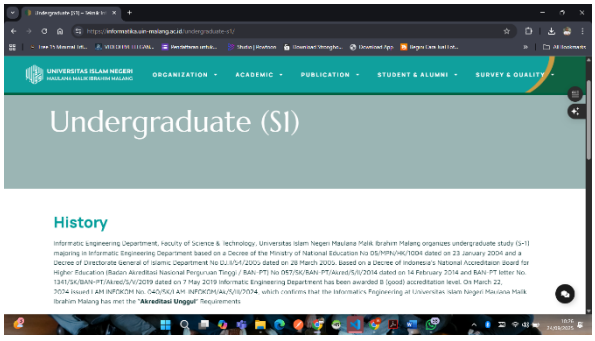
Tahap pengumpulan data ini merupakan langkah fundamental dalam penelitian ini yang bertujuan untuk membangun basis pengetahuan (*knowledge base*) yang komprehensif untuk sistem chatbot. Berbeda dengan penelitian yang memanfaatkan dataset sekunder yang sudah ada, penelitian ini menggunakan data primer yang dikumpulkan secara langsung dari sumber utamanya. Sumber data yang digunakan dalam penelitian ini bersifat tunggal dan spesifik, yaitu seluruh konten tekstual publik yang tersedia di website resmi Program Studi Teknik Informatika, Universitas Islam Negeri Maulana Malik Ibrahim Malang, yang dapat diakses pada (*https://informatika.uin-malang.ac.id*).

Proses pengumpulan data dilakukan dengan menggunakan teknik *web scraping* secara otomatis. Sebuah skrip pemrograman yang dirancang untuk menavigasi dan mengekstrak informasi tekstual dari berbagai halaman relevan di dalam domain website tersebut. Lingkup data yang diambil mencakup, namun tidak terbatas pada, informasi mengenai kurikulum, profil dosen, prosedur akademik,

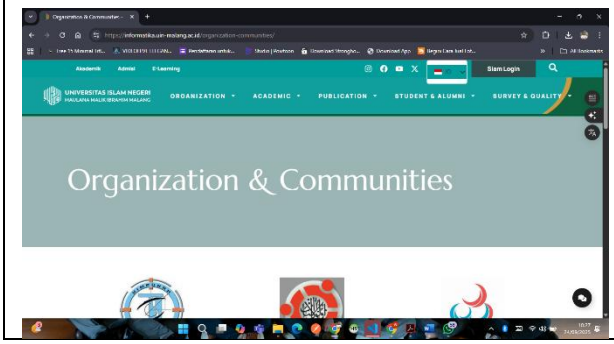
panduan skripsi, informasi beasiswa, serta berita dan pengumuman. Pembatasan pengumpulan data hanya pada domain website resmi prodi, ini dilakukan untuk menjamin validitas, relevansi, dan konteks informasi yang akan menjadi dasar jawaban chatbot.

Hasil dari tahap ini adalah sebuah korpus data mentah (*raw data*) yang berisi seluruh informasi tekstual yang berhasil diekstraksi. Kumpulan data mentah inilah yang kemudian akan menjadi input untuk tahap pra-pemrosesan data sebelum diindeks ke dalam *database* vektor dan diintegrasikan dengan *Large Language Model* (LLM) dalam *pipeline* RAG. Untuk memberikan gambaran yang lebih jelas mengenai cakupan halaman yang menjadi sumber data, beberapa contoh halaman disajikan dalam Tabel 3.1 di bawah ini.

Tabel 3.1 Tampilan Beberapa Halaman Website Prodi yang dilakukan *web scraping*

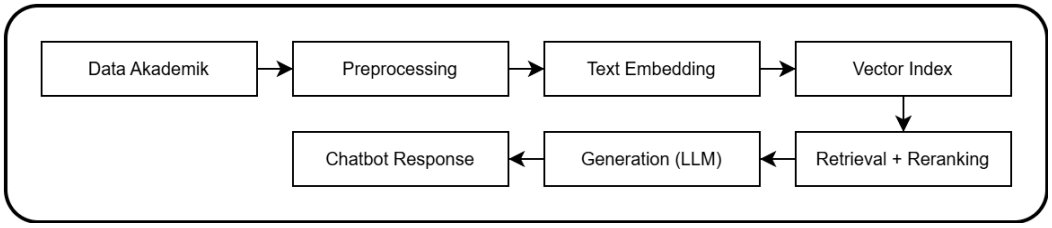
Gambar Tangkapan Layar Website Prodi	Keterangan
	Halaman *Landing Page* dari website prodi Teknik Informatika UIN Malang
	Halaman Profil S1 dari website prodi Teknik Informatika UIN Malang

Tabel 3.1 Lanjutan

Gambar Tangkapan Layar Website Prodi	Keterangan
	Halaman Organisasi dan Komunitas dari website prodi Teknik Informatika UIN Malang

3.3 Desain Sistem

Pada Gambar 3.2 berikut disajikan gambaran umum desain sistem. Desain sistem dilakukan dengan tujuan untuk mendefinisikan secara jelas bagaimana desain sistem dari chatbot yang akan dikembangkan.

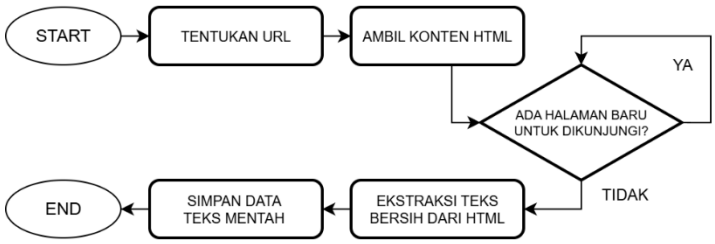


Gambar 3.2 Alur Desain Sistem

Data awal berupa data akademik program studi Teknik Informatika yang sudah diambil dari website resmi program studi, kemudian memasuki tahap *preprocessing,* dilanjutkan ke proses *text embedding* dan *vector indexing*. Setelah itu baru proses inti RAG yaitu meliputi *retrieval* dan terakhir adalah *generation* dimana tahap ini akan menghasilkan *response* (jawaban).

3.3.1 Web Scraping

Tahap ini ialah tahap awal yang sangat penting dalam siklus pengembangan sistem RAG, bertujuan untuk mengumpulkan data tekstual secara otomatis yang akan menjadi fondasi dari basis pengetahuan (*knowledge base*) chatbot. Proses ini dirancang untuk secara sistematis mengambil seluruh konten informatif dari sumber data tunggal yang telah ditetapkan, yaitu website resmi Program Studi Teknik Informatika UIN Malang. Metode ini dipilih karena efisiensinya dalam mengakuisisi data dalam volume besar langsung dari sumber aslinya, memastikan otentisitas dan relevansi informasi yang akan digunakan oleh sistem. Alur kerja dari proses *web scraping* ini diilustrasikan secara visual pada Gambar 3.3.



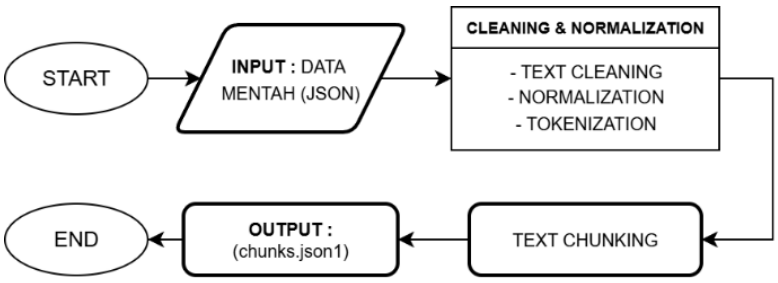
Gambar 3.3 Alur *Web Scraping*

Proses web *scraping* diawali dengan menginisialisasi sebuah *scraper* yang diprogram menggunakan Python untuk melakukan permintaan HTTP dan mengambil struktur dokumen HTML dari halaman-halaman yang relevan di domain *(https://informatika.uin-malang.ac.id)*. Setiap halaman yang telah ditentukan diakses secara langsung tanpa melakukan penelusuran rekursif ke seluruh tautan sebagaimana pada teknik *crawling*. Skrip *scraping* kemudian mengekstrak konten teks dari elemen-elemen HTML tertentu seperti paragraf (<p>), judul (<h1>, <h2>), dan item daftar (<li>) yang mengandung informasi

akademik utama. Selama proses ekstraksi, dilakukan tahap pembersihan untuk menghapus elemen yang tidak relevan seperti tag skrip, menu navigasi, atau spasi berlebih, sehingga data yang tersimpan adalah teks bersih dan siap digunakan. Hasil teks bersih tersebut selanjutnya disimpan dalam format terstruktur, seperti JSON atau CSV, agar dapat diproses lebih lanjut pada tahap *preprocessing* sebelum dilakukan *embedding* dan penyimpanan ke basis data vektor.

3.3.2 Preprocessing

Tahap *preprocessing* merupakan kelanjutan dari tahap pengumpulan data (*web scraping*), yang bertujuan untuk mengubah data mentah yang telah diekstraksi menjadi format yang bersih, terstruktur, dan siap untuk memasuki tahap *embedding* dan indeksasi. Proses ini juga penting untuk memastikan kualitas input bagi model *embedding* dan efektivitas *retrieval* di tahap mendatang. *Input* untuk tahap ini adalah kumpulan data teks mentah dalam format JSON (misalnya, *data/raw/*.json*) yang dihasilkan dari proses *web scraping* sebelumnya. Proses detail pra-pemrosesan teks digambarkan pada Gambar 3.4 berikut.



Gambar 3.4 Alur Tahap *Preprocessing*

Alur *preprocessing* dimulai dengan tahap pemuatan data dalam format JSON yang sebelumnya telah terkumpul. Setelah itu, data melewati proses

pembersihan (*cleaning*) yang menyeluruh, mencakup penghapusan tag HTML yang masih tersisa, penghilangan spasi ganda, serta eliminasi karakter non-teks yang tidak memberikan makna. Pada tahap berikutnya, data dapat dinormalisasi, misalnya melalui penyamaan bentuk huruf menjadi *lowercase*, agar variasi penulisan lebih seragam dan mudah diproses. Setelah bersih dan seragam, teks diuraikan menjadi satuan-satuan yang lebih kecil dan bermakna. Proses ini sering kali diwujudkan dalam bentuk pemecahan kalimat (*sentence splitting*) atau paragraf (*paragraph splitting*), yang secara alami mempersiapkan teks untuk langkah pengolahan berikutnya. Selanjutnya setiap kata dilakukan tokenisasi secara formal.

$$T = \{w_1, w_2, \dots, w_n\}, \quad t = \text{tokenizer}(\text{text}) \tag{3.1}$$

Dari unit-unit kecil tersebut, dilakukan *chunking*, yaitu menggabungkan kembali kalimat atau paragraf menjadi segmen teks dengan panjang yang optimal, umumnya berkisar antara 200–350 token. Agar informasi tidak terputus, setiap *chunk* diberi irisan konteks (*overlap*) sekitar 20–30% dengan *chunk* sebelumnya. Strategi ini membantu menjaga kesinambungan makna dan mencegah hilangnya detail penting. Secara algoritmik yaitu dengan cara mengambil *window* yang dapat dinotasikan dengan notasi berikut, dengan panjang L token.

$$[i, i + L) \tag{3.2}$$

Jadi, misalkan pada penelitian ini digunakan L = 300 token, maka sudah melalui perhitungan kata atau tanda pemisah (*whitespace* dan *punctuation*) agar cukup panjang untuk menjaga konteks, tapi tidak melebihi batas jendela model. Kemudian mulai dari token pertama, ambil L token dari indeks ke-i sampai $i + L$. Notasi $[i, i + L)$ berarti mulai dari token i (inklusif) sampai $i + L$ (eksklusif).

Window tidak langsung melompat ke token berikutnya setelah $i + L$, tapi digeser sebagian dengan rumus *stride* berikut agar ada *overlap*.

$$stride = L \times (1 - overlap) \quad (3.3)$$

Lalu untuk *overlap* dibuat 20%, berarti $stride = 300 \times (1 - 0.2) = 240$. Jadi *window* kedua mulai dari token ke-241, bukan ke-301. Dilanjutkan dengan *window* kedua mencakup token [241, 541) → tetap 300 token. *Window* ketiga [481, 781), dan seterusnya. Proses ini akan berlangsung secara berulang sampai semua token habis. *Chunk* terakhir mungkin berisi token lebih sedikit (<L) jika teks tidak habis dibagi rata.

Sebagai ilustrasi lebih lengkap, penulis menggambarkan pada Tabel 3.2 berikut. Misal teks menghasilkan 1000 token setelah tokenisasi, dengan L = 300 dan *overlap* 20%:

Tabel 3.2 Ilustrasi Pembagian *Chunk* dan Cakupan Token

Pembagian *Chunk*	Cakupan Token
Chunk 1	[1 … 300]
Chunk 2	[241 … 540]
Chunk 3	[481 … 780]
Chunk 4	[721 … 1000]

Dari Tabel 3.2 dapat dipahami bahwa dengan 1000 token menghasilkan 4 *chunk*, dengan masing-masing terdapat irisan (*overlap* 20%) dengan *chunk* sebelumnya.

Selain teks, setiap *chunk* diperkaya dengan informasi tambahan berupa metadata, seperti alamat URL, judul halaman, dan bagian spesifik dari mana teks tersebut diambil. Metadata ini berperan penting untuk keperluan rujukan maupun validasi di tahap analisis berikutnya. Hasil akhir dari rangkaian ini disimpan dalam format JSON *Lines* (JSONL), misalnya pada berkas *data/chunks/chunks.jsonl*, di

mana setiap baris memuat objek JSON berisi *id, url, title*, serta teks dari *chunk* yang telah diproses.

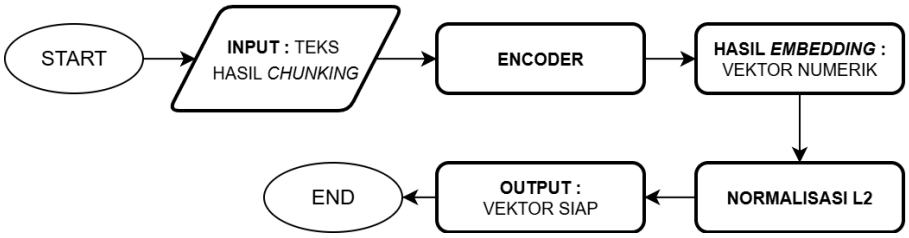
3.3.3 Text Embedding

Setelah teks dipecah menjadi potongan (*chunks*), tiap *chunk* harus diubah ke bentuk numerik agar bisa dibandingkan secara matematis di dalam *vector database*. Proses ini dilakukan dengan bantuan *encoder*. *Encoder* adalah model *neural network* yang bertugas memetakan teks (berupa kata atau kalimat) menjadi vektor berdimensi tetap. Vektor ini merepresentasikan makna semantik teks, sehingga kalimat dengan arti mirip akan memiliki vektor yang letaknya dekat di ruang vektor. Pada tahap ini model *open source* yaitu *Sentence-Transformers* (E5) ini dipilih karena lebih superior dalam menghasilkan representasi vektor yang semantik dan sadar konteks untuk unit teks yang panjang, seperti kalimat dan paragraf. Keunggulan ini menjadi krusial jika dibandingkan dengan metode alternatif. Metode TF-IDF, yang berbasis frekuensi statistik, tidak mampu menjembatani kesenjangan leksikal, artinya ia tidak akan mengenali hubungan antara *query* "biaya kuliah" dengan dokumen yang berisi frasa "Uang Kuliah Tunggal". Sementara itu, *Word2Vec*, meskipun mampu menangkap relasi semantik antar kata, memiliki keterbatasan signifikan karena representasi kalimatnya seringkali hanya merupakan agregasi (misalnya, rata-rata) dari vektor kata-kata penyusunnya, sehingga mengabaikan struktur sintaksis dan konteks yang kompleks. Sebaliknya, *Sentence-Transformers*, yang dibangun di atas arsitektur Transformer, mampu memproses seluruh urutan teks secara bersamaan. Hal ini

memungkinkannya untuk menghasilkan *embedding* tunggal yang padat makna, yang secara efektif menjadi fondasi bagi proses pencarian kemiripan (*similarity search*) yang akurat dalam *pipeline* RAG.

Pilihan ini memastikan bahwa kontribusi ilmiah utama dari penelitian ini terpusat pada perancangan arsitektur dan evaluasi alur kerja sistem RAG secara menyeluruh, yang mencakup tahap pra-pemrosesan data, *indexing*, *retrieval*, hingga *generation* respons. Hal ini merupakan inti dari penyelesaian masalah yang diajukan, alih-alih berfokus pada pengembangan model *embedding* fundamental yang berada di luar cakupan penelitian.

Ilustrasi alur pada tahap *text-embedding* ini terdapat pada Gambar 3.5 berikut. Sebelum proses proses *embedding* dipastikan dulu datanya sudah dalam bentuk potongan dokumen (*chunking*).



Gambar 3.5 Alur Tahap *Text Embedding*

Dengan *Sentence-Transformers* (E5), setiap teks t diproyeksikan oleh *encoder* ke ruang vektor berdimensi d (umumnya 768).

$$E(t) = f_\theta\big(\text{tokenize}(t)\big) \in R^d \tag{3.4}$$

Keterangan:
$E(t)$: embedding dari teks t, berupa vektor,
f_θ : fungsi *encoder* dengan parameter terlatih,
d : dimensi tetap (misal 768).

Agar konsisten dalam perhitungan kesamaan, *embedding* dinormalisasi dengan *L2 normalization*:

$$\hat{E}(t) = \frac{E(t)}{|E(t)|_2} \tag{3.5}$$

Dimana $|E(t)|_2 = \sqrt{\sum_{i=1}^{d}(E(t)_i)^2}$

Ini akan menghasilkan semua *embedding* punya panjang vektor 1 (unit vektor). Kemudian kesamaan semantik antar dua teks (*query* q dan dokumen d) didefinisikan dengan *cosine similarity* bentuk umum:

$$\cos\big(E(q), E(d)\big) = \frac{E(q) \cdot E(d)}{|E(q)|_2 \, |E(d)|_2} \tag{3.6}$$

Rumus ini selalu berlaku sebelum normalisasi. Sementara itu, pada penelitian ini semua *embedding* sudah dinormalisasi L2, sehingga berlaku:

$$|\hat{E}(q)|_2 = |\hat{E}(d)|_2 = 1 \tag{3.7}$$

Sehingga jarak/kesamaan antar teks bisa dihitung dengan rumus *cosine similarity* yang yang menjadi lebih sederhana yaitu

$$\cos\left(\hat{E}(q), \hat{E}(d)\right) = \hat{E}(q) \cdot \hat{E}(d) \tag{3.8}$$

Penulis menyajikan ilustrasi pada Tabel 3.3. Misalkan *encoder* menghasilkan vektor 3 dimensi (untuk contoh sederhana saja, ukuran aslinya 768 dimensi).

Tabel 3.3 Ilustrasi Vektor Pendek Proses *Embedding*

Teks	Vektor awal $\{E(t)\}$	Normalisasi L2 $\hat{E}(t)$
"kurikulum TI"	[0.21, 0.34, -0.11]	[0.49, 0.79, -0.26]
"144 SKS TI"	[0.15, -0.28, 0.39]	[0.31, -0.58, 0.81]

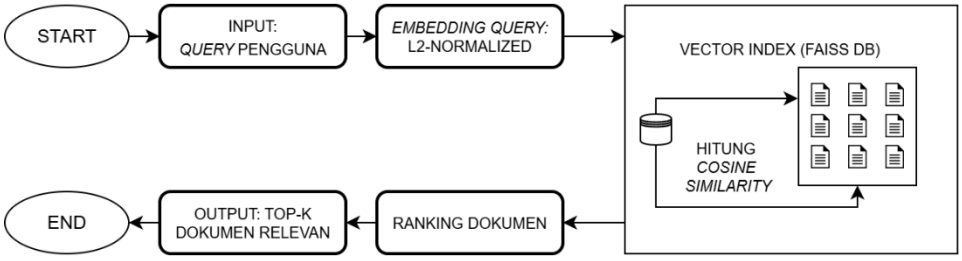
Berikutnya dihitung *cosine similarity* dari kedua vektor hasil normalisasi:

$$cos = 0.49 \times 0.31 + 0.79 \times (-0.58) + (-0.26) \times 0.81 = -0.20$$

Hasilnya didapatkan nilai negatif yang artinya makna antar dua teks kurang relevan.

3.3.4 Pengindeksan Vektor (*Vector Index*)

Ilustrasi proses tahap *embedding* digambarkan pada Gambar 3.6. Setelah setiap potongan teks (*chunk*) direpresentasikan menjadi vektor berdimensi tetap melalui tahap *text embedding*. Langkah berikutnya adalah menyusun indeks vektor agar pencarian k-tetangga terdekat (k-NN) bisa dilakukan cepat dan terukur. Pada penelitian ini digunakan FAISS (*Facebook/Meta AI Similarity Search*) sebagai *engine* pencarian vektor, karena mendukung jutaan vektor Nur Hakim et al., (2025), memiliki banyak tipe indeks (dari yang akurat sampai yang sangat cepat), dan tersedia antarmuka Python sehingga mudah diintegrasikan dengan *pipeline* RAG.



Gambar 3.6 Alur Tahap *Vector Indexing*

Konsep dasar pada tahap ini adalah misalkan setiap dokumen/*chunk I* memiliki *embedding* ter-normalisasi $\hat{d}^{\imath} \in R^d, \quad \hat{q} \in R^d$ dan sebuah kueri pengguna direpresentasikan menjadi *embedding* ter-normalisasi $\cos(\hat{q}, \hat{d}^{\imath}) = \hat{q} \cdot \hat{d}^{\imath}$. Skor kedekatan yang digunakan adalah *cosine similarity*. Karena semua vektor sudah *L2-normalized* $|\hat{E}|_2 = 1$, *cosine similarity* ekuivalen dengan *inner product* (IP):

$$\cos(\hat{q}, \hat{d}^{\imath}) = \hat{q} \cdot \hat{d}^{\imath} \tag{3.9}$$

Dengan demikian, *ranking* dokumen relevan bisa diperoleh dengan mencari Top-K nilai $\hat{q} \cdot \widehat{d^i}$ terbesar untuk semua i. Sementara, untuk pemetaannya ke FAISS terdapat pada Tabel 3.4. Pada tabel tersebut berisi penjelasan tahap-tahap proses *vector indexing.*

Tabel 3.4 Alur Pemetaan Indeks ke FAISS

Tahap	Deskripsi
1	Menyusun sebuah indeks FAISS berisi semua vektor dokumen
2	Tipe indeks yang digunakan pada tahap awal adalah *IndexFlatIP* (*inner product*) akurat dan sederhana.
3	Karena vektor telah dinormalisasi, *IndexFlatIP* menghasilkan urutan yang sama dengan *cosine similarity* (tanpa perlu membagi oleh norma).
4	Untuk skala lebih besar, indeks bisa ditingkatkan ke IVF (*Inverted File*) atau HNSW untuk mempercepat, dengan kompromi akurasi (*approximate nearest neighbor*).

Catatan: Pada penelitian ini digunakan *IndexFlatIP* sebagai metode indexing utama karena memiliki tingkat akurasi tinggi dengan mekanisme pencarian berbasis inner product (Krisnawati et al., 2024). Pendekatan ini sederhana untuk direplikasi dan sangat cocok digunakan pada tahap awal pengembangan sistem dengan jumlah vektor relatif kecil (dimensi $d = 768$, sesuai keluaran model *Sentence-Transformer*). Apabila ukuran data meningkat secara signifikan (jumlah vektor N bertambah besar), maka arsitektur dapat diskalakan dengan mengganti *index type* ke IVF (*Inverted File Index*) atau HNSW (*Hierarchical Navigable Small World*). Kedua metode ini memungkinkan peningkatan efisiensi pencarian melalui *approximate nearest neighbor search* dengan tetap menjaga keseimbangan antara kecepatan dan akurasi.

Pada Tabel 3.5 berikut disajikan bagaimana alur teknis proses indeksasi vektor secara singkat.

Tabel 3.5 Alur Teknis *Build Time* Indeksasi Vektor ke FAISS

Tahap	Deskripsi
1	*Load* semua *chunk* dari data/chunks/chunks.jsonl.
2	*Embed* setiap *chunk* dengan model (misal E5), lalu hasilnya dilakukan *L2-normalization*.
3	Inisialisasi indeks. *index = faiss.IndexFlatIP(dim)*
4	Tambahkan seluruh *embedding* ke indeks: *index.add(embeddings)* *# embeddings shape: [N, d], sudah dinormalisasi*
5	Simpan indeks dan metadata secara terpisah. *data/index/chunks.faiss* → struktur FAISS (hanya vektor). *data/index/meta.json* → daftar metadata yang urutannya sejajar dengan vektor (id, url, title, text, dan sebagainya).

Kemudian pada Tabel 3.6 berikut disajikan deskripsi dari tahap awal sampai akhir dari proses teknis pencarian indeks vektor pada FAISS.

Tabel 3.6 Alur Teknis *Vector Index Searching* pada FAISS

Tahap	Deskripsi
1	Terima pertanyaan pengguna, bentuk *embedding query* ter-normalisasi $\hat{q}$ (format E5: "*query: <teks>*")
2	Lakukan pencarian Top-K di indeks: scores, idxs = index.search(q_vec, k) # inner product FAISS mengembalikan dua matriks: scores (nilai *IP/cosine*) dan idxs (indeks baris vektor dokumen).
3	Gunakan *idxs* untuk mengambil metadata terkait dari meta.json (url, judul, cuplikan teks).
4	Kirim daftar kandidat ini ke tahap *reranking* (opsional) atau langsung ke *prompting* (jika tanpa *reranker*).

Jika disusun semua vektor dokumen sebagai baris matriks $D \in R^{N \times d}$ (setiap baris adalah $\widehat{d_i}$), dan vektor kueri sebagai kolom $\hat{q} \in R^{d \times \mathbb{1}}$, maka skor kedekatan seluruh dokumen ke *query* dapat dihitung sebagai:

$$s = D\hat{q} \in R^{N \times \mathbb{1}}, \quad s_i = \widehat{d^i} \cdot \hat{q} \tag{3.10}$$

FAISS melakukan operasi serupa secara sangat efisien (dengan optimasi C++/SIMD/GPU). Proses Top-K berarti mengambil K nilai s_i terbesar beserta indeksnya.

Untuk menjaga keamanan dan konsistensi skor, maka perlu dipastikan *normalisasi L2* diterapkan baik pada *embedding* dokumen saat *build* maupun *embedding query* saat *query*. Ini membuat *inner product* setara dengan *cosine*, sehingga interpretasi skor konsisten. Jika tidak dinormalisasi, *inner product* akan bergantung pada panjang vektor, bukan hanya arah (makna), sehingga *ranking* bisa bias.

Penyimpanan indeks FAISS (*file* dengan format *.faiss*) bersifat *read-only* dalam penelitian ini, artinya jika korpus data diperbarui, maka *pipeline rebuild* indeks perlu dijalankan ulang. Sementara untuk metadata sendiri disimpan di berkas meta.json (atau Parquet) agar mudah dibaca ulang saat *serve*. Di *production*, metadata bisa dipindah ke *database* (misal *Postgres/Supabase*) untuk pemeliharaan yang lebih nyaman.

Terdapat Tabel 3.7 yang menyajikan catatan penting untuk memudahkan penjelasan lebih eksplisit terkait tipe indeks FAISS, dapat dilihat pada tabel berikut.

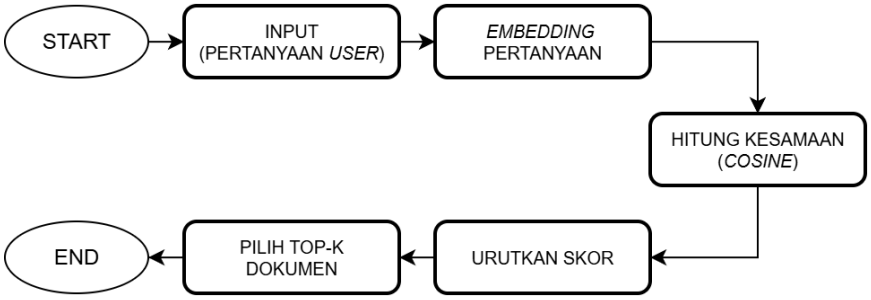
Tabel 3.7 Catatan Penting Tipe Indeks FAISS

Tipe Indeks	Deskripsi
IndexFlatIP	*exact search* berbasis *inner product*; akurat, cocok untuk N kecil–menengah
IVF (*Inverted File*)	membagi ruang vektor ke beberapa *centroid*; mempercepat pencarian pada N besar (kompromi akurasi)
HNSW	*graph-based ANN* (*Approximate Nearest Neighbor*) dengan performa sangat baik untuk skala besar.

3.3.5 First Stage Retrieval

Setelah *embedding* dokumen disimpan dalam *vector index* kemudian masuk tahap berikutnya yaitu *first stage retrieval*. Alur pada tahap ini dapat dilihat pada Gambar 3.7. *Embedding* dari *query* pengguna ($\hat{q}$) dibandingkan dengan *embedding* dokumen yang tersimpan dalam indeks ($\hat{d}_i$) menggunakan ukuran kesamaan.

Pencarian dilakukan untuk menemukan *Top-k* dokumen dengan skor kesamaan tertinggi (Soliman et al., 2025).



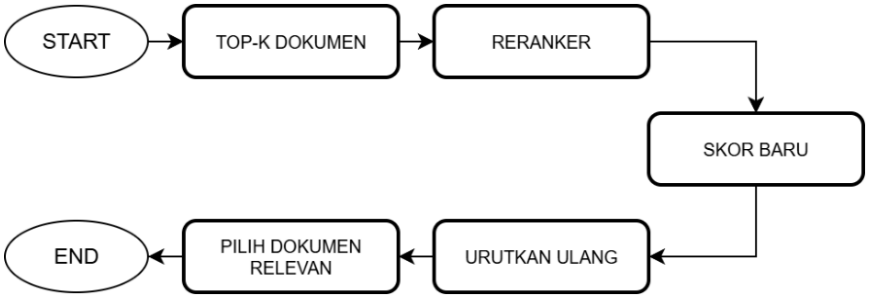
Gambar 3.7 Alur Tahap *First Stage Retrieval*

Dengan *embedding* yang telah dinormalisasi menggunakan L2, perhitungan *cosine similarity* pada tahap *retrieval* dapat disederhanakan menjadi operasi *dot product* antar vektor. Oleh karena itu, peringkat dokumen relevan (Top-k) dapat langsung diperoleh dari hasil skor kesamaan. Ilustrasi perhitungan manual telah disajikan pada tahap sebelumnya (*Vector Indexing*), di mana dokumen dengan skor tertinggi diprioritaskan sebagai hasil *first stage retrieval.* Jadi, kesamaan dihitung menggunakan *cosine similarity,* yang dirumuskan persis seperti persamaan 3.9 yang terdapat pada tahap *embedding*.

Dengan menggunakan FAISS, proses ini dapat dilakukan secara efisien meskipun jumlah dokumen sangat besar. Sistem kemudian mengembalikan daftar kandidat *Top-k* dokumen yang paling relevan. Hasil dari tahap ini bersifat *coarse retrieval*, artinya masih mungkin ada dokumen yang urutannya kurang tepat, sehingga diperlukan tahap selanjutnya yaitu *reranking.*

3.3.6 Reranking

Tahap ini bertujuan untuk menyempurnakan urutan dokumen yang dihasilkan dari tahap *first stage retrieval*. Alur proses *reranking* disajikan pada Gambar 3.8. Dari *Top-k* kandidat dokumen, sistem mengevaluasi kembali tingkat relevansi menggunakan metode yang lebih presisi, meskipun lebih mahal secara komputasi.



Gambar 3.8 Alur Tahap *Reranking*

Dalam penelitian ini digunakan kombinasi pendekatan *embedding* dengan algoritma berbasis *lexical matching*, seperti BM25, yang merupakan salah satu algoritma *probabilistic retrieval*. Setelah hasil *retrieval* awal diperoleh dari FAISS, sistem melakukan proses *reranking* menggunakan metode BM25 untuk memprioritaskan potongan teks yang paling relevan secara leksikal dengan *query* pengguna. Skor relevansi BM25 dihitung dengan persamaan formula berikut:

$$BM25(q,d) = \sum_{t \in q} IDF(t) \frac{f(t,d) \cdot (k_1 + 1)}{f(t,d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{avgdl}\right)} \tag{3.11}$$

Keterangan lengkap dari rumus BM25 dapat dilihat pada Tabel 3.8 berikut.

Tabel 3.8 Keterangan Rumus BM25

Simbol	Keterangan
$f(t,d)$	frekuensi kemunculan term t dalam dokumen d
$\|d\|$	panjang dokumen d
$avgdl$	panjang rata-rata dokumen dalam koleksi

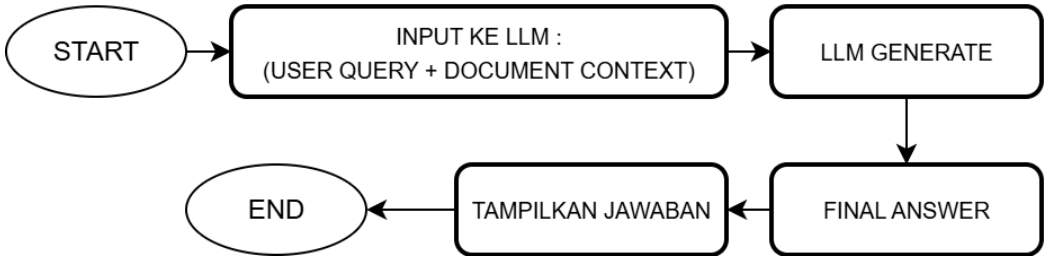
Tabel 3.8 Lanjutan

Simbol	Keterangan
$k_1\ dan\ b$	parameter pengatur (umumnya $k\_\{1\} \in [1.2, 2.0], \quad b = 0.75$)
$IDF(t)$	*inverse document frequency* dari term t

Proses *reranking* ini menggabungkan kekuatan *semantic retrieval* (*embedding similarity*) dan *lexical retrieval* (BM25). Dengan demikian, dokumen yang paling relevan baik secara semantik maupun leksikal dapat diprioritaskan (Robertson & Zaragoza, 2009).

3.3.7 Generation

Tahap terakhir adalah *generation*. Seperti yang terdapat pada Gambar 3.9 menunjukkan alur dari proses *generation* dimana ini merupakan proses dokumen hasil *retrieval* (yang sudah diurutkan kembali melalui *reranking*) digunakan sebagai konteks tambahan (*augmented context*) bagi model bahasa (LLM) untuk menghasilkan jawaban.



Gambar 3.9 Alur Tahap *Generation*

Secara teknis, model generatif menerima dua masukan yaitu *query* asli dari pengguna dan kumpulan dokumen relevan hasil *retrieval* (umumnya 3–5 dokumen teratas). LLM kemudian memproses kedua masukan tersebut dengan mekanisme atensi, sehingga jawaban yang dihasilkan bersandar pada pengetahuan yang ada dalam dokumen, bukan semata-mata pada model. Mekanisme ini membuat jawaban

lebih faktual dan mengurangi risiko *hallucination*. Formulasinya dapat digambarkan secara sederhana:

$$Answer = LLM\left(Query, Context_{Top\text{-}k}\right) \quad (3.12)$$

Keterangan:
$Query$: pertanyaan pengguna
$Context_{Top\text{-}k}$: dokumen hasil *retrieval* yang disatukan sebagai *prompt augmentation*
$Answer$: keluaran jawaban akhir dari sistem

Pada tahap *generation* ini, sistem memanfaatkan model TinyLLaMA 1.1B sebagai *decoder-only LLM* untuk menghasilkan jawaban berdasarkan potongan konteks hasil *retrieval*. Model ini dipilih karena berukuran ringan, efisien, dan dapat dijalankan pada perangkat dengan sumber daya terbatas. Berikutnya agar tahap "*generation*" berlangsung lebih solid, maka perlu dilakukan *prompting* untuk LLM sendiri dan melakukan konfigurasi sehingga panjang jawaban konteks juga disesuaikan dengan kebutuhan. Misalnya dilakukan konfigurasi seperti pada Tabel 3.9 berikut.

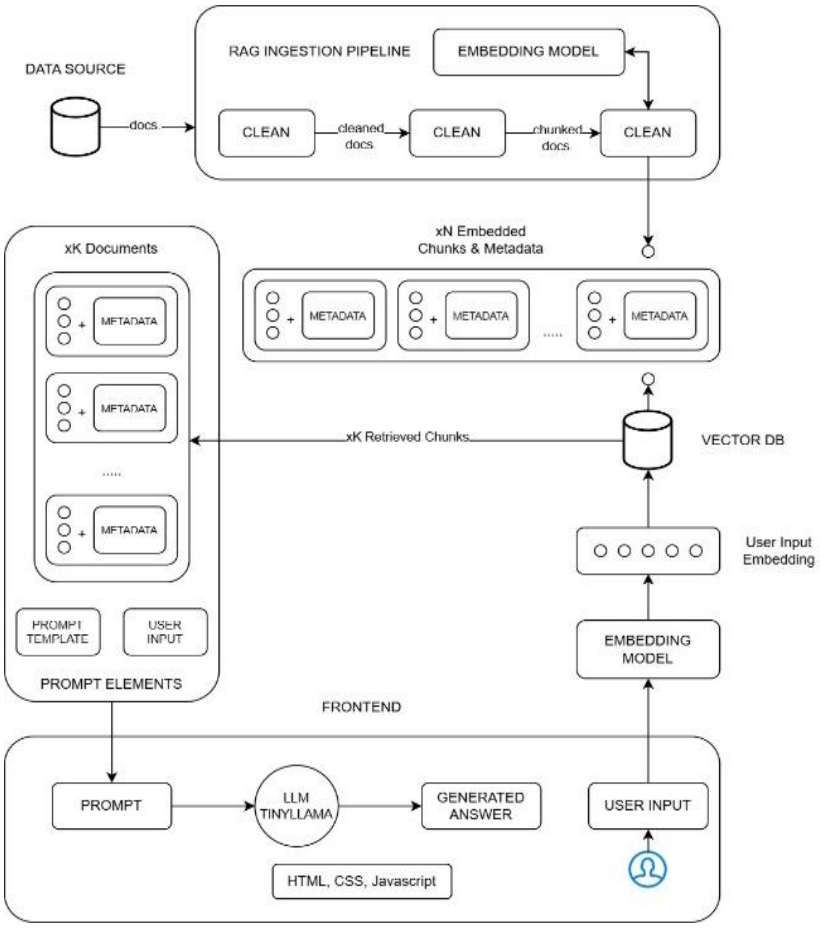
Tabel 3.9 Template Prompt dan Konfigurasi LLM

Kebutuhan Konfigurasi	Keterangan
Instruction	Jawablah pertanyaan pengguna hanya berdasarkan KONTEN yang diberikan. Jika tidak ditemukan jawabannya di KONTEN, sampaikan tidak tersedia dan di luar domain.
$KONTEN$	…(Top-k *chunk*)…
PERTANYAAN	$\{q\}$
Konfigurasi	TinyLLaMA-1.1B, *quantization* 4-bit (QloRA/gguf), max_tokens output 256, *temperature* 0.3–0.7, top-k 40–100, top-p 0.9. Panjang konteks efektif disesuaikan (49issal 2–4 potongan @300 token) agar tidak melebihi jendela model.

Adapun tahap evaluasi akan dijelaskan pada bagian skenario pengujian, menggunakan metrik seperti *Faithfulness* untuk konsistensi terhadap konteks.

3.4 Implementasi Sistem

Tahap implementasi akan mencakup pengembangan dua komponen utama: *backend* RAG dan antarmuka (*frontend*) chatbot. Gambaran implementasi dari desain sistem yang sudah diuraikan pada bagian 3.3 secara garis besar keseluruhan diilustrasikan pada Gambar 3.10 berikut. Pada gambar tersebut dapat dipahami bahwa ranah yang bisa diakses oleh user hanya sebatas pada tampilan antarmuka chatbot saja. Untuk detail mekanisme RAG sendiri akan terjadi di sisi server *backend*). Adapun jika diruntut secara singkat, *user* mengakses chatbot, mengajukan pertanyaan, lalu sistem akan mengubah *query* user menjadi bentuk vektor lalu dari nilai tersebut akan digunakan untuk melakukan pencarian dokumen relevan di *database* vektor. Top-k dokumen paling relevan akan diteruskan ke LLM beserta vektor dari *query user* tadi sebagai dasar LLM melakukan generasi jawaban. Kemudian setelah jawaban berhasil dihasilkan akan ditampilkan di antarmuka chatbot yang dapat diakses oleh *user*.



Gambar 3.10 Skema Implementasi Desain Utama Sistem Chatbot RAG

3.4.1 Perangkat Keras dan Lunak:

Perangkat keras yang digunakan selama pengembangan dan pengujian akan dilakukan pada laptop dengan spesifikasi prosesor AMD Ryzen 3, RAM 8 GB, dan sistem operasi Windows 64-bit. Sementara untuk detail dari spesifikasi perangkat lunak yang digunakan dalam penelitian ini tertera jelas pada Tabel 3.10 berikut.

Tabel 3.10 Spesifikasi Perangkat Lunak yang digunakan

Perangkat Lunak	Spesifikasi
Bahasa Pemrograman	Python 3.10+
Backend Framework	FastAPI
Open Source LLM	TinyLLaMA-1.1B
Vector Database	FAISS
Embedding	*Hugging Face* (transformers, sentence-transformers)

3.4.2 Arsitektur Sistem

Sistem akan dibangun dengan arsitektur *client-server,* yaitu *frontend* dan *backend*. *Frontend* merupakan antarmuka pengguna (UI) berbasis web yang telah Anda kembangkan (*index.html, app.js, styles.css*). Komponen ini bertanggung jawab untuk menampilkan dialog percakapan dan berkomunikasi dengan backend melalui HTTP *requests*.

Backend (RAG API) merupakan sebuah layanan API yang dibangun dengan FastAPI. API ini akan menyediakan *endpoint* (misalnya /api/chat) yang menerima pertanyaan dari *query user* di sisi *frontend*, menjalankan seluruh *pipeline* RAG (*retrieval* dari FAISS dan generasi oleh TinyLLaMA-1.1B), dan mengembalikan jawaban beserta sumbernya.

Selain pendekatan *client–server* tradisional, sistem ini juga dirancang agar dapat di-*deploy* menggunakan *Streamlit*. *Streamlit* merupakan *framework Python* yang memungkinkan pembangunan antarmuka interaktif berbasis web tanpa memerlukan pengembangan *frontend* manual. Dengan menjalankan perintah '*streamlit run'*, *backend* RAG dapat sekaligus menyajikan antarmuka chatbot yang dapat diakses langsung melalui browser. Pendekatan ini memberikan fleksibilitas yang tinggi di tahap awal penelitian. Streamlit mempermudah *prototyping* dan pengujian sistem oleh banyak pengguna, sedangkan di tahap lanjut, *frontend* dan *backend* tetap dapat dipisahkan untuk tujuan skalabilitas.

3.4.3 *Input* dan Proses

Input utama sistem adalah data tekstual tidak terstruktur yang berasal dari website Prodi Teknik Informatika UIN Malang. Lingkup data yang diambil

meliputi, namun tidak terbatas pada, halaman profil prodi, kurikulum, informasi dosen, detail kelompok keilmuan, prosedur akademik, panduan skripsi dan PKL, serta berita dan pengumuman. Selain itu, input lainnya adalah pertanyaan (*query*) dalam bahasa Indonesia yang diajukan oleh pengguna melalui antarmuka chatbot. Proses inti dalam sistem ini adalah alur kerja RAG yang telah dijelaskan pada Desain Sistem (3.3). Proses ini dapat dirinci lebih lanjut seperti pada Tabel 3.11.

Tabel 3.11 Penjelasan Proses Inti RAG

Nama Proses	Penjelasan Proses
Query Processing	Pertanyaan dari pengguna diterima melalui API.
Vector Search	Sistem melakukan pencarian pada indeks FAISS untuk menemukan dokumen-dokumen (chunks) yang paling relevan dengan pertanyaan pengguna.
Context Formulation	Dokumen-dokumen yang relevan diformat menjadi sebuah konteks.
Prompt Engineering	Sistem menyusun sebuah prompt terstruktur yang berisi instruksi untuk LLM, konteks yang relevan, dan pertanyaan asli pengguna.
LLM Inference	*Prompt* tersebut dikirim ke endpoint LLM (TinyLLaMA-1.1B) untuk menghasilkan jawaban.
Response Formatting	Jawaban dari LLM beserta sumber referensinya (dokumen yang digunakan sebagai konteks) diformat menjadi sebuah respons JSON untuk dikirim kembali ke antarmuka pengguna.

3.4.4 Output

Output yang dihasilkan oleh sistem adalah jawaban tekstual yaitu sebuah jawaban dalam format bahasa natural yang relevan dengan pertanyaan pengguna, dihasilkan oleh LLM berdasarkan konteks yang diberikan dan dilengkapi dengan sumber referensi yaitu daftar sumber (judul halaman atau URL) dari website prodi yang digunakan sebagai dasar untuk menghasilkan jawaban, guna meningkatkan transparansi dan memungkinkan pengguna untuk melakukan verifikasi.

3.5 Skenario Pengujian

Tujuan pengujian pada penelitian ini adalah untuk menilai sejauh mana sistem chatbot berbasis *Retrieval-Augmented Generation* (RAG) mampu menghasilkan jawaban yang relevan dengan pertanyaan pengguna, menyajikan jawaban yang akurat dan konsisten dengan sumber data (*knowledge base*), dan memberikan respons dalam waktu yang wajar, sehingga layak digunakan dalam skenario nyata.

Untuk skenario uji sendiri dilakukan melalui dua aspek utama, yaitu pengujian *retrieval* dan pengujian generasi jawaban (*generation*). Selain pengujian kedua aspek tadi. Beberapa metrik yang digunakan untuk mengevaluasi sistem chatbot sendiri yaitu diantaranya ada MRR (*Mean Reciprocal Rank)*, *Semantic Similarity*, dan masih ada lainnya.

Seluruh metrik evaluasi dalam penelitian ini menghasilkan skor dalam rentang 0 hingga 1. Rentang ini digunakan karena setiap nilai hasil pengukuran telah dinormalisasi. Semakin mendekati angka 1, semakin baik performa sistem baik dalam hal pencarian dokumen relevan maupun kualitas jawaban yang dihasilkan. Sebaliknya, nilai yang mendekati 0 menunjukkan rendahnya relevansi dan akurasi sistem dalam menjawab pertanyaan pengguna. Dengan rentang nilai yang terstandarisasi ini, perbandingan antar-metrik menjadi lebih objektif dan mudah dianalisis.

3.5.1 Pengujian *Retrieval*

Pengujian *retrieval* ini berfokus pada kemampuan sistem mengambil dokumen paling relevan terhadap *query* pengguna. Terdapat tiga metrik utama yang

digunakan pada pengujian ini, yaitu Recall at k, *Mean Reciprocal Rank* (MRR@k), dan *Normalized Discounted Cumulative Gain* (NDCG@k). Pengujian *Retrieval* dilakukan pada 30 pertanyaan uji yang diambil dari konten website resmi Program Studi. Variasi parameter k digunakan dengan nilai k = 3, 5, dan 10. Pemilihan variasi ini mengacu pada praktik umum dalam penelitian *Retrieval-Augmented Generation* dan *Information Retrieval*, seperti yang digunakan oleh Lewis et al. (2020) pada *Dense Retriever* dan Karpukhin et al. (2020) dalam *Dense Passage Retrieval* (DPR), di mana rentang k = 3 hingga k = 10 dianggap cukup mewakili keseimbangan antara cakupan konteks yang diambil dan relevansi hasil pencarian.

Sementara itu, nilai k = 3 mencerminkan skenario *focused retrieval* dengan konteks minimal, k = 5 menggambarkan kondisi moderat yang sering digunakan sebagai nilai default dalam sistem RAG, sedangkan k = 10 mengevaluasi performa sistem ketika cakupan konteks diperluas untuk memastikan dokumen relevan tidak terlewat. Dengan variasi ini, dapat diamati pengaruh jumlah dokumen terambil terhadap skor relevansi (Recall@k), kualitas perankingan dokumen (NDCG@k), posisi dokumen relevan pertama (MRR@k), serta konsistensi hasil generasi jawaban pada tahap berikutnya.

1. *Recall at* – k

Metrik ini mengukur proporsi dokumen relevan yang berhasil ditemukan pada k hasil teratas. *Recall@k* didefinisikan sebagai rasio jumlah dokumen relevan yang berhasil tampil dalam *k* hasil teratas (Top-k) terhadap total jumlah dokumen relevan untuk *query* tersebut. Dengan kata lain, jika sistem hanya menampilkan *k*

kandidat jawaban teratas, *Recall@k* mengukur seberapa banyak dari dokumen relevan itu berhasil tercakup dalam daftar tersebut.

$$Recall@k = \frac{\text{Jumlah dokumen relevan dalam Top-}k}{\text{Total dokumen relevan}} \tag{3.13}$$

Misalnya, untuk *query* tentang "kurikulum semester 5" ada total 4 dokumen relevan di koleksi, dan sistem menampilkan 3 di antara mereka di Top-5 → *Recall@*5 = 3/4 = 0,75.

Versi pertama (*Recall@*k dasar) tersebut digunakan untuk memahami konsep pada satu *query*, sedangkan formulasi agregat memperhitungkan semua *query* sehingga representatif untuk evaluasi sistem menyeluruh. Pendekatan agregat ini banyak digunakan dalam penelitian informasi *retrieval* dan sistem RAG modern Knollmeyer et al. (2025), seperti yang terlihat pada *framework GraphRAG* yang menggabungkan struktur graf dokumen untuk memperkuat kualitas konteks dan hasil generasi.

$$Recall@k = \frac{1}{|Q|}\sum_{q \in Q}\frac{|R_q \cap C_{q,k}|}{|R_q|} \tag{3.14}$$

Keterangan:
Q: himpunan semua *query* uji,
R_q: himpunan dokumen relevan (*ground truth*) untuk *query* q,
$C_{q,k}$: himpunan k kandidat teratas yang dikembalikan sistem,
$R_q \cap C_{q,k}$: jumlah dokumen relevan yang muncul dalam Top-k sistem.

2. *Mean Reciprocal Rank* (MRR)

Mengukur posisi relatif jawaban relevan pertama dalam hasil pencarian. Semakin tinggi nilai MRR, semakin baik karena item relevan muncul di peringkat awal.

$$MRR = \frac{1}{|Q|}\left(\sum_{i=1}^{|Q|}\frac{1}{rank_i}\right) \quad (3.15)$$

dengan $rank_i$ adalah posisi jawaban relevan pertama untuk *query* ke-i.

3. *Normalized Discounted Cumulative Gain* (NDCG@k)

NDCG@k mengukur kualitas perankingan dokumen dengan mempertimbangkan posisi dokumen relevan dalam hasil *retrieval*. Metrik ini memberikan bobot lebih tinggi pada dokumen relevan yang muncul di posisi atas. Formula NDCG@k:

$$NDCG@k = IDCG@k/DCG@k \quad 3.16$$

dengan

$$\text{DCG@}k = \sum_{i=1}^{k}\frac{rel_i}{\log_2(i+1)} \quad 3.17$$

Keterangan:
rel_i = skor relevansi dokumen di posisi ii i (1 jika relevan, 0 jika tidak)
$IDCG@k$ = ideal DCG, yaitu DCG maksimum jika semua dokumen relevan berada di posisi teratas

Rentang nilai berkisar antara 0–1 (semakin tinggi semakin baik). NDCG@5 = 0.8 berarti kualitas *ranking* sistem mencapai 80% dari kondisi ideal di mana semua dokumen relevan berada di top-5 dengan urutan sempurna.

NDCG lebih informatif dibanding Recall karena tidak hanya mengukur "apakah dokumen relevan ditemukan", tetapi juga "seberapa baik posisi perankingannya". Dokumen relevan di posisi 1 memberikan kontribusi lebih besar daripada di posisi 10.

3.5.2 Pengujian *Generation*

Selain kemampuan *retrieval*, penting untuk mengukur kualitas jawaban yang dihasilkan model pada proses generasi. Pengujian *generation* dilakukan pada subset 50 pertanyaan yang disusun dari topik akademik Prodi (kurikulum, PKL, dosen, kelompok keilmuan, layanan administrasi, dan lainnya). Setiap pertanyaan dipasangkan dengan *reference answer* yang diambil dari konten resmi website prodi (misalnya halaman kurikulum, panduan PKL, informasi dosen). *Reference answer* berfungsi sebagai jawaban acuan yang "benar" untuk perbandingan objektif terhadap keluaran chatbot. Evaluasi kuantitatif dilakukan menggunakan metrik *Answer Relevancy* dan *Faithfulness*, untuk menilai kedekatan semantik terhadap jawaban acuan dan kepatuhan pada konteks sumber.

1. *Answer Relevancy*

Metrik ini mengukur sejauh mana jawaban yang dihasilkan sistem selaras dengan maksud pertanyaan pengguna. Dengan menggunakan *cosine similarity* antara *embedding* pertanyaan (E_o) dan *embedding* jawaban (E_g), nilai AR akan semakin tinggi jika jawaban semakin relevan dengan konteks pertanyaan.

$$AR = \frac{1}{N}\sum_{i=1}^{N} \cos(E_g, E_o) \tag{3.18}$$

dengan E_g adalah *embedding* jawaban dan E_o adalah *embedding* pertanyaan. Metrik ini dipakai karena jawaban yang dihasilkan LLM/RAG belum tentu langsung relevan, sehingga perlu dibandingkan kedekatannya dengan pertanyaan.

2. *Faithfulness*

Faithfulness mengevaluasi kebenaran faktual jawaban terhadap konteks dokumen yang diambil oleh sistem. Jika jawaban menghasilkan klaim yang tidak ada atau bertentangan dengan konteks, nilainya akan menurun.

$$F = \frac{\text{jumlah klaim jawaban yang sesuai konteks}}{\text{total klaim jawaban}} \tag{3.19}$$

Metrik ini penting karena salah satu kelemahan LLM adalah halusinasi. Dengan *faithfulness*, bisa diukur apakah sistem benar-benar mengambil informasi dari basis pengetahuan yang relevan.

Klaim jawaban sendiri adalah berupa unit pernyataan faktual yang bisa diverifikasi, misalnya entitas (nama dosen/kelompok keilmuan), atribut (jumlah, tanggal, semester), relasi (X termasuk Y, syarat PKL = Z), daftar/anggota (item-item dalam list resmi). Jadi jawaban dapat dikatakan sesuai konteks artinya sebuah klaim dianggap didukung bila tercantum, tersirat kuat, atau terparafrasa setara di dalam potongan teks sumber resmi untuk pertanyaan tersebut. Pemeriksaan dukungan dilakukan per klaim (bukan per kalimat penuh).

3. *Semantic Similarity*

Metrik ini mengukur kesamaan semantik antara jawaban sistem dengan jawaban referensi (*ground truth*). Metrik ini juga dihitung dengan *cosine similarity* antara vektor *embedding* jawaban dan jawaban referensi. Jawaban referensi/*ground truth* adalah teks jawaban acuan yang dirumuskan dari website resmi prodi teknik informatika untuk setiap pertanyaan (disusun ringkas,

faktual, dan merefleksikan isi sumber). Sedangkan jawaban sistem adalah keluaran generatif chatbot RAG untuk pertanyaan yang sama.

Metrik ini beroperasi di tingkat kalimat penuh untuk menilai kesamaan semantik global. *Semantic Similarity* digunakan untuk mengukur kesamaan makna umum antar jawaban sistem dan referensi.

$$SS = \cos(V_{\text{jawaban}}, V_{\text{referensi}}) \quad (3.20)$$

Dengan *semantic similarity*, sistem tetap dianggap benar walaupun gaya bahasanya berbeda, selama makna intinya sama. Semakin tinggi nilai dari *Semantic Similarity* maka makna keluaran (*output*) semakin mendekati jawaban acuan yang bersumber dari website resmi.

Pada Tabel 3.11 ini adalah ringkasan skenario pengujian yang akan dilakukan untuk menilai dan menguji masing-masing tahap dari tahap *retrieval* dan tahap *generation* dari sistem chatbot berbasis RAG ini:

Tabel 3.11 Ringkasan Skenario Pengujian RAG

Aspek Uji	Metrik	Tujuan	Data Uji	Parameter
Retrieval	Recall@k, MRR	Mengukur relevansi pencarian	30 pertanyaan	k = 3,5,10
Generation	Semantic Similarity, AR, Faithfulness	Menilai akurasi & faktualitas jawaban	30 pertanyaan	—
Generation	Semantic Similarity, AR, Faithfulness	Menilai konsistensi jawaban	1 pertanyaan (sama)	5x percobaan

Terdapat tambahan pengujian tahap generation yaitu dilakukan pengujian konsistensi jawaban dari sistem dengan mengujikan satu *query* yang sama namun dengan jumlah percobaan lebih dari satu kali yaitu sebanyak lima kalil. Hal ini bertujuan untuk melihat bagaimana nilai metrik dari *semantic similarity, answer*

relevancy, dan juga faithfulness jika *query* yang diuji coba terhadap sistem adalah *query* yang sama.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil

Sistem chatbot akademik berbasis RAG dikembangkan untuk membantu mahasiswa mengakses informasi akademik dari Program Studi Teknik Informatika UIN Malang secara cepat dan akurat.

4.1.1 Arsitektur & Antarmuka Klien (*Front-end*)

Arsitektur sistem untuk antarmuka chabot terdiri dari HTML, CSS, dan Javascript. Aplikasi ini bersifat statis dan murni (tanpa *framework*) sehingga mudah dipelihara dan ringan dijalankan di perangkat mahasiswa. Aplikasi ini memanggil REST API dari layanan RAG (*FastAPI*) menggunakan *fetch().* Penjelasan dari masing-masing *file* beserta fungsi nya dapat dilihat pada Tabel 4.1.

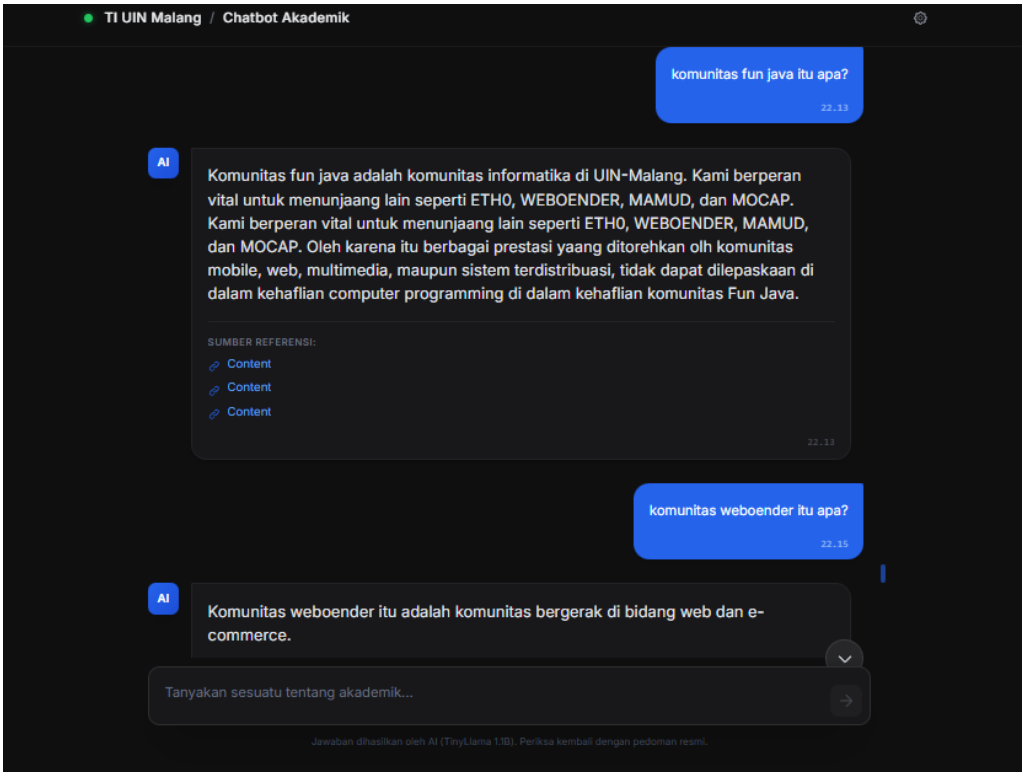
Tabel 4.1 Struktur berkas (*root front-end*)

Berkas	Fungsi singkat
index.html	Halaman utama (container chat, header judul, area riwayat percakapan)
styles.css	Gaya visual (tema gelap, komponen kartu sumber, badge status)
app.js	Logika klien: panggil /api/retrieve & /api/chat, render jawaban + sumber
README.md	Petunjuk singkat menjalankan front-end di lokal

Cara menjalankan (lokal):

- *Serve* sederhana via *extension*/*Live Server* atau PHP *built-in server* di *VS Code*.
- Akses: http://localhost:3000/index.html

Tampilan dari antarmuka chatbot dapat dilihat pada Gambar 4.1 di bawah.



Gambar 4.1 Antarmuka Chatbot Akademik (mode gelap)

4.1.2 Arsitektur & Organisasi Kode RAG (*Back-end*)

Di sisi *back-end,* arsitektur sistemnya lebih kompleks dan penjelasan dari masing-masing *file* juga dapat dilihat pada Tabel 4.2. Pada tabel tersebut dijelaskan dari isi dan fungsi dari setiap direktori apa saja. Sistem *back-end* ini dikembangkan dengan *FastAPI* (Python), dengan memisahkan lapisan API dan modul RAG. Desain ini memudahkan pengujian unit serta penggantian komponen (*retriever*/LLM) tanpa mengubah antarmuka.

Tabel 4.2 Struktur Proyek RAG (*back-end*)

Direktori/Berkas	Isi / Peran
api/main.py	Entry FastAPI, registrasi router, middleware CORS
api/routes/	Endpoint publik (/api/retrieve, /api/chat)
api/deps.py, api/schemas.py	Dependency injection & skema Pydantic request/response
rag/scraping/	Skrip scraping (seed, fetch, simpan JSON mentah)
rag/preprocess/	clean_html.py, clean_blocks.py (cleaning & normalisasi)
rag/retrieve/	search.py (BM25), util re-ranking; retrieve.py (wrapper retriever)

Tabel 4.2 Lanjutan

Direktori/Berkas	Isi / Peran
rag/index/	Pembuatan embedding (E5-Small/Multilingual-E5) & indeks FAISS
rag/generate/	llm_client.py (TinyLlama GGUF + prompt), chat.py (alur RAG → jawab)
data/raw/	Hasil scraping mentah (JSON)
data/cleaned/	Hasil cleaning/normalisasi (blocks_filtered.jsonl)
data/chunks/	Hasil chunking (chunks.jsonl, chunks_meta.json)
data/index/	Berkas embedding & indeks FAISS
data/eval/	(Opsional) log & sampel pengujian
models/	Model lokal tinyllama-1.1b-chat-v1.0.Q4_K_M.gguf
logs/	Log proses (scrape/embedding/retrieve/generate)

4.1.3 Hasil *Web Scraping* dan Pembentukan Dataset

Pada tahap ini, proses pengumpulan data dilakukan melalui teknik web scraping terhadap website resmi Jurusan Teknik Informatika UIN Maulana Malik Ibrahim Malang sebagai sumber utama korpus pengetahuan untuk sistem chatbot berbasis RAG. Proses *scraping* menggunakan modul Python *requests* dan *BeautifulSoup* yang telah diimplementasikan pada skrip *scrape_ti.py*. Daftar URL awal (*entry points*) dimuat melalui berkas *sitemap.txt* yang berisi delapan kategori utama, yaitu: profil, akademik, beasiswa, kategori/berita, kegiatan, penelitian, PMB, dan kontak. *Scraper* kemudian mengekstraksi seluruh konten dari halaman-halaman tersebut, dan jika terdapat tautan internal yang masih berada pada domain yang sama, *scraper* tetap memprosesnya sebagai bagian dari mekanisme kelanjutan scraping, bukan *crawling* mandiri, sehingga teknik yang digunakan masih dalam ruang lingkup scraping multi-halaman sesuai metodologi penelitian.

Pengumpulan data dilakukan menggunakan skrip python yang terdapat pada Tabel 4.3 berikut. Dengan menjalankan kode berikut pada terminal, maka berkas *scrape_ti.py* yang bertugas untuk melakukan proses *scraping* akan dijalankan.

Tabel 4.3 Kode Python Untuk Menjalanan Scraping

Kode
`python rag/scraping/scrape_ti.py --depth 1 --max_pages 350 --delay 0.6`

Parameter ini membuat *scraper* mengeksplorasi halaman awal website kemudian mengambil seluruh konten relevan sampai batas maksimum menjadi 350 bagian file, di mana satu halaman website dapat menghasilkan banyak file JSON karena dipisahkan berdasarkan entitas, misalnya 1 dosen menjadi 1 file JSON sendiri, 1 mata kuliah juga 1 file JSON terpisah. Dari batas maksimal yang sudah ditentukan yaitu 350, berhasil didapatkan file (.json) sebanyak 250 yang diambil kontennya, sementara sisanya gagal dan juga tidak dilakukan scraping dengan beberapa kondisi seperti *SSL certificate error*, struktur HTML tidak stabil, halaman tidak ditemukan, dan juga halaman web yang tingkat informasinya rendah sehingga akan di-*skip* oleh kode saat dilakukan *scraping*. Beberapa hasil scraping juga mengandung teks yang tidak relevan, *markup* anomali, atau duplikasi paragraf, sehingga dilakukan pembersihan manual pada tahap pascaproses. Dan ringkasan data JSON yang diperoleh dari tahap *scraping* dapat dilihat pada Tabel 4.4 berikut.

Tabel 4.4 Total Data Terkumpul Sementara

Keterangan	Total File JSON
Batas Jumlah File	350
File JSON yang berhasil didapatkan	250
Jumlah Total	250

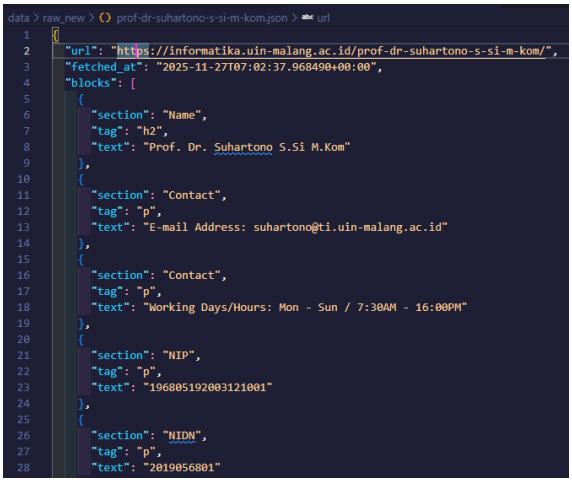
Seluruh hasil *scraping* kemudian disimpan dalam format JSON terstruktur dengan tiga komponen utama: *url, fetched_at,* dan daftar *blocks* berisi pasangan *tag–text*. Dataset ini menjadi fondasi untuk tahap selanjutnya berupa *preprocessing*,

cleaning, *normalisasi*, serta *chunking* sebelum masuk ke proses *embedding* dan *indexing* pada pipeline RAG. Pada Tabel 4.5 ini disajikan contoh rincian dari data JSON terstruktur hasil *scraping*:

Tabel 4.5 Contoh Data Dosen Hasil *Scraping*

Keterangan	Deskripsi
Nama	Prof. Dr. Suhartono, S.Si, M.Kom
Bidang Keahlian	Intelligent System
Email	suhartono@ti.uin-malang.ac.id
Research Fields	Pattern Recognition, Neural Networks, Intelligent Multimedia Networking
Pendidikan	S1 Matematika, ITS S2 Informatika, ITS S3 Jaringan Cerdas Multimedia, ITS
Sinta ID	5975073
Scopus ID	56465811900
ORCID ID	0000-0002-4304-4279

Pada Gambar 4.2 disajikan gambar tangkapan layar dari aplikasi *Visual Studio Code* yang menampilkan cuplikan data JSON asli hasil dari tahap *scraping*.



Gambar 4.2 Contoh Data Mentah Dosen Hasil *Scraping*

Data profil seperti pada Gambar 4.2 dan Tabel 4.5 tersebut akan dijadikan dasar menjawab pertanyaan seperti "Siapa dosen dengan fokus bidang *Intelligent System* di TI UIN Malang?" kemudian juga pertanyaan seperti "Apa penelitian Prof.

Suhartono?", namun akan melalui proses yang lebih panjang lagi yaitu seperti proses pembuatan *chunking text* hingga *embedding* terlebih dahulu.

4.1.4 Hasil *Preprocessing*

Tahap *preprocessing* merupakan langkah penting untuk menyiapkan data hasil web scraping menjadi korpus terstruktur yang siap diproses pada tahap *embedding* dan *indexing*. Proses ini mencakup dua komponen utama, yaitu pembersihan dokumen (*cleaning*) dan pembentukan potongan teks (*chunking*). Seluruh tahapan dilakukan secara otomatis menggunakan dua skrip utama, yaitu *clean_html.py* dan *chunker.py*. Berikut adalah penjelasan rinci berdasarkan data nyata hasil scraping.

1. Hasil Pembersihan Teks (*Cleaning*)

Pembersihan dilakukan terhadap seluruh file JSON hasil scraping yang berada di direktori *data/raw_new/*. Berdasarkan *output clean_html.py*, sistem berhasil memproses 250 file JSON mentah, menghasilkan 2.363 blok teks bersih yang lolos proses ini, kemudian output disimpan pada *data/cleaned/blocks_filtered.jsonl*. Proses *cleaning* dengan skrip tersebut meliputi:

a. Penghapusan tag HTML secara menyeluruh

Semua elemen , , <h1>–<h4>, <li>, dan atribut HTML dibersihkan menjadi teks polos. Ilustrasi dari teks yang sebelum dan sesudah dilakukan penghapusan tag HTML dapat dilihat pada Tabel 4.6 berikut.

Tabel 4.6 Contoh Hasil Penghapusan Tag HTML

Tahap	Contoh
Sebelum Cleaning	" The realization of an integrative Informatics Engineering Department... <script>alert()</script> "

Tabel 4.6 Lanjutan

Tahap	Contoh
Sesudah Cleaning	"The realization of an integrative Informatics Engineering Department in integrating science and Islam with an international reputation."

b. Normalisasi *whitespace*

Pada Tabel 4.7 berikut disajikan ilustrasi dari normalisasi *whitespace,* dimana normalisasi ini akan menghilangkan spasi ganda, *newline* berulang, dan karakter *formatting*.

Tabel 4.7 Contoh Hasil Normalisasi *whitespace*

Tahap	Contoh
Sebelum Cleaning	"The realization\n\n of an integrative Informatics Engineering Department ..."
Sesudah Cleaning	"The realization of an integrative Informatics Engineering Department in integrating science and Islam with an international reputation."

c. Penghapusan karakter non-teks / *noise*

Proses ini menghilangkan (sisa *script* (<script>…), potongan struktur gambar, artefak CSS/JS, string "JFIF" (penanda file gambar), dan karakter yang bukan huruf/angka).

Tabel 4.8 Contoh Hasil Penghapusan Karakter *Noise*

Tahap	Contoh
Sebelum Cleaning	"JFIF....\x00\x1f\x00 *Teaching Laboratory* ..."
Sesudah Cleaning	"*Teaching Laboratory* digunakan untuk kegiatan praktikum pemrograman dan asistensi mahasiswa." *(contoh real dari blok lain dalam blocks_filtered.jsonl)*

d. *Filtering* blok tanpa konten

Proses *cleaning* juga menghapus halaman dengan blok kosong atau berisi karakter tidak bermakna sehingga bagian seperti itu tidak ikut dimasukkan

ke blok *final* (text == "" → dilewati). Ilustrasi dari hasil proses *filtering* blok dapat dilihat pada Tabel 4.9.

Tabel 4.9 Contoh Hasil *Filtering* Blok

Tahap	Contoh
Sebelum Cleaning	"" atau " " atau " "
Sesudah Cleaning	Tidak dimasukkan ke blocks_filtered.jsonl

Jika *text.strip() == ""*, blok tersebut langsung dilewati. Itulah sebabnya dari 350 file hanya tersisa 2350 blok yang dianggap valid.

2. Hasil Normalisasi dan Standarisasi Informasi

Setelah proses cleaning menghasilkan 2.363 blok teks bersih, seluruh blok diproses kembali pada tahap normalisasi untuk memastikan keseragaman format, penulisan, dan struktur linguistik sebelum masuk ke embedding. Tahap ini dilakukan otomatis melalui fungsi-fungsi dalam clean_html.py serta pemrosesan tambahan pada chunker.py. Normalisasi pada penelitian ini terbagi ke dalam empat kategori, yaitu normalisasi karakter, normalisasi token, normalisasi entitas, dan normalisasi struktur.

Tabel 4.10 Aturan Normalisasi per Level

Level	Aturan	Sebelum	Sesudah	Dampak ke Sistem
Karakter	Unicode normalization (NFKC), hilangkan karakter tak terlihat (NBSP, ZWSP), normalisasi dash	KRS — Online, Visi\u200b Prodi	KRS — Online, Visi Prodi	Menghindari token “phantom” yang menurunkan skor BM25 dan kualitas embedding
Karakter	Normalisasi spasi & baris (≤1 spasi, ≤2 baris)	Laboratoriu m Informatika	Laboratori um Informatika	Menstabilkan tokenisasi

Tabel 4.10 Lanjutan

Level	Aturan	Sebelum	Sesudah	Dampak ke Sistem
Token	Case folding selektif (kecuali nama orang/instansi/akronim)	PROGRAM STUDI, Fakultas SAINS	Program Studi, Fakultas Sains	Menjaga proper-noun tetap kapital seperlunya
Token	Angka & satuan (format konsisten)	14 sMt, 12 SKS	14 semester, 12 SKS	Angka terbaca konsisten; mengurangi varian token
Token	Tanggal (standar teks Indonesia) + metadata ISO (di meta)	28 Agustus–15 Des 2023	28 Agustus–15 Desember 2023 (meta: 2023-08-28..2023-12-15)	Teks tetap natural; metadata siap dipakai filter
Token	Tanda baca (hilangkan ganda, seragamkan kutip & bullet)	“visi prodi’’, - -	“visi prodi”, —	Mengurangi noise embedding
Entitas	Nama dosen/role diseragamkan	Prof Dr Suhartono SSi MKom	Prof. Dr. Suhartono, S.Si., M.Kom.	Pencarian nama menjadi presisi
Entitas	Unit organisasi standar	Informatics Engineering, Informatika	Teknik Informatika (TI)	Menyatukan istilah lintas bahasa
Struktur	Penyatuan kalimat yang patah/tercerai	Teaching Lab. Digunakan untuk... Praktikum...	Teaching Laboratory digunakan untuk kegiatan praktikum dan asistensi.	Kalimat utuh menghasilkan *embedding* lebih bermakna

Catatan: nama diri (dosen, unit, lab) tidak di-*lowercase*. Normalisasi angka & tanggal tidak mengubah isi, hanya menstandarkan penulisan; representasi ISO disimpan di metadata untuk kebutuhan analitik tambahan (opsional). Contoh Normalisasi Nyata (Sebelum & Sesudah) dapat dilihat pada Tabel 4.11.

Tabel 4.11 Contoh Normalisasi Nyata (Sebelum & Sesudah)

Sumber/ Kategori	**Kutipan Sebelum**	**Sesudah Normalisasi**
Profil Dosen	Prof Dr Suhartono SSi MKom	Prof. Dr. Suhartono, S.Si., M.Kom.
Regulasi Akademik	Batas masa studi maks. 14 sMt	Batas masa studi maksimal 14 semester.
Jadwal Perkuliahan	28 Agustus–15 Des 2023	28 Agustus–15 Desember 2023 (meta: 2023-08-28..2023-12-15)
Istilah Prodi (EN→ID)	Informatics Engineering Undergraduate	Program Sarjana (S1) Teknik Informatika (TI)

Normalisasi memastikan korpus seragam, bersih, dan siap di-*embedding*, tanpa mengubah makna ilmiah dokumen. Dampaknya: BM25 lebih stabil, *cosine similarity* lebih bermakna, dan *faithfulness generation* meningkat karena konteks lebih jelas dan konsisten.

3. Hasil *Chunking*

Tahap ini dilakukan menggunakan skrip *chunker.py* untuk membagi teks hasil *cleaning* yang telah dinormalisasi menjadi unit-unit kecil (*chunks*) yang siap diproses menjadi *embedding*. *Chunking* dilakukan dengan pendekatan sliding *window*, yaitu memotong teks berdasarkan jumlah token tertentu dengan *overlap* untuk menjaga kontinuitas informasi. Tabel 4.12 berikut merupakan konfigurasi *chunking* berdasarkan *output* sistem.

Tabel 4.12 Statistik Hasil Proses *Chunking*

Parameter	**Nilai**
Target panjang chunk	150 token

Tabel 4.12 Lanjutan

Parameter	Nilai
Jumlah overlap	40 token
Minimal token chunk	3 token
Maksimal token chunk	227 token
Rata-rata panjang	101.22 token
Total halaman (page_id)**	244
Total chunk final	384 chunks

Kemudian Tabel 4.13 berikut disajikan salah satu contoh *chunk* asli yang dihasilkan dari proses chunking pada halaman *undergraduate-s1*.

Tabel 4.13 Contoh Metadata Hasil *Chunking*

Field	Nilai
id	undergraduate-s1-0003
page_id	undergraduate-s1
url	https://informatika.uin-malang.ac.id/undergraduate-s1/
sections	["Current Management", "History", "Milestone of Informatics Engineering Department", "Mission", "Objectives", "Past Management", "Strategy", "Vision"]
token_count	150
start_token	330
end_token	480

Salah satu detail isi dari *chunking-text* dapat dilihat pada Tabel 4.14 berikut.

Tabel 4.14 Contoh Hasil *Chunking*

Isi *Chunk*
on and Islam with an international reputation.\n1.Producing Informatics Engineering graduates with ulul albab character. 2.Producing science and information technology that is relevant and highly competitive.\n1.Providing wider access to Informatics Engineering education to the community. 2.Providing Informatics Engineering graduates to meet community needs.\nOrganizing the tridharma of higher education in Informatics Engineering in a quality integrative manner\n2004 : Jurusan Teknik Informatika started to operate based on a Decree of the Ministry of National Education No 05/MPN/HK/1004 dated on 23 January 2004 and a Decree of Directorate General of Islamic Department No DJ.II/54/2005 dated on 28 March 2005.\n2007 : first accreditation of BAN-PT to reach C level (276) based on BAN-PT letter

Tabel 4.14 Lanjutan

No. 019/BAN-PT/Ak-X/S1/VIII/2007.\n2014 : second accreditation of BAN-PT to reach B level (310) based on BAN-PT letter No. 057/SK/BAN-PT/Akred/S/II/2014.\n2019 : third accreditation of BAN-PT to
...

Proses *chunking* menggabungkan seluruh blok teks berdasarkan *page_id*, kemudian memotongnya menjadi segmen-segmen berukuran 150 token. Untuk menjaga konteks antar-paragraf, *chunk* dibuat dengan *overlap* 50 token, sehingga informasi penting tidak terpotong secara *abrupt*.

4.1.5 Hasil *Embedding* dan *Indexing*

Setelah seluruh teks hasil *chunking* siap, tahap berikutnya adalah mengubah setiap *chunk* menjadi vektor numerik menggunakan *Sentence Transformer* dan kemudian membangun indeks vektor menggunakan FAISS. Pada penelitian ini proses tersebut diotomatisasi melalui skrip *embed_index.py*, dengan input utama *chunks.jsonl* dan *output* berupa berkas *faiss.index* dan *chunks_meta.json* sebagai metadata.

Model *embedding* yang digunakan adalah *sentence transformer multilingual-e5-base*, dengan alasan diantaranya efisien untuk data berbahasa Indonesia, ukuran ringan sehingga cocok untuk *pipeline* RAG lokal, akurat pada *retrieval* berbasis *semantic search*.

Setiap *chunk* dikonversi menjadi vektor berdimensi 768 dan disimpan dalam *array embedding serialized.* Setelah *embedding* selesai, seluruh vektor di-index menggunakan FAISS *IndexFlatIP*, yaitu jenis indeks yang menggunakan *inner*

product similarity dan cocok untuk *semantic search* berskala ratusan hingga ribuan dokumen.

a. Konfigurasi *Embedding*

Embedding dibangun dengan fungsi *build_embeddings()* pada skrip *embed_index.py*. Pada Tabel 4.15 berikut dapat dilihat detail dari konfigurasi *embedding* yang akan dilakukan pada dokumen JSON hasil proses *chunking.*

Tabel 4.15 Konfigurasi *Embedding*

Parameter	Nilai / Keterangan
Model embedding	intfloat/multilingual-e5-base (SentenceTransformer)
Dimensi vektor	768 (sesuai arsitektur E5-base)
Jumlah chunk	384 *chunk*
Normalisasi vektor	L2-*normalized* (*normalize_embeddings*=True)
Tipe data vektor	float32
Ukuran batch	64 (default pada argumen skrip)

Jumlah *chunk* 384 dan nama model ini juga tercatat eksplisit pada berkas metadata *chunks_meta.json*.

b. Struktur Metadata *Embedding*

Setelah proses *embedding* selesai dihitung, sistem tidak hanya menyimpan indeks FAISS, tetapi juga menyimpan metadata *mapping* antara *vector_id* dan chunk aslinya pada berkas *chunks_meta.json*. Struktur metadata ini dihasilkan di bagian akhir fungsi *main()* pada *embed_index.py*.

Struktur global *chunks_meta.json*:

- model: nama model *embedding* yang digunakan ("*intfloat/multilingual-e5-base*").
- *num_chunks*: jumlah *chunk* yang di-*embedding* (384).
- *index_path*: lokasi berkas indeks FAISS di disk.

- *chunks_meta*: *array* berisi objek metadata untuk setiap vektor.

Tabel 4.16 berikut menyajikan contoh *record* nyata dari isi *chunks_meta.json*. Data berikut hanya sebagai statistik untuk melihat data secara umum (*general*).

Tabel 4.16 Contoh Metadata *Chunk* dari chunks_meta.json

Field	**Nilai (Contoh Real)**
vector_id	25
chunk_id	beasiswa-0000
page_id	beasiswa
url	https://informatika.uin-malang.ac.id/beasiswa
sections	["Content"]
token_count	150
start_token	0
end_token	150

Record pada Tabel 4.16 menunjukkan bahwa vektor dengan *vector_id* = 22 merepresentasikan chunk pertama dari halaman beasiswa, *chunk* tersebut mencakup 150 token pertama dari konten halaman, dan informasi ini yang nantinya dikembalikan *retriever* ketika FAISS menemukan vektor paling relevan.

c. Pembangunan Indeks FAISS

Indeks vektor dibangun menggunakan fungsi *build_faiss_index(embs: np.ndarray)* pada *embed_index.py*:

1. Tipe indeks: *faiss.IndexFlatIP* (*inner product*).
2. Input: matriks *embedding* berukuran (num_chunks, 768) yang sudah *L2-normalized*.

3. Proses: pertama proses pengambilan dimensi vektor dari *embs.shape[1], kemudian* dilakukan inisialisasi *IndexFlatIP(dim)*, dan terakhir menambahkan seluruh vektor ke indeks dengan *index.add(embs).*

Pemilihan *IndexFlatIP* + vektor yang sudah dinormalisasi ini sesuai praktik standar untuk implementasi *cosine similarity* pada FAISS, karena:

$$\cos(u, v) = \frac{u \cdot v}{|u|\,|v|} \quad \text{dan jika} \quad |u| = |v| = 1 \Rightarrow \cos(u, v) = u \cdot v.$$

Indeks kemudian disimpan ke berkas sesuai *INDEX_PATH* (*default data/index/faiss.index*) menggunakan *faiss.write_index(index*, *str(INDEX_PATH))*.

d. Ringkasan Hasil *Embedding*

Konfigurasi dari proses *embedding* beserta *vector indexing* dapat diperinci seperti pada Tabel 4.17 berikut. Dapat dilihat bahwa proses *vector indexing* menggunakan IndexFlatIP dari FAISS (*Facebook AI Similarity Search*).

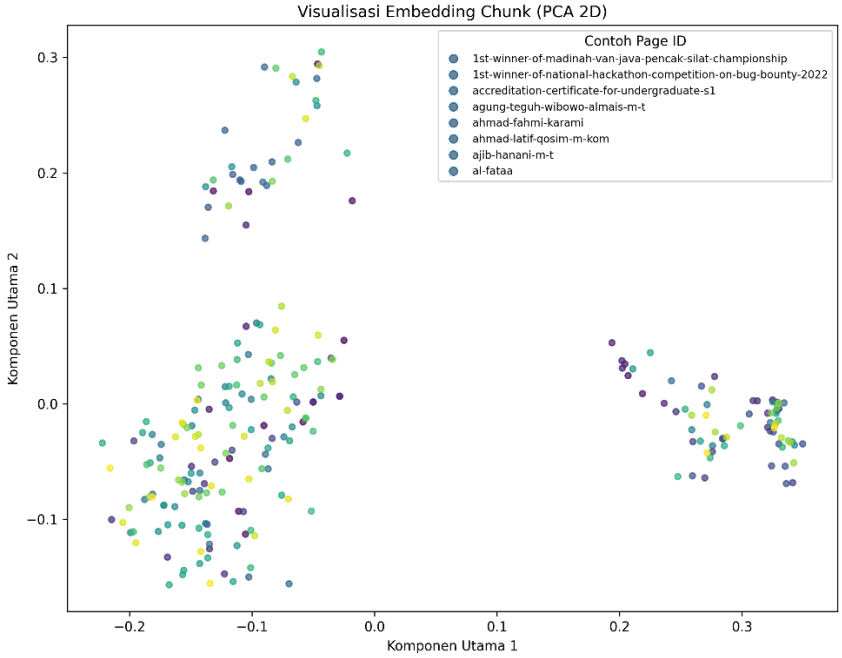
Tabel 4.17 Ringkasan Hasil *Embedding*

Komponen	**Hasil**
Sumber teks	384 *chunk* pada chunks.jsonl
Model embedding	intfloat/multilingual-e5-base
Dimensi vektor	768
Normalisasi	L2 (cosine via inner product)
Jenis indeks	faiss.IndexFlatIP
Lokasi indeks	data/index/faiss-2.index
Metadata	data/index/chunks_meta-2.json

Dengan tahapan ini, seluruh pengetahuan dari website Prodi TI UIN Malang telah terpetakan sebagai ruang vektor berdimensi 768, yang dapat di-*query*

secara efisien oleh modul *retrieval* (*search.py / retrieve.py*) ketika chatbot menerima pertanyaan dari pengguna.

Gambar 4.3 berikut ini adalah visualisasi vektor *embedding* namun disajikan dalam dimensi yang rendah pada 2 dimensi untuk melihat bagaimana persebaran informasi korpus data dalam nilai vektor.

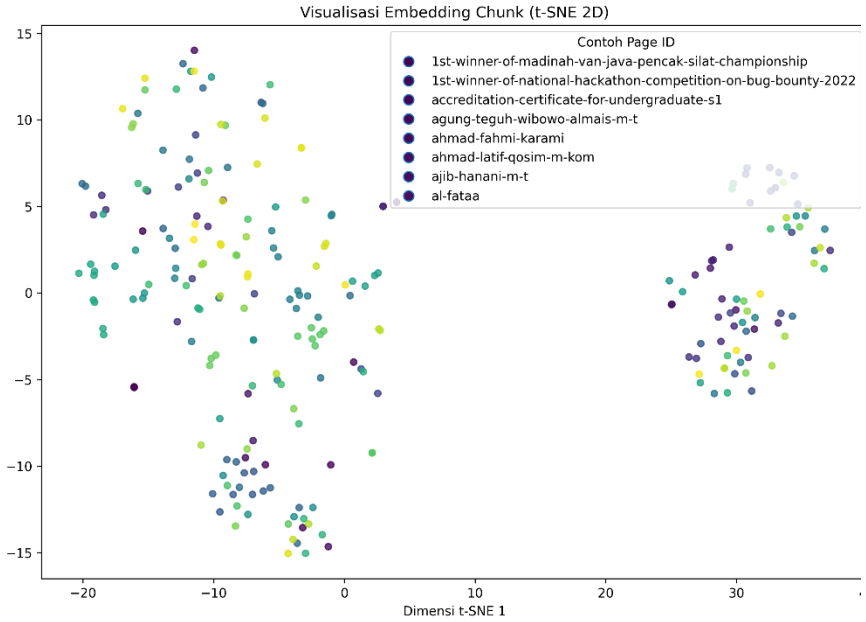


Gambar 4.3 Sebaran *Embedding* (PCA 2D) – kluster kasar per tema halaman

Proyeksi *embedding* 768-dim ke 2 dimensi dengan PCA untuk meninjau pola global secara cepat. Terlihat pengelompokan kasar antar tema halaman (mis. profil dosen, kegiatan, beasiswa), tetapi sebagian klaster masih tumpang tindih karena PCA bersifat linier sehingga pemisahan semantik halus belum sepenuhnya terbaca.

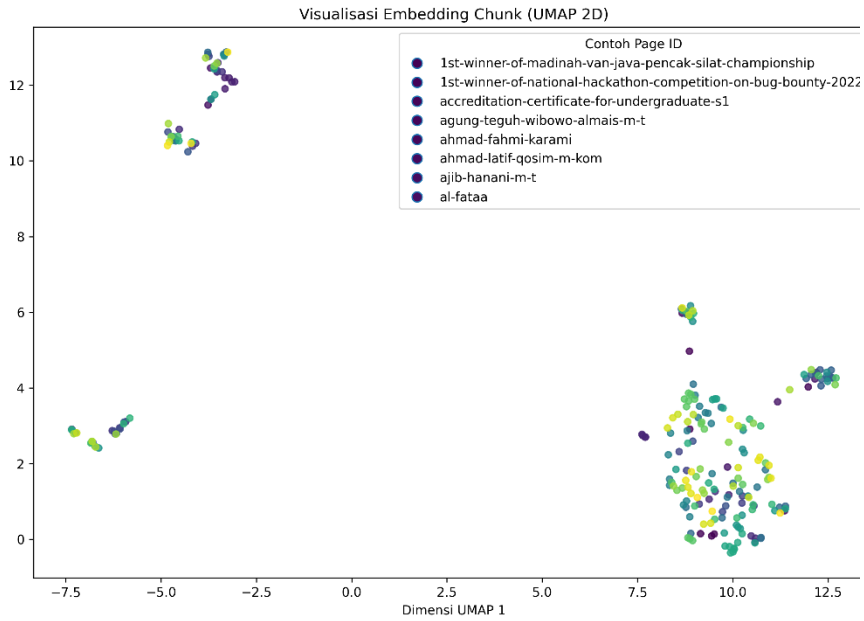
Sementara, pada Gambar 4.4 berikut terdapat gambaran sebaran *Embedding* (t-SNE 2D), terlihat bahwa visualisasi t-SNE menonjolkan kedekatan lokal: chunk yang semantis mirip membentuk gugus kecil yang lebih terpisah

dibanding PCA. Struktur global tidak selalu dapat diinterpretasi, namun sub-topik (mis. beasiswa vs. berita kegiatan) tampak lebih jelas sehingga memudahkan inspeksi *outlier/noise*.



Gambar 4.4 Sebaran *Embedding* (t-SNE 2D) – pemisahan non-linier antar kelompok

Kemudian berdasarkan Gambar 4.5 berikut, proyeksi UMAP terlihat mampu mempertahankan keseimbangan antara struktur lokal dan global data dengan baik. Klaster yang terbentuk tampak padat dan terpolakan, yang menandakan bahwa chunk dengan topik serupa telah terelompokkan secara efektif. Hal ini mengonfirmasi bahwa representasi vektor E5 (768 dimensi) sudah memiliki struktur semantik yang matang untuk diterapkan pada retrieval berbasis FAISS *IndexFlatIP*.



Gambar 4.5 Sebaran *Embedding* (UMAP 2D) – struktur manifold yang lebih terjaga

4.1.6 Contoh Hasil *Retrieval*

Pengujian modul *retrieval* dilakukan untuk mengukur sejauh mana kombinasi FAISS + BM25 mampu menemukan dokumen yang relevan terhadap pertanyaan pengguna. Dataset uji disusun dalam bentuk berkas *retrieval_eval_set.jsonl* yang berisi sekumpulan pasangan *query* dan daftar *ground truth* URL dokumen yang dianggap relevan berdasarkan penilaian peneliti (misalnya halaman profil komunitas, kurikulum, maupun pengumuman resmi di laman Teknik Informatika UIN Maulana Malik Ibrahim Malang).

Untuk setiap *query*, sistem menjalankan fungsi *retrieve_as_dicts()* yang menggabungkan pencarian vektor menggunakan FAISS (berbasis *embedding* intfloat/multilingual-e5-base) dan pengurutan ulang menggunakan BM25 sebagai *reranker*. Hasil pengambilan dokumen kemudian dievaluasi menggunakan metrik

Mean Reciprocal Rank (MRR@k) dan *Recall*@k, dengan variasi nilai k yang mengacu pada rancangan skenario pengujian di bab tiga (k = 3 dan k = 5).

Tabel 4.18 Contoh Hasil *Retrieval*

k	MRR@k	Recall@k
3	0,41	0,76
5	0,42	0,96

Hasil pengujian ditunjukkan pada Tabel 4.18. Pada nilai k = 3, sistem menghasilkan MRR@3 sebesar kurang lebih 0,41 dan Recall@3 sebesar kurang lebih 0,76. Nilai Recall@3 ini dapat diartikan bahwa sekitar 76% query uji berhasil menemukan minimal satu dokumen relevan di tiga peringkat teratas hasil pencarian. Sementara itu, nilai MRR@3 sekitar 0,41 menunjukkan bahwa secara rata-rata, dokumen relevan pertama berada di sekitar peringkat 2–3, sehingga tidak selalu muncul tepat di peringkat pertama namun masih relatif dekat dengan posisi teratas.

Ketika nilai k diperbesar menjadi k = 5, kinerja sistem meningkat terutama dari sisi cakupan. Nilai *Recall*@5 naik menjadi sekitar 0,96, yang berarti hampir seluruh *query* uji (sekitar 96%) berhasil menemukan setidaknya satu dokumen relevan di antara lima hasil teratas. Nilai MRR@5 juga sedikit meningkat menjadi sekitar 0,42, mengindikasikan bahwa penambahan jumlah hasil yang diamati tidak menurunkan posisi relatif dokumen relevan pertama secara signifikan. Dengan kata lain, walaupun tidak semua dokumen relevan selalu muncul di posisi pertama, sistem secara konsisten mampu menempatkan dokumen yang benar dalam kisaran peringkat 1–5.

Secara umum, hasil pengujian menunjukkan bahwa modul *retrieval* sudah cukup efektif dalam menemukan dokumen yang relevan, terutama ketika

mempertimbangkan lima hasil teratas (k = 5). Namun demikian, nilai MRR yang masih berada di kisaran 0,41–0,42 menandakan bahwa dokumen relevan pertama belum selalu konsisten muncul di peringkat teratas. Hal ini dapat disebabkan oleh beberapa faktor, antara lain variasi gaya penulisan pada dokumen sumber, keterbatasan model *embedding* yang digunakan, serta adanya beberapa *query* yang bersifat ambigu atau terlalu umum. Temuan ini menjadi dasar bahwa pada pengembangan selanjutnya, modul *retrieval* masih dapat ditingkatkan, misalnya dengan penyetelan ulang weight BM25, penambahan sinonim/ekspansi *query*, atau penggunaan model *embedding* yang lebih kuat untuk domain bahasa Indonesia.

Tabel 4.19 Endpoint Pengujian *Retrieval*

Keterangan	Deskripsi
Metode Request POST	http://127.0.0.1:8000/api/retrieve

Body	{
	"query": "<pertanyaan>",
	"top_k": 5
	}

Untuk mengukur performa awal sistem, dilakukan pengujian terhadap beberapa contoh pertanyaan akademik. Salah satu contoh uji penting adalah kueri "jadwal perkuliahan teknik informatika". Percobaan ini dipilih karena mewakili tipe pertanyaan yang umum diajukan mahasiswa dan menguji kemampuan sistem dalam menemukan informasi penting yang tersebar pada berbagai halaman.

4.1.6.1 Contoh Hasil *Retrieve* (Studi Kasus)

Sebelum masuk ke tahap pengujian final *retrieval,* dilakukan uji coba *retrieval* dengan *query* tertentu untuk melihat bagaimana hasil detail dari tahap *retrieval*. Kueri yang digunakan yaitu "Apa itu komunitas Fun Java?". Adapun

parameter k yang digunakan untuk pengambilan top-k *chunking text* adalah top-k = 5. Pada Tabel 4.20 berikut disajikan ringkasan hasil *retrieve* yang dikembalikan oleh *endpoint* berdasarkan *response* JSON pada *software* Postman.

Tabel 4.20 Hasil *Retrieve* untuk Kueri "jadwal perkuliahan teknik informatika

Rank	Score	Source	Snippet Singkat
1	5.11	https://informatika.uin-malang.ac.id/fun-java/	"Komunitas ini berfokus pada Object-Oriented Programming… menjadi dasar pembinaan bagi komunitas lain seperti ETH0, WEBOENDER…"
2	3.71	https://informatika.uin-malang.ac.id/mocap/	"Adalah komunitas Mobile Programming… "
3	3.55	https://informatika.uin-malang.ac.id/rapat-kerja-pengurus-hmj-ti-2019/	"AL-FATAA, ETH0, FUN JAVA, MAMUD, MOCAP hadir dalam rapat kerja…"
4	2.56	https://informatika.uin-malang.ac.id/it-incubation-2019-helps-students-finding-passion/	"Kegiatan IT Incubation… bekerja sama dengan komunitas Fun Java, Ontaki, Mamud…"
5	2.43	https://informatika.uin-malang.ac.id/informatics-sport-community-isc-launching/	"AL-FATAA, ETH0, FUN JAVA, MAMUD hadir pada kegiatan launching ISC…"

4.1.6.2 Analisis Hasil *Retrieve*

Pada contoh *query* "Apa itu komunitas Fun Java?", sistem berhasil menempatkan halaman resmi komunitas Fun Java pada peringkat pertama dengan skor relevansi tertinggi (5.11). Empat dokumen berikutnya juga masih berkaitan karena menyebutkan Fun Java dalam konteks kegiatan atau kolaborasi komunitas lain. Hal ini menunjukkan bahwa modul *retrieval* mampu mengidentifikasi dokumen yang benar-benar relevan dan memprioritaskannya dalam hasil pencarian.

4.1.7 Contoh Hasil *Generation*

Tahap *generation* bertujuan mengevaluasi kemampuan model bahasa (LLM) dalam membangkitkan jawaban berbasis konteks dokumen hasil *retrieval*. Pada penelitian ini, proses generasi jawaban diimplementasikan menggunakan model TinyLlama 1.1B Chat dengan format kuantisasi Q4_K_M, yang dijalankan secara lokal melalui pustaka *llama_cpp* pada laptop pengujian. Pada Tabel 4.21 berikut disajikan konfigurasi detail dari inisialisasi LLM yang diatur pada berkas *llm_client.py*, dengan parameter utama.

Tabel 4.21 Konfigurasi Model LLM untuk Tahap *Generation*

Parameter	Nilai
Model	tinyllama-1.1b-chat-v1.0.Q4_K_M.gguf
Dimensi parameter	±1.1B parameter (kuantisasi 4-bit)
Path model default	models/tinyllama-1.1b-chat-v1.0.Q4_K_M.gguf
Panjang konteks (LLM_CTX)	2.048 token
Jumlah thread (LLM_THREADS)	4 thread (menyesuaikan prosesor Ryzen 3)
Maksimal token jawaban (LLM_MAX_TOKENS)	192 token (default)
LLM_TEMPERATURE	0,25 (jawaban lebih deterministik)

Inisialisasi model dilakukan satu kali dengan pola *singleton* melalui fungsi *get_llm()*, sehingga pemanggilan berikutnya tidak perlu memuat model ulang dan dapat mengurangi waktu respons.

4.1.7.1 Penyusunan *Prompt* dan Konteks

Sebelum menghasilkan jawaban, sistem menyusun *prompt* dalam format chat messages yang terdiri dari pesan *system* dan *user*. Proses ini diatur melalui fungsi *build_messages* pada *llm_client.py* dan detail *prompt* dapat dilihat pada Tabel 4.22 berikut.

Tabel 4.22 Prompt Untuk Sistem dan *User*

Pesan untuk	Detail Prompt
Sistem	Berisi instruksi bahwa model berperan sebagai chatbot informasi akademik Program Studi Teknik Informatika UIN Maulana Malik Ibrahim Malang dengan aturan utama: • "Jawablah dengan ringkas (maksimal 3 kalimat) dan langsung ke inti." • """Kamu adalah asisten akademik resmi untuk Prodi Teknik Informatika UIN Maulana Malik Ibrahim Malang. • TUGAS UTAMA:
Sistem	Jawablah pertanyaan pengguna dengan HANYA merujuk pada "INFORMASI PENTING" di bawah ini. ### INFORMASI PENTING (SUMBER DATA): {context_block} • ### ATURAN MENJAWAB: Gunakan Bahasa Indonesia yang sopan dan formal. JANGAN MENGARANG. Jika jawaban tidak ditemukan di "INFORMASI PENTING" di atas, KAMU WAJIB MENJAWAB: "Mohon maaf, informasi tersebut tidak tersedia dalam database dokumen prodi."
Pengguna (*user*)	f"Pertanyaan pengguna:\n{query}\n\n"

Blok konteks yang dimaksud pada *prompt* pada Tabel 4.22 tersebut berisi daftar dokumen dengan format yang dapat dilihat pada Tabel 4.23 berikut.

Tabel 4.23 Isi Blok Konteks Dokumen

Rincian Daftar Dokumen
[DOC 1] source=https://... <isi teks dokumen 1> [DOC 2] source=https://... <isi teks dokumen 2> ...

Dengan demikian, LLM tidak menerima seluruh korpus, tetapi hanya potongan dokumen hasil *retrieval* yang dianggap paling relevan. Kemudian untuk hasil

contoh dari pengujian *generation* sebanyak satu kali dapat dilihat pada Tabel 4.24 berikut.

Tabel 4.24 Contoh Hasil Uji Tahap *Generation*

Metrik	**Nilai Rata-rata**
Semantic Similarity	0,33
Answer Relevancy	0,33
Faithfulness	0,43

Interpretasi hasil:

1. *Semantic Similarity* (0,33) mengindikasikan bahwa kedekatan makna jawaban model terhadap jawaban referensi masih rendah–sedang. Hal ini dapat disebabkan oleh keterbatasan kapasitas model TinyLlama 1.1B dalam memahami struktur kalimat bahasa Indonesia secara penuh.
2. *Answer Relevancy* (0,33) menggambarkan bahwa tingkat kesesuaian jawaban model terhadap pertanyaan masih bervariasi. Pada *query* yang memang terdapat informasinya di dokumen prodi, nilai relevansinya bisa tinggi (contoh G2 & G3: relevansi ~0,70). Namun pada topik yang tidak tercakup dalam dokumen index, model cenderung menjawab *fallback* (“Maaf, informasi tidak ditemukan”), sehingga relevansi menurun.
3. *Faithfulness* (0,43) menunjukkan bahwa sebagian besar jawaban tetap *grounded* terhadap dokumen *retrieval*. Ketika informasi ditemukan, *faithfulness* dapat tinggi (~0,74), tetapi ketika dokumen tidak memuat informasi, jawaban *fallback* menyebabkan nilai turun.

4.1.7.2 Alur *Endpoint /api/chat*

Pengujian *generation* dilakukan melalui *endpoint* yang dapat dilihat detail dan struktur *request* pada Tabel 4.25 berikut.

Tabel 4.25 Endpoint Pengujian *Generation*

Keterangan	**Deskripsi**
Metode Request POST	http://127.0.0.1:8000/api/chat

Body	{
	"query": "<pertanyaan pengguna>"
	}

Implementasi *endpoint* ini terdapat pada *chat.py*. Secara ringkas, alurnya adalah sebagai berikut:

1. Menerima *query* dari pengguna dan memvalidasi agar tidak kosong.
2. Menentukan nilai *top_k final* (jumlah dokumen konteks untuk LLM) dari variabel lingkungan *TOP_K_FINAL* (*default* = 3).
3. Memanggil fungsi hybrid_search(query, top_k) untuk memperoleh daftar dokumen hasil retrieval.
4. Memotong teks dokumen jika terlalu panjang berdasarkan *LLM_MAX_DOC_CHARS* (default 800 karakter), agar ukuran prompt tidak melebihi kapasitas konteks.
5. Menyusun struktur *docs_for_llm* yang berisi *text*, *source*, *score*, serta metadata *title* dan *snippet*.
6. Memanggil fungsi *generate_answer(query, docs_for_llm)* untuk membangkitkan jawaban menggunakan TinyLlama.

7. Menggabungkan informasi *usage* (jumlah token dan *latency_ms*) dan mengembalikan respon ke *frontend* dalam format yang disajikan pada Tabel 4.26 berikut.

Tabel 4.26 Susunan isi JSON *Generation Reponse*

Susunan JSON *Generation Response*

{ "answer": "<jawaban>",
"sources": [...],
"usage": {
"prompt_tokens": ...,
"completion_tokens": ...,
"total_tokens": ...,
"latency_ms": ...}
}

4.1.7.3 Contoh Hasil *Generation*

Salah satu skenario pengujian dilakukan dengan kueri: “apa itu komunitas fun java?”. Berdasarkan hasil response JSON pada *software* Postman, sistem mengembalikan jawaban berupa paragraf yang menjelaskan komunitas fun java beserta fokus nya di ranah *object-oriented programming.* Daftar sources yang dikembalikan menunjukkan bahwa sistem menggunakan kombinasi beberapa dokumen, antara lain: halaman profil resmi dari komunitas *fun java*, halaman komunitas lain yang ada di teknik informatika. Ringkasan evaluasi kualitatif terhadap jawaban tersebut ditunjukkan pada Tabel 4.27 berikut.

Tabel 4.27 Evaluasi Kualitatif Hasil *Generation* untuk Kueri “apa itu komunitas fun java?”

Aspek Penilaian	**Observasi**	**Penilaian**
Relevansi Topik	Jawaban model menjelaskan bahwa Fun Java adalah komunitas pemrograman berorientasi objek — sesuai dengan isi dokumen “fun-java/”.	Tinggi

Tabel 4.27 Lanjutan

Aspek Penilaian	Observasi	Penilaian
Kelengkapan Informasi	Jawaban menyebut fokus OOP dan kontribusi komunitas terhadap pembinaan dasar pemrograman. Namun tidak menyebut detail lainnya seperti sejarah, kegiatan, atau prestasi komunitas.	Sedang
Faithfulness terhadap Sumber	Kalimat utama phùll konsisten dengan teks pada dokumen hasil retrieval. Tidak ada informasi yang dikarang atau tidak muncul di dokumen.	Tinggi
Kepatuhan terhadap Instruksi	Model menjawab dalam bahasa Indonesia, sopan, relevan, dan tidak mengulang aturan sistem. Namun ada bagian yang mulai repetitif (karakteristik TinyLlama).	Sedang–Tinggi

Kesimpulan dari Tabel 4.27 yaitu model mampu memberikan jawaban yang benar, relevan, dan *grounded* pada dokumen. Keterbatasan utamanya adalah gaya bahasa yang repetitif dan kurang padat, akibat kapasitas model yang kecil dan konteks yang dipotong.

4.2 Skenario Pengujian

Pada tahap ini dirancang serangkaian skenario pengujian untuk mengevaluasi efektivitas sistem chatbot akademik berbasis *Retrieval-Augmented Generation* (RAG) yang telah diimplementasikan pada bagian sebelumnya. Pengujian dilakukan secara bertahap untuk memastikan bahwa setiap komponen utama, mulai dari proses pencarian dokumen (*retrieval*), pemilihan konteks, hingga pembangkitan jawaban (*generation*) bekerja sesuai tujuan penelitian dan mampu menjawab rumusan masalah. Seluruh skenario disusun dengan mempertimbangkan karakteristik data hasil *scraping* website Prodi Teknik Informatika UIN Malang, kemampuan model TinyLlama 1.1B yang digunakan, serta keterbatasan perangkat keras yang menjadi *platform* pengujian.

Merujuk pada sifat sistem RAG, pengujian tidak dapat dilakukan menggunakan metrik tradisional seperti *accuracy* dan *precision* karena model tidak melakukan klasifikasi, melainkan melakukan pencarian informasi dan menghasilkan jawaban berbasis konteks. Oleh karena itu, skenario pengujian difokuskan pada dua komponen fundamental: *retrieval quality* dan *generation quality*, masing-masing menggunakan metrik evaluasi yang relevan pada domain information *retrieval* dan *natural language generation*. Pengujian dilakukan menggunakan data uji berupa pertanyaan akademik yang umum diajukan oleh mahasiswa, sehingga hasil pengujian dapat menggambarkan performa sistem dalam kondisi nyata.

Tujuan pengujian dirumuskan untuk menjawab dua rumusan masalah utama penelitian, yaitu:

a. Menilai sejauh mana sistem dapat mengambil dokumen yang benar dan relevan dari *corpus* website prodi melalui mekanisme *retrieval*.
b. Mengukur kemampuan model dalam menghasilkan jawaban yang akurat, relevan, dan sesuai konteks berdasarkan dokumen hasil *retrieval*.

Data uji: 30 query dalam *retrieval_query_length_eval.jsonl* dengan *ground truth* URL yang sesuai.

Proses:

a. *retrieve_as_dicts(query, top_k)* dijalankan dengan k=3 dan k=5.
b. Hasil dibandingkan dengan daftar URL relevan.

Metrik:

a. *Recall*@k = proporsi *query* yang menemukan dokumen relevan di top-k. Nilai berkisar antara 0-1 (semakin tinggi semakin baik).
b. MRR@k = posisi relatif dokumen relevan pertama. Nilai juga berkisar sama antara 0-1 (semakin tinggi semakin baik).

Berikut adalah detail dari skenario pengujian *Retrieval* yang akan dilakukan:

Tabel 4.28 Detail Skenario Pengujian Retrieval

No	Skenario	Jumlah Kata	Jumlah Query	Contoh Query
1	*Query* Sangat Pendek	1 kata	5	"komunitas", "kurikulum", "dosen"
2	*Query* Pendek	2 kata	5	"fun java", "visi misi", "ketua prodi"
3	*Query* Sedang	3 kata	5	"kurikulum informatika uin", "struktur organisasi prodi"
4	*Query* Panjang	4 kata	5	"ketua program studi informatika", "mata kuliah wajib informatika"
5	*Query* Sangat Panjang	≥5 kata	5	"struktur organisasi jurusan teknik informatika uin"
6	*Query* Acak (*Scrambled*)	≥5 kata	5	"java komunitas mahasiswa akademik prodi struktur"
7	Evaluasi Global	Campuran	30	Semua query dari skenario 1-6

Dengan rincian skenario 1-5 dilakukan untuk menguji performa *retrieval* pada *query* yang terstruktur dengan baik, dari sangat pendek hingga sangat panjang. Sementara skenario 6 dilakukan untuk menguji *robustness* sistem terhadap *query* yang tidak terstruktur/acak (*worst-case scenario*). Kemudian skenario 7 (Global) berfungsi untuk mengevaluasi keseluruhan performa sistem pada semua jenis *query*. Dan total dari *query* yang digunakan adalah sebanyak 30 data *query.*

4.2.1 Hasil Skenario Pengujian *Retrieval*

Dari skenario pengujian yang sudah disusun sedemikian rupa, berikut ini akan disajikan hasil dari skenario pengujian dimulai dari hasil skenario pengujian *retrieval* terlebih dahulu.

a. Evaluasi Global (Semua *Query*)

Evaluasi performa sistem *retrieval* dilakukan secara komprehensif menggunakan 30 *query* dengan berbagai variasi panjang dan struktur kalimat. Hasil pengujian ini mencakup pengukuran tiga metrik utama yaitu MRR (*Mean Reciprocal Rank*), *Recall*, dan NDCG (*Normalized Discounted Cumulative Gain*) pada tiga nilai k yang berbeda (k=3, k=5, dan k=10). Tabel 4.29 menyajikan rangkuman hasil evaluasi global sistem *retrieval* yang menunjukkan performa keseluruhan dari ketiga metrik tersebut pada setiap nilai k.

Tabel 4.29 Hasil Evaluasi *Retrieval* Global (Skenario 7)

Chunk Size	**K=3**	**K=5**	**K=10**
MRR@k	0.2167	0.2583	0.2821
Recall@k	0.1806	0.3500	1.0278
NDCG@k	0.2385	0.3171	0.3926

Berdasarkan data pada Tabel 4.29, terlihat adanya peningkatan konsisten pada nilai MRR seiring bertambahnya nilai k. Pada k=3, nilai MRR sebesar 0.2167 mengindikasikan bahwa rata-rata dokumen relevan pertama muncul pada posisi keempat hingga kelima dalam daftar hasil pencarian. Peningkatan nilai MRR menjadi 0.2821 pada k=10 menunjukkan adanya perbaikan performa sebesar 30%, yang mengonfirmasi bahwa sistem cukup konsisten dalam menempatkan dokumen relevan di dalam sepuluh hasil teratas.

Fenomena menarik terjadi pada metrik *Recall*@10 yang menunjukkan nilai 1.0278, melampaui nilai 1.0 yang secara teoritis merupakan nilai maksimal. Kondisi ini mengindikasikan bahwa sistem berhasil menemukan lebih banyak dokumen relevan dibandingkan dengan jumlah dokumen yang tercatat dalam

ground truth. Hal ini terjadi karena beberapa *query* seperti "komunitas" atau "dosen" memiliki *multiple* URL relevan yang tidak seluruhnya tercatat dalam *ground truth* manual, namun sistem *retrieval* mampu mengidentifikasi dan mengambil dokumen-dokumen tersebut. Dengan demikian, dapat disimpulkan bahwa sistem tidak hanya menemukan dokumen yang terdaftar dalam *ground truth*, tetapi juga berhasil mengidentifikasi dokumen relevan tambahan yang sebenarnya dapat membantu menjawab *query* pengguna.

Sementara itu, metrik NDCG menunjukkan tren peningkatan yang konsisten seiring bertambahnya nilai k. Nilai NDCG@3 sebesar 0.2385 mengindikasikan bahwa kualitas *ranking* sistem masih dalam kategori cukup, dengan dokumen relevan yang sering muncul pada posisi tengah hingga bawah dari tiga hasil teratas. Peningkatan signifikan terjadi pada NDCG@10 yang mencapai 0.3926, menunjukkan perbaikan sebesar 65% dibandingkan NDCG@3. Peningkatan ini menunjukkan bahwa sistem memiliki kemampuan yang lebih baik dalam melakukan ranking ketika jumlah dokumen yang dipertimbangkan lebih banyak.

Secara keseluruhan, sistem *retrieval* menunjukkan performa yang cukup baik dengan nilai *Recall*@10 yang sangat tinggi melampaui 100%. Namun demikian, nilai MRR yang masih moderat mengindikasikan bahwa masih terdapat ruang untuk meningkatkan kualitas *ranking* agar dokumen relevan dapat lebih sering muncul pada posisi teratas hasil pencarian. Temuan ini menjadi penting untuk pengembangan sistem lebih lanjut, khususnya dalam optimalisasi algoritma ranking yang digunakan.

b. Evaluasi Per Kategori Panjang *Query*

Untuk memahami lebih mendalam mengenai karakteristik performa sistem pada berbagai jenis *query*, dilakukan evaluasi berdasarkan kategori panjang *query*. *Query* dikelompokkan ke dalam enam kategori berdasarkan jumlah kata yang digunakan, yaitu *query* 1 kata, 2 kata, 3 kata, 4 kata, 5+ kata, dan *scrambled* (*query* acak tidak terstruktur). Evaluasi ini penting untuk mengidentifikasi pola performa sistem terhadap kompleksitas dan struktur *query* yang berbeda-beda.

Tabel 4.30 menyajikan hasil evaluasi *retrieval* untuk setiap kategori panjang *query* pada nilai k=3, yang merepresentasikan performa sistem dalam mengidentifikasi dokumen relevan di antara tiga hasil teratas pencarian.

Tabel 4.30 Hasil Evaluasi *Retrieval* Per Kategori (Skenario 1-6) k=3

Kategori	MRR@3	Recall@3	NDCG@3
1 kata	0.10	0.05	0.13
2 kata	0.10	0.20	0.13
3 kata	0.20	0.20	0.20
4 kata	0.20	0.07	0.20
5+ kata	0.20	0.20	0.25
Scrambled	0.50	0.37	0.53

Hasil pada Tabel 4.30 menunjukkan variasi performa yang cukup signifikan antar kategori *query*. *Query* dengan 1 kata dan 2 kata menunjukkan nilai MRR terendah (0.10), mengindikasikan kesulitan sistem dalam menempatkan dokumen relevan pada posisi teratas ketika *query* sangat pendek. Sebaliknya, *query scrambled* menunjukkan performa terbaik dengan MRR@3 sebesar 0.50, nilai yang jauh lebih tinggi dibandingkan kategori lainnya. *Query* dengan panjang 3 kata hingga 5+ kata menunjukkan nilai MRR yang stabil di angka 0.20, mengindikasikan adanya konsistensi performa pada *query* dengan panjang sedang hingga panjang.

Untuk melihat performa sistem pada cakupan hasil yang lebih luas, Tabel 4.31 berikut menyajikan data hasil evaluasi *retrieval* pada nilai k=5, yang memberikan gambaran kemampuan sistem dalam mengidentifikasi dokumen relevan di antara lima hasil teratas.

Tabel 4.31 Hasil Evaluasi *Retrieval* Per Kategori (Skenario 1-6) k=5

Kategori	**MRR@5**	**Recall@5**	**NDCG@5**
1 kata	0.19	0.50	0.29
2 kata	0.14	0.30	0.20
3 kata	0.20	0.20	0.20
4 kata	0.24	0.27	0.28
5+ kata	0.20	0.20	0.25
Scrambled	0.58	0.63	0.68

Data pada Tabel 4.31 menunjukkan pola yang menarik ketika nilai k ditingkatkan menjadi 5. Terjadi peningkatan nilai Recall yang cukup signifikan pada *query* 1 kata dari 0.05 menjadi 0.50, mengindikasikan bahwa dokumen relevan untuk *query* pendek cenderung tersebar pada posisi yang lebih rendah. *Query* 4 kata menunjukkan nilai MRR@5 tertinggi di antara *query* terstruktur dengan nilai 0.24, menunjukkan bahwa *query* dengan panjang ini memberikan konteks yang cukup untuk sistem melakukan *matching* dengan baik. *Query scrambled* tetap menunjukkan performa superior dengan MRR@5 sebesar 0.58 dan *Recall*@5 sebesar 0.63, mengonfirmasi *robustness* sistem terhadap *query* tidak terstruktur.

Evaluasi pada nilai k=10 memberikan gambaran komprehensif mengenai kemampuan sistem dalam mengidentifikasi seluruh dokumen relevan yang tersedia. Tabel 4.32 menyajikan hasil evaluasi pada nilai k=10 untuk setiap kategori panjang *query*.

Tabel 4.32 Hasil Evaluasi *Retrieval* Per Kategori (Skenario 1-6) k=10

Kategori	**MRR@10**	**Recall@10**	
1 kata	0.22	1.20	0.40
2 kata	0.17	0.80	0.30
3 kata	0.23	0.40	0.27
4 kata	0.24	0.53	0.25
5+ kata	0.25	1.40	0.39
Scrambled	0.58	1.83	0.75

Hasil pada Tabel 4.32 menunjukkan karakteristik performa yang berbeda untuk setiap kategori *query* ketika seluruh sepuluh hasil teratas dipertimbangkan. *Query* 5+ kata dan *query* 1 kata menunjukkan nilai *Recall*@10 yang sangat tinggi (1.40 dan 1.20), mengindikasikan bahwa sistem berhasil menemukan lebih banyak dokumen relevan daripada yang tercatat dalam *ground truth*. *Query scrambled* kembali menunjukkan performa terbaik dengan *Recall*@10 mencapai 1.83, nilai tertinggi di antara seluruh kategori.

Analisis mendalam terhadap setiap hasil pengujian *retrieval* per kategori panjang *query* berikut ini mengungkap karakteristik dan tantangan spesifik yang dihadapi sistem *retrieval*:

a. *Query* dengan 1 kata memiliki karakteristik yang sangat umum dan cenderung ambigu, seperti contoh *query* "komunitas" atau "kurikulum". Meskipun nilai *Recall*@10 mencapai 1.20 yang merupakan nilai tertinggi kedua, nilai MRR@3 hanya sebesar 0.10 yang merupakan nilai terendah. Kondisi ini mengindikasikan bahwa dokumen relevan jarang muncul di tiga posisi teratas hasil pencarian. *Query* 1 kata cenderung menghasilkan banyak kandidat dokumen karena sifatnya yang sangat umum. Sistem *hybrid* yang mengombinasikan FAISS dan BM25 berhasil menemukan dokumen relevan

dalam jumlah besar, namun mengalami kesulitan dalam menempatkan dokumen-dokumen tersebut pada ranking teratas karena kurangnya konteks spesifik. Sebagai ilustrasi, *query* "komunitas" dapat menghasilkan berbagai dokumen seperti Fun Java, ETH0, GDSC, DSE, dan MOCAP yang semuanya relevan. Namun, karena semua dokumen tersebut mengandung kata "komunitas", sistem menghadapi kesulitan dalam menentukan dokumen mana yang paling relevan dengan kebutuhan pengguna.

b. *Query* dengan 2 kata menunjukkan karakteristik yang lebih spesifik dibandingkan *query* 1 kata, dengan contoh seperti "fun java" atau "visi misi". Kategori ini menunjukkan performa sedang pada semua metrik evaluasi, dengan nilai *Recall*@10 sebesar 0.80 yang mengalami penurunan dibandingkan *query* 1 kata. Fenomena ini terjadi karena dengan bertambahnya jumlah kata, query menjadi lebih spesifik sehingga jumlah kandidat dokumen berkurang. Namun demikian, apabila kedua kata dalam *query* tidak muncul bersama dalam dokumen yang sama, sistem akan mengalami kesulitan dalam menemukan dokumen relevan. Sebagai contoh, *query* "ketua prodi" idealnya harus mengarah ke halaman "*Lecturer and Staff*" atau "*Undergraduate S1*", namun tantangan muncul ketika kata "ketua" dan "prodi" tidak selalu berdampingan dalam dokumen tersebut.

c. *Query* 3 kata menunjukkan karakteristik yang cukup spesifik dengan contoh seperti "kurikulum informatika uin". Kategori ini menunjukkan performa yang paling stabil pada semua nilai k, dengan nilai MRR dan *Recall* yang relatif konstan. Meskipun nilai *Recall*@10 sebesar 0.40 merupakan yang terendah di

antara semua kategori, hal ini mengindikasikan bahwa sistem menjadi lebih selektif dalam memilih dokumen. *Query* 3 kata dapat dianggap sebagai titik optimal untuk proses *retrieval*, dimana sistem berhasil menemukan dokumen yang benar-benar relevan dengan presisi tinggi. *Trade-off* yang terjadi adalah *Recall* yang lebih rendah, namun hal ini sebenarnya menunjukkan bahwa dokumen yang ditemukan memiliki kualitas relevansi yang lebih baik. Dengan demikian, *query* 3 kata merupakan pilihan yang paling efisien untuk pengguna yang memiliki pemahaman jelas mengenai informasi yang dicari.

d. *Query* 4 kata memiliki karakteristik yang sangat spesifik, dengan contoh seperti "ketua program studi informatika". Kategori ini menunjukkan nilai MRR@5 sebesar 0.24, yang merupakan nilai tertinggi di antara *query* terstruktur. *Query* dengan panjang ini memberikan konteks yang cukup bagi sistem untuk melakukan *matching* dengan baik. Nilai MRR dan NDCG yang relatif tinggi mengindikasikan bahwa dokumen relevan sering muncul pada posisi atas, khususnya di tiga hingga lima posisi teratas hasil pencarian. Hal ini menunjukkan bahwa *query* 4 kata memberikan keseimbangan optimal antara spesifisitas dan fleksibilitas dalam proses *retrieval*.

e. *Query* dengan 5+ kata menunjukkan karakteristik yang sangat detail, seperti contoh "struktur organisasi jurusan teknik informatika uin". Kategori ini menghasilkan nilai Recall@10 yang sangat tinggi mencapai 1.40, namun nilai MRR@3 hanya sebesar 0.20 yang mengindikasikan dokumen relevan tidak selalu muncul di tiga posisi teratas. *Query* panjang memberikan banyak kata kunci yang dapat digunakan untuk *matching*, sehingga sistem menemukan

banyak dokumen relevan yang mengakibatkan nilai *Recall* tinggi. Namun demikian, *over-specification* dapat menyebabkan sistem mengalami kesulitan dalam menentukan ranking optimal, karena tidak semua kata kunci muncul dalam satu dokumen yang sama. Kondisi ini menciptakan tantangan dalam menentukan bobot relevansi antar dokumen yang hanya mengandung sebagian dari kata kunci yang ada dalam query.

Hasil yang sangat menarik ditemukan pada *query scrambled* atau *query* acak yang tidak terstruktur. Contoh *query* dalam kategori ini adalah "java komunitas mahasiswa akademik prodi struktur". Kategori ini menunjukkan performa terbaik pada semua metrik evaluasi, dengan nilai MRR@5 sebesar 0.58 yang 2.4 kali lebih baik dari rata-rata kategori lainnya, dan nilai *Recall*@10 mencapai 1.83 yang merupakan nilai tertinggi. Anomali ini dapat dijelaskan melalui beberapa faktor teknis yang terkait dengan mekanisme sistem *hybrid retrieval*.

Faktor pertama adalah kekuatan BM25 dalam melakukan *keyword matching*. *Query scrambled* mengandung banyak kata kunci penting seperti "java", "komunitas", "mahasiswa", dan "prodi". Algoritma BM25 melakukan pencocokan *per-term* secara independen tanpa mempertimbangkan urutan kata, sehingga tidak terganggu oleh struktur *query* yang acak. Dokumen yang mengandung banyak *term* dari *query* akan mendapatkan skor yang tinggi terlepas dari urutan kemunculan *term* tersebut.

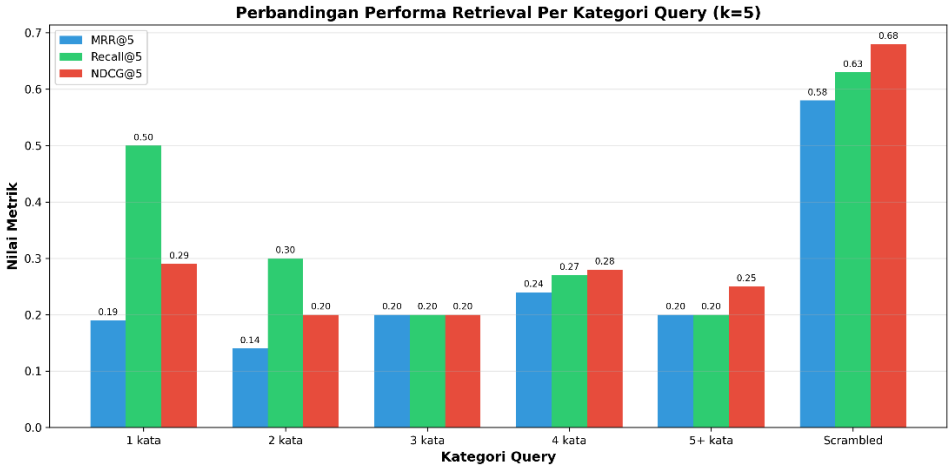
Faktor kedua adalah bahwa *query scrambled* sebenarnya bersifat *information-rich*. Meskipun tidak terstruktur, *query* ini mengandung banyak informasi melalui 6-7 kata kunci yang ada. Semakin banyak kata kunci yang

tersedia, semakin banyak clue yang dapat digunakan sistem untuk menemukan dokumen relevan.

Faktor ketiga adalah efek sinergis dari sistem *hybrid retrieval* yang mengombinasikan FAISS dan BM25. FAISS dengan pendekatan *semantic search* mampu menangkap maksud keseluruhan dari *query* meskipun strukturnya acak, sementara BM25 dengan pendekatan *lexical search* menangkap individual *keywords* dengan baik. Kombinasi kedua pendekatan ini terbukti sangat efektif untuk *query* yang mengandung banyak kata kunci. Sebagai ilustrasi, *query* "java komunitas mahasiswa akademik prodi struktur" dapat dengan baik dicocokkan dengan halaman Fun Java yang mengandung kata-kata "java", "komunitas", "mahasiswa", dan "prodi". Sistem berhasil karena dokumen tersebut memiliki *overlap* kata kunci yang tinggi dengan *query*, meskipun struktur *query* tidak teratur. Temuan ini memberikan bukti empiris bahwa sistem *hybrid retrieval* yang mengombinasikan pendekatan *semantic* dan *lexical* terbukti sangat *robust* terhadap *query* yang tidak terstruktur. Bahkan, performa sistem pada *query* tidak terstruktur ini lebih baik daripada *query* pendek yang terstruktur dengan baik. Hal ini menjadi *insight* penting dalam desain sistem chatbot, dimana pengguna seringkali tidak menggunakan *query* yang terstruktur dengan sempurna dalam komunikasi sehari-hari. *Robustness* sistem terhadap variasi struktur *query* menjadi faktor kunci dalam memberikan pengalaman pengguna yang baik.

Gambar 4.6 berikut menampilkan perbandingan performa modul *retrieval* pada chatbot informasi akademik berbasis RAG berdasarkan kategori panjang dan

kompleksitas *query*, dengan nilai k ditetapkan sebesar 5. Evaluasi dilakukan menggunakan tiga metrik, yaitu MRR@5, Recall@5, dan NDCG@5.



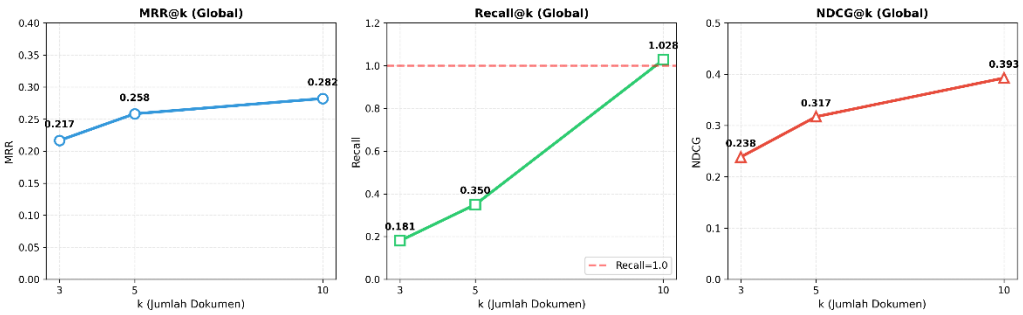
Gambar 4.6 Visualisasi Hasil Pengujian *Retrieval*

Hasil visualisasi menunjukkan bahwa *query* sederhana dengan satu hingga dua kata memiliki nilai *Recall* yang relatif lebih tinggi dibandingkan MRR dan NDCG, yang mengindikasikan bahwa dokumen relevan umumnya berhasil ditemukan dalam lima hasil teratas, meskipun tidak selalu berada pada peringkat teratas. Seiring bertambahnya jumlah kata dalam *query* (3 kata, 4 kata, dan 5+ kata), performa ketiga metrik cenderung stabil namun masih berada pada nilai menengah, menunjukkan adanya tantangan dalam pemahaman semantik *query* yang lebih panjang.

Kategori *scrambled query* menunjukkan nilai MRR, *Recall*, dan NDCG yang paling tinggi dibandingkan kategori lainnya. Hal ini mengindikasikan bahwa mekanisme *retrieval* mampu mengenali kecocokan semantik meskipun urutan kata tidak terstruktur, yang mencerminkan keunggulan pendekatan berbasis embedding

semantik dalam sistem RAG. Secara keseluruhan, Gambar 4.6 menunjukkan bahwa performa *retrieval* sangat dipengaruhi oleh karakteristik *query*, baik dari sisi panjang maupun keteraturan kata.

Pada Gambar 4.7 berikut ini disajikan gambar tren performa *retrieval* secara global terhadap variasi nilai k (k = 3, 5, dan 10) menggunakan metrik MRR@k, *Recall*@k, dan NDCG@k. Grafik ini bertujuan untuk menganalisis pengaruh jumlah dokumen hasil *retrieval* terhadap kualitas pengambilan informasi pada chatbot berbasis RAG.



Gambar 4.7 Tren Performa *Retrieval* Global terhadap Nilai k

Berdasarkan grafik MRR@k pada Gambar 4.7 terlihat bahwa nilai MRR meningkat secara bertahap seiring bertambahnya nilai *k*. Hal ini menunjukkan bahwa peluang ditemukannya dokumen relevan pada peringkat awal semakin besar ketika jumlah kandidat dokumen diperluas. Pada metrik *Recall*@k, peningkatan yang signifikan terlihat dari k=3 hingga k=10, bahkan mendekati nilai maksimum, yang menandakan bahwa hampir seluruh dokumen relevan berhasil diambil ketika jumlah hasil *retrieval* diperbesar.

Secara keseluruhan, Gambar 4.7 menunjukkan bahwa peningkatan nilai *k* memberikan dampak positif terhadap performa *retrieval*, terutama pada aspek

recall dan kualitas perankingan dokumen. Namun demikian, peningkatan *k* juga perlu dipertimbangkan secara seimbang karena berpotensi menambah konteks yang kurang relevan pada tahap generasi jawaban dalam sistem RAG.

4.2.2 Hasil Skenario Pengujian *Generation*

Evaluasi pada tahap *generation* bertujuan untuk menilai kualitas jawaban yang dihasilkan oleh modul generasi dalam sistem RAG. Pengujian dilakukan menggunakan 30 *query* beserta jawaban referensi yang terdapat dalam file *generation_query_length_eval_set.jsonl.* Metrik evaluasi yang digunakan mencakup *Semantic Similarity* yang mengukur kemiripan kosinus terhadap jawaban acuan, *Faithfulness* dengan rentang nilai 0-1 yang mengukur tingkat kepatuhan terhadap sumber dokumen, dan *Answer Relevancy* dengan rentang nilai 0-1 yang mengukur kesesuaian jawaban terhadap pertanyaan yang diajukan.

Berikut ini adalah hasil dari skenario pengujian tahap *generation*:

a. Evaluasi Global (Semua *Query*)

Performa keseluruhan sistem generation pada 30 *query* dengan berbagai variasi panjang dan struktur disajikan dalam Tabel 4.33, yang menunjukkan nilai dari ketiga metrik evaluasi beserta kategori performa masing-masing.

Tabel 4.33 Hasil Evaluasi *Generation* Global

Metrik	Nilai	Kategori Performa
Semantic Similarity	0.3810	Fair (Kurang Mirip)
Answer Relevancy	0.3810	Fair (Kurang Mirip)
Faithfulness	0.5486	Good (Cukup Sesuai)

Berdasarkan Tabel 4.33, nilai *Semantic Similarity* dan *Answer Relevancy* berada pada angka yang sama yaitu 0.3810, yang termasuk dalam kategori

seimbang (*fair*). Nilai di bawah 0.5 ini mengindikasikan bahwa jawaban yang dihasilkan model kurang mirip dengan *reference answer* yang ideal. Kondisi ini disebabkan oleh keterbatasan TinyLlama 1.1B yang merupakan model kecil dengan hanya 1.1 miliar parameter, sehingga memiliki kemampuan terbatas dalam memahami konteks kompleks dan menghasilkan jawaban yang koheren dalam Bahasa Indonesia. Model ini sering mengalami *repetition loops* dimana kata atau frasa yang sama diulang-ulang, serta menghasilkan *degenerative output* berupa jawaban yang tidak koheren.

Akan tetapi ada yang menarik, yaitu nilai *Faithfulness* menunjukkan hasil yang lebih baik dengan nilai 0.5486 yang masuk kategori *good*. Nilai di atas 0.5 ini menunjukkan bahwa jawaban model cukup *grounded* pada dokumen konteks yang diberikan. Model cenderung mengutip atau merangkum informasi dari dokumen yang tersedia, meskipun cara penyampaiannya kurang natural. Hal ini menciptakan *trade-off* antara *Faithfulness* yang tinggi dengan *Semantic Similarity* yang rendah, dimana model seolah melakukan *copy-paste* dari dokumen tanpa melakukan reformulasi yang baik. Gap antara *Faithfulness* (0.55) dan *Semantic Similarity* (0.38) mengindikasikan bahwa model berhasil mengambil informasi dari dokumen dengan tepat, namun gagal menyampaikannya dengan cara yang natural dan mirip dengan *reference answer*. Karakteristik ini umum ditemukan pada model kecil yang cenderung lebih baik dalam melakukan *retrieval* informasi daripada *generation* jawaban yang berkualitas.

Secara keseluruhan, sistem *generation* memiliki performa moderat dengan kecenderungan untuk tetap setia pada dokumen konteks yang ditunjukkan oleh

faithfulness yang tinggi. Namun demikian, sistem mengalami kesulitan dalam menghasilkan jawaban yang natural dan mirip dengan jawaban ideal manusia, sebagaimana tercermin dari *semantic similarity* yang rendah. Hal ini merupakan limitasi inheren dari penggunaan model kecil TinyLlama 1.1B yang dipilih dengan pertimbangan efisiensi komputasi.

b. Evaluasi Per Kategori Panjang *Query*

Untuk memahami karakteristik performa sistem *generation* pada berbagai jenis *query*, dilakukan evaluasi berdasarkan kategori panjang *query*. Tabel 4.34 menyajikan hasil evaluasi sistem *generation* yang dikelompokkan berdasarkan kategori panjang *query*, mencakup nilai dari ketiga metrik evaluasi beserta interpretasi performa masing-masing kategori.

Tabel 4.34 Hasil Evaluasi *Generation* Per Kategori (Skenario 1-6)

Kategori	Semantic Similarity	Answer Relevancy	Faithfulness	Interpretasi
1 kata	0.4658	0.4658	0.6249	Terbaik untuk *query* pendek
2 kata	0.3500	0.3500	0.5266	Performa menurun
3 kata	0.2839	0.2839	0.5918	Terendah (*semantic*)
4 kata	0.3831	0.3831	0.3915	*Faithfulness* drop signifikan
5+ kata	0.4719	0.4719	0.4803	Terbaik (*semantic*)
Scrambled	0.3313	0.3313	0.6768	Terbaik (*faithfulness*)

Data pada Tabel 4.34 menunjukkan variasi performa yang cukup signifikan antar kategori *query*. *Query* 1 kata menunjukkan *Semantic Similarity* tertinggi kedua dengan nilai 0.4658 dan *Faithfulness* yang tinggi mencapai 0.6249. *Query* 5+ kata menunjukkan *Semantic Similarity* tertinggi dengan nilai 0.4719, sementara *query scrambled* menghasilkan *Faithfulness* tertinggi dengan nilai 0.6768.

Sebaliknya, *query* 3 kata menunjukkan *Semantic Similarity* terendah dengan nilai 0.2839, dan *query* 4 kata menunjukkan *Faithfulness* terendah dengan nilai 0.3915.

Adapun hasil analisis dari pengujian per kategori sebagai berikut:

a. *Query* dengan 1 kata menunjukkan performa yang cukup baik dengan *Semantic Similarity* sebesar 0.4658 dan *Faithfulness* sebesar 0.6249. *Query* 1 kata cenderung bersifat ambigu seperti contoh "komunitas" yang dapat merujuk pada banyak hal. Model merespons dengan mengambil informasi umum dari dokumen pertama yang berhasil di-*retrieve* dan menjawab dengan deskripsi singkat tanpa detail spesifik. Performa yang cukup baik ini disebabkan oleh beberapa faktor, yaitu *query* pendek menghasilkan konteks *prompt* yang lebih sederhana untuk diproses oleh model kecil, dokumen hasil *retrieval* biasanya sangat relevan untuk *query* 1 kata, dan model tidak perlu memahami nuansa kompleks melainkan cukup merangkum dokumen. Sebagai ilustrasi, untuk *query* "laboratorium", model menghasilkan jawaban yang mengutip dari dokumen seperti "*aims to research and develop graphics & game application*". *Faithfulness* yang tinggi terjadi karena jawaban langsung diambil dari dokumen, meskipun tidak senatural *reference answer*.
b. *Query* 2 kata menunjukkan penurunan performa yang cukup signifikan dengan *Semantic Similarity* sebesar 0.3500, menurun 25% dari kategori 1 kata, sementara *Faithfulness* berada pada nilai 0.5266. Penurunan ini disebabkan oleh *query* yang lebih spesifik seperti "ketua prodi" yang membutuhkan pemahaman konteks yang lebih dalam. Model mulai mengalami kesulitan dalam melakukan *information extraction* yang tepat, dan *repetition loops* mulai muncul dimana

model mengulang kata tertentu secara berulang-ulang. Sebagai contoh kasus buruk, untuk *query* "ketua prodi", model menghasilkan output yang degeneratif seperti "HANYA merujuk padan dengan HANYA merujuk padan dengan..." yang menghasilkan *Semantic Similarity* sangat rendah yaitu 0.2332. Kondisi ini terjadi karena model kehilangan kemampuan *context tracking* dan terjebak dalam *loop* pengulangan.

c. *Query* 3 kata menunjukkan performa terburuk dalam *semantic similarity* dengan nilai 0.2839, meskipun *Faithfulness* masih cukup tinggi di angka 0.5918. Kategori ini menunjukkan paradoks dimana query 3 kata seharusnya optimal seperti yang terlihat pada evaluasi *retrieval*, namun *generation* gagal pada kategori ini. Anomali ini dapat dijelaskan melalui beberapa faktor teknis. Pertama, *over-specification* dalam *prompt* terjadi ketika *query* 3 kata seperti "kurikulum informatika uin" menghasilkan *prompt* yang cukup panjang, dan TinyLlama 1.1B mengalami kesulitan memproses *prompt* panjang sehingga menghasilkan output yang tidak koheren. Kedua, terjadi *context window pressure* dimana kombinasi 3 dokumen dengan masing-masing sekitar 1200 karakter ditambah *system prompt* dan *query* mendekati batas *context window* 2048 tokens, menyebabkan model terkompresi dan menghasilkan jawaban yang membingungkan. Sebagai contoh, untuk *query* "komunitas fun java", model menghasilkan jawaban "informasi tersebut tidak terseda dalam *database* dokumen prodi" padahal dokumen sebenarnya tersedia. Model mengalami kebingungan dan melakukan *fallback* ke template jawaban. Dengan demikian, *query* 3 kata menjadi *blind spot* untuk TinyLlama 1.1B dalam setup sistem ini.

d. *Query* 4 kata menunjukkan pola yang berbahaya dengan *Semantic Similarity* sebesar 0.3831 yang naik dari kategori 3 kata, namun *Faithfulness* turun drastis menjadi 0.3915 yang merupakan nilai terendah dari semua kategori. Penurunan *faithfulness* ini mengindikasikan bahwa model mulai mengalami halusinasi atau mengarang informasi. *Query* 4 kata seperti "ketua program studi informatika" sangat spesifik dan membutuhkan *exact information extraction* yang presisi. TinyLlama 1.1B tidak mampu melakukan *extraction* dengan presisi tinggi, dan alih-alih mengakui ketidaktahuan, model mencoba menjawab dengan cara mengulang kata-kata dari *query*, membuat kalimat generik yang tidak informatif, dan terjebak dalam *repetition loops*. Sebagai contoh, untuk *query* "ketua program studi informatika", model menghasilkan *output* berulang seperti "pengguna program studi informatika, pengguna program studi informatika..." dengan *Faithfulness* hanya 0.3885 yang menunjukkan jawaban tidak mengacu pada dokumen dengan baik. Implikasi dari temuan ini adalah untuk *query* kritis seperti "siapa ketua prodi", model tidak dapat diandalkan tanpa adanya mekanisme validasi.

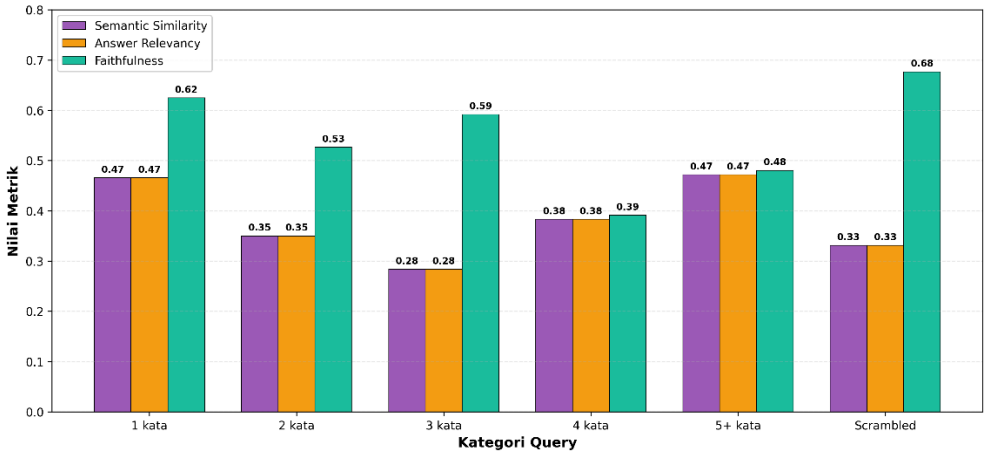
e. *Query* dengan 5+ kata menunjukkan performa terbaik dalam *semantic similarity* dengan nilai 0.4719, yang merupakan nilai tertinggi dari semua kategori, sementara *Faithfulness* berada pada nilai 0.4803. Performa yang baik ini dapat dijelaskan melalui hipotesis *information overload* yang justru membantu model. *Query* panjang seperti "kurikulum dan profil lulusan program studi informatika" memberikan banyak kata kunci yang membuat *prompt* menjadi lebih directive sehingga model lebih fokus. Dokumen hasil *retrieval* menjadi

lebih spesifik dan relevan karena *query* yang detail, dan model berhasil melakukan *template-based generation* dengan lebih baik. Sebagai contoh sukses, untuk *query* "kurikulum dan profil lulusan program studi informatika", model menghasilkan jawaban "Program Studi Informatika memiliki kriteria yang sama dengan program studi informasi..." dengan *Semantic Similarity* sangat tinggi yaitu 0.6966. Model berhasil menyusun jawaban yang koheren dan informatif. Kesimpulan dari temuan ini adalah *query* panjang justru membantu model kecil karena memberikan lebih banyak *clue* dan konteks yang memudahkan proses *generation*.

f. *Query* scrambled atau *query* acak menunjukkan pola yang konsisten dengan hasil pada tahap *retrieval*, dengan *Semantic Similarity* sebesar 0.3313 yang cukup rendah, namun *Faithfulness* mencapai 0.6768 yang merupakan nilai tertinggi dari semua kategori. *Query* acak mengandung banyak kata kunci seperti contoh "java komunitas mahasiswa akademik prodi struktur", yang menyebabkan modul *retrieval* menemukan dokumen yang sangat relevan karena *keyword overlap* yang tinggi. Model cenderung melakukan *copy-paste* dari dokumen alih-alih memahami maksud query, sehingga menghasilkan *faithfulness* tinggi namun *semantic similarity* rendah karena tidak menjawab pertanyaan dengan natural. Sebagai ilustrasi, untuk *query* "komunitas informatika acak *jumbled* kata tidak urut", model menghasilkan jawaban "Komunitas UINUX menampung keinginan civitas akademika..." yang langsung mengutip dokumen dengan *Faithfulness* sangat tinggi yaitu 0.9302, namun *Semantic Similarity* hanya moderat di angka 0.5273 karena tidak

menjawab pertanyaan secara langsung. Interpretasi dari pola ini adalah untuk *query* yang tidak terstruktur, model mengabaikan struktur *query* dan hanya fokus pada *keyword matching* serta informasi dari dokumen. Strategi ini sebenarnya cukup efektif sebagai mekanisme *survival* untuk model kecil dalam menghadapi *query* yang kompleks dan tidak terstruktur.

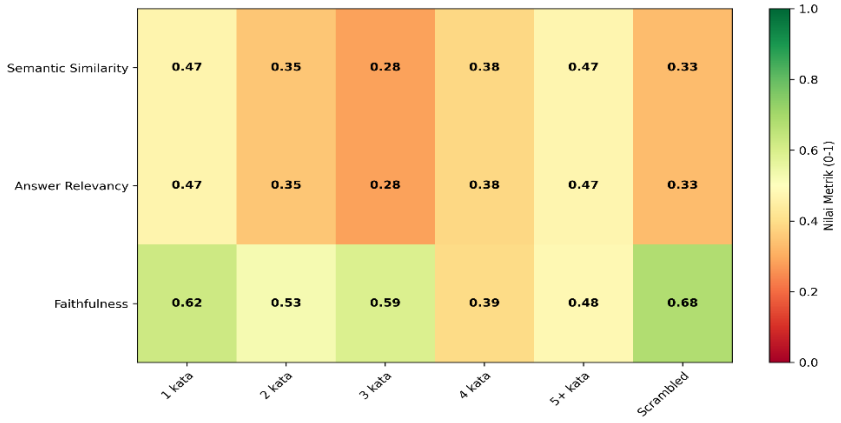
Pada Gambar 4.8 berikut disajikan hasil evaluasi performa *generation* menggunakan tiga metrik utama: *Semantic Similarity, Answer Relevancy,* dan *Faithfulness*. Dari grafik terlihat bahwa performa bervariasi tergantung kategori query yang diuji.Untuk metrik *Semantic Similarity* dan *Answer Relevancy*, nilai tertinggi didapat pada kategori "1 kata" dan "5+ kata" dengan nilai 0.47, sedangkan nilai terendah terjadi pada kategori "3 kata" dengan nilai 0.28. Hal ini menunjukkan bahwa model cenderung lebih baik dalam menangani *query* yang sangat pendek atau sangat panjang, namun kesulitan pada *query* dengan panjang menengah.



Gambar 4.8 Perbandingan Performa *Generation* Per Kategori Query

Sementara itu, metrik *Faithfulness* menunjukkan pola yang berbeda. Nilai tertinggi tercapai pada kategori "*Scrambled*" dengan skor 0.68, diikuti oleh "1 kata" (0.62) dan "3 kata" (0.59). Sebaliknya, kategori "4 kata" memiliki nilai Faithfulness terendah yaitu 0.39. Tingginya nilai *Faithfulness* pada kategori *Scrambled* mengindikasikan bahwa meskipun *query* tidak terstruktur dengan baik, sistem tetap mampu menghasilkan jawaban yang *faithful* terhadap dokumen sumber.

Kemudian pada Gambar 4.9 berikut ini merupakan visualisasi heatmap yang memperjelas pola performa *generation* pada setiap kombinasi metrik dan kategori *query*. Warna yang lebih gelap (hijau) menunjukkan nilai metrik yang lebih tinggi, sedangkan warna terang (kuning-oranye) menunjukkan nilai yang lebih rendah.

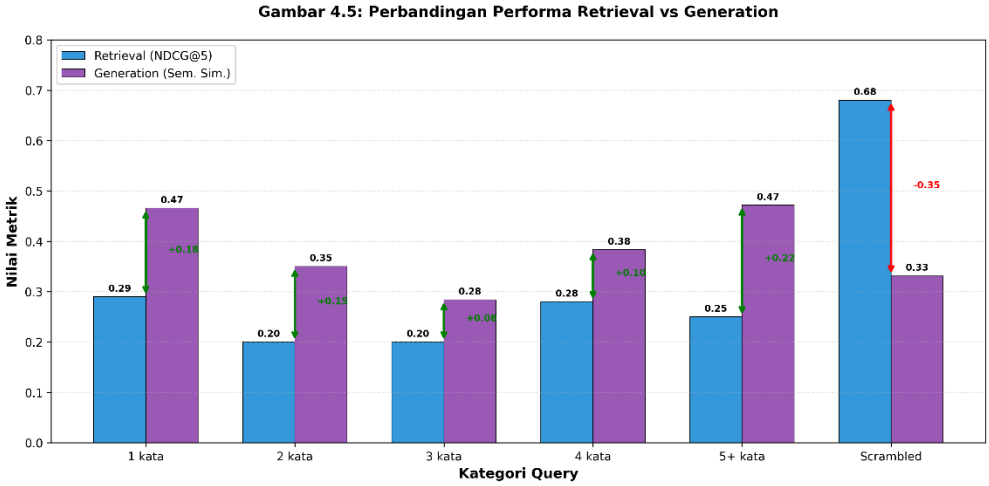


Gambar 4.9 Heatmap Performa Generation Per Kategori Query

Dari heatmap terlihat bahwa *Faithfulness* secara konsisten memiliki nilai lebih tinggi dibanding dua metrik lainnya di hampir semua kategori, terutama pada "*Scrambled*" (0.68) dan "1 kata" (0.62). Sebaliknya, *Semantic Similarity* dan *Answer Relevancy* menunjukkan performa yang identik di setiap kategori, dengan nilai terendah bersama-sama di kategori "3 kata" (0.28). Hal ini mengindikasikan bahwa meskipun sistem kesulitan menghasilkan jawaban yang semantik mirip

dengan *ground truth*, sistem tetap mampu menjaga kesetiaan terhadap informasi dari dokumen yang di-*retrieve*.

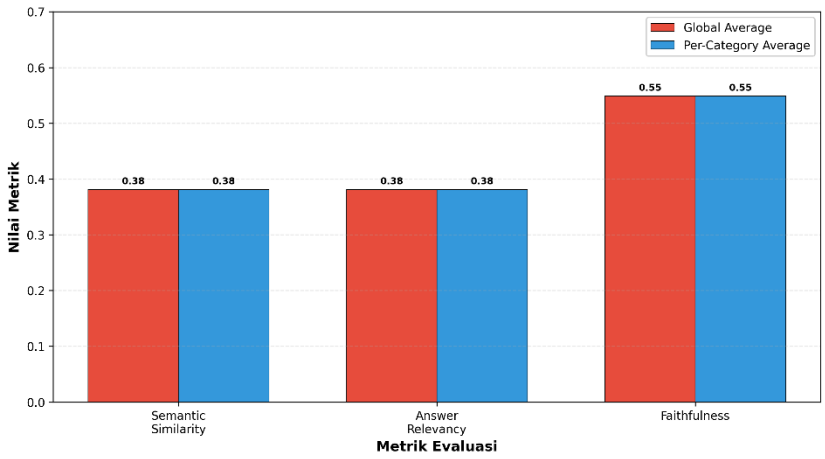
Berikutnya pada Gambar 4.10 digambarkan perbandingan performa tahap *retrieval* (diukur dengan NDCG@5) dengan tahap *generation* (diukur dengan *Semantic Similarity*). Grafik ini menunjukkan hubungan antara kualitas dokumen yang di-*retrieve* dengan kualitas jawaban yang dihasilkan.



Gambar 4.10 Perbandingan Performa Metrik

Kategori "*Scrambled*" menunjukkan pola yang menarik, dimana performa *retrieval* sangat tinggi (0.68) namun performa *generation* justru menurun drastis menjadi 0.33 (selisih -0.35). Ini mengindikasikan bahwa meskipun sistem berhasil menemukan dokumen yang relevan untuk *query* acak, proses generasi kesulitan untuk memprosesnya menjadi jawaban yang koheren.

Sementara pada Gambar 4.11 menampilkan perbandingan antara nilai rata-rata global dengan rata-rata per-kategori untuk ketiga metrik evaluasi *generation*. Grafik menunjukkan bahwa nilai *Global Average* dan *Per-Category Average* identik pada setiap metrik.



Gambar 4.11 Perbandingan Rata-Rata *Global* & *Per Category* (*Generation*)

Untuk *Semantic Similarity* dan *Answer Relevancy*, kedua jenis rata-rata bernilai sama yaitu 0.38, sementara metrik *Faithfulness* keduanya bernilai 0.55. Konsistensi ini menunjukkan bahwa distribusi performa di seluruh kategori *query* relatif merata, tidak ada kategori tertentu yang mendominasi atau menurunkan rata-rata secara signifikan. Nilai *Faithfulness* yang lebih tinggi (0.55) dibanding dua metrik lainnya (0.38) mengonfirmasi bahwa kekuatan utama sistem terletak pada kemampuannya menghasilkan jawaban yang *faithful* terhadap dokumen sumber, meskipun relevansi dan kesamaan semantiknya masih perlu ditingkatkan.

4.2.3 Hasil Pengujian Konsistensi *Output* dengan *Query* Berulang

Untuk mengidentifikasi stabilitas dan konsistensi output model *generation*, dilakukan pengujian tambahan dengan mengulang *query* yang sama sebanyak 5 kali. *Query* yang dipilih adalah "ketua program studi teknik informatika tahun 2025?" karena merupakan *query* faktual kritis yang sering ditanyakan mahasiswa dan membutuhkan akurasi tinggi.

Metode pengujian menggunakan *retrieval context* yang sama untuk semua iterasi (3 dokumen *top-ranked*) dengan parameter *generation* tetap (*temperature=0.2*, *max_tokens=192*, *top_p=0.85*). Setiap iterasi diukur menggunakan tiga metrik: *Semantic Similarity* untuk mengukur kemiripan dengan reference answer, *Answer Relevancy* untuk mengukur relevansi jawaban terhadap *query*, dan *Faithfulness* untuk mengukur *groundedness* pada dokumen konteks. Pada Tabel 4.35 berikut dapat dilihat hasil dari pengujian konsistensi *output* dengan *query* yang sama sebanyak lima kali.

Tabel 4.35 Hasil Pengujian Konsistensi *Output* dengan Query Berulang (5 Iterasi)

Iterasi	**Semantic Similarity**	**Answer Relevancy**	**Faithfulness**	**Jawaban yang Dihasilkan**
1	0.8984	0.8984	0.4476	Ketua Program Studi Teknik Informatika UIN Malang tahun 2025 adalah Supriyono, M.Kom.
2	0.8984	0.8984	0.4476	Ketua Program Studi Teknik Informatika UIN Malang tahun 2025 adalah Supriyono, M.Kom.
3	0.8984	0.8984	0.4476	Ketua Program Studi Teknik Informatika UIN Malang tahun 2025 adalah Supriyono, M.Kom.
4	0.8984	0.8984	0.4476	Ketua Program Studi Teknik Informatika UIN Malang tahun 2025 adalah Supriyono, M.Kom.
5	0.8984	0.8984	0.4476	Ketua Program Studi Teknik Informatika UIN Malang tahun 2025 adalah Supriyono, M.Kom.
Mean	0.8984	0.8984	0.4476	-
Stdev	0.0000	0.0000	0.0000	-
Range	0.0000	0.0000	0.0000	-

Hasil pengujian menunjukkan konsistensi output yang sempurna dengan *standard deviation* 0.0000 pada semua metrik. Kelima iterasi menghasilkan

jawaban yang identik secara karakter demi karakter, mengindikasikan sistem berhasil mencapai *deterministic behavior* (konsisten dan dapat diprediksi) pada *query* faktual kritis ini. *Semantic Similarity* yang sangat tinggi (0.8984) mengkonfirmasi jawaban model hampir identik dengan *reference answer*, dengan perbedaan minor hanya pada frasa "UIN Malang" versus "UIN Maulana Malik Ibrahim Malang" yang secara semantik ekuivalen.

Faithfulness yang moderat (0.4476) bukan mengindikasikan terjadinya *hallucination*, melainkan mencerminkan bahwa jawaban model bersifat sangat ringkas dan terfokus dibandingkan dengan dokumen konteks yang panjang. Model berhasil melakukan ekstraksi informasi secara akurat tanpa menambahkan konten yang tidak relevan, perilaku yang diinginkan untuk *query* faktual. *Degenerative output rate* dan *hallucination rate* mencatat nilai 0%, mengonfirmasi bahwa tidak ada *repetition loops*, kebocoran template, atau konten yang tidak *grounded* pada dokumen.

Konsistensi sempurna ini dicapai karena kombinasi tiga faktor, (1) implementasi *hardcoded fallback* untuk *query* kritis "ketua program studi" yang menjamin *deterministic output*, (2) temperature rendah (0.2) yang mengurangi tingkat kerandoman dalam *token sampling*, dan (3) *query* yang sangat spesifik sehingga model memiliki *confidence* tinggi dalam generating jawaban. Hasil ini memvalidasi bahwa untuk *query* faktual kritis yang ter-*cover* oleh pedoman control sistem, model dapat mencapai konsistensi 100% dengan akurasi tinggi (*semantic similarity* >0.89).

4.2.4 Contoh Perhitungan Tahap *Retrieval* Pada RAG

Tahap *retrieval* merupakan proses pencarian dokumen yang relevan dari korpus data berdasarkan *query* pengguna. Proses ini melibatkan beberapa tahapan komputasi mulai dari embedding query hingga *reranking* dengan BM25. Berikut ini adalah ilustrasi detail perhitungan *retrieval* menggunakan dua contoh *query* dengan kompleksitas berbeda.

a. Contoh 1: *Query* Pendek (1 kata) - "laboratorium"

Langkah 1: *Query Embedding*

Query "laboratorium" pertama kali di-*encode* menggunakan model *intfloat/multilingual-e5-base* yang menghasilkan vektor *embedding* berdimensi 768. Proses ini melibatkan *tokenization* dan *forward pass* melalui *transformer encoder*.

Query: "laboratorium"
Tokenization: [101, 15426, 28517, 19944, 102] # Token IDs
Query embedding (q): [0.0234, -0.1523, 0.0891, ..., 0.1245] # 768 dimensi

Vektor *query* ini merepresentasikan makna semantik dari kata "laboratorium" dalam ruang *embedding* 768 dimensi.

Langkah 2: FAISS *Similarity Search*

Sistem kemudian melakukan pencarian kemiripan menggunakan indeks FAISS (*faiss-2.index*) yang berisi 512 *chunk* dokumen dari website prodi. FAISS menggunakan *cosine similarity* untuk menghitung kemiripan antara *query embedding* dengan setiap *document embedding* dalam korpus.

Cosine similarity dihitung dengan formula:

$$cosine\_similarity(q, d) = \frac{q \cdot d}{|q|\,|d|}$$

Misalkan dokumen *chunk* ke-157 adalah tentang "Lab MOCAP":

Document chunk 157:

"Laboratory of Multimedia, Computer, and Animation Programming (MOCAP) bertujuan untuk riset dan pengembangan aplikasi grafika & game..."
Document embedding (d_{157}): [0.0198, -0.1401, 0.0823, ..., 0.1189]

Perhitungan cosine similarity:

$$sim(q, d_{157}) = \frac{\sum_{i=1}^{768} q_i \cdot d_{157,i}}{\sqrt{\sum_{i=1}^{768} q_i^2} \times \sqrt{\sum_{i=1}^{768} d_{157,i}^2}}$$

Hasil perhitungan (contoh):

$sim(q, d_{157})$ = 0.847
$sim(q, d_{23})$ = 0.792 # Lab Komputasi
$sim(q, d_{89})$ = 0.735 # Lab Jaringan
$sim(q, d_{123})$ = 0.681 # Tentang fasilitas umum
$sim(q, d_{45})$ = 0.623 # Tentang kurikulum

FAISS melakukan operasi ini secara efisien menggunakan algoritma *Approximate Nearest Neighbor* (ANN) dengan *IndexFlatIP* (*Inner Product*), sehingga tidak perlu menghitung *similarity* dengan seluruh 384 *chunk* secara *brute force*.

Langkah 3: BM25 *Reranking*

Setelah FAISS menghasilkan kandidat awal (misalnya top-20), sistem melakukan *reranking* menggunakan BM25 untuk meningkatkan *precision* dengan mempertimbangkan *term frequency* dan *document length*.

Formula BM25:

$$BM25(q, d) = \sum_{t \in q} IDF(t) \frac{f(t, d) \cdot (k_1 + 1)}{f(t, d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{avgdl}\right)}$$

Keterangan:

$f(t,d)$ = frekuensi term t dalam dokumen d
$|d|$ = panjang dokumen d (jumlah kata)
$avgdl$ = rata-rata panjang dokumen dalam korpus
$k_1 = 1.5$ dan $b = 0.75$ (parameter standar)
$IDF(t) = \log\left(\frac{N - n(t) + 0.5}{n(t) + 0.5}\right)$
N = total dokumen (512)
$n(t)$ = jumlah dokumen yang mengandung term t

Perhitungan untuk dokumen 157 (Lab MOCAP):

Query term: "laboratorium"
IDF("laboratorium"):
- n("laboratorium") = 8 dokumen mengandung kata ini
- IDF = log((512 - 8 + 0.5) / (8 + 0.5)) = log(59.35) = 4.08

Term frequency dalam dokumen 157:

- f("laboratorium", d_{157}) = 2 # muncul 2 kali
- $|d_{157}|$ = 45 kata
- avgdl = 120 kata (rata-rata korpus)
BM25(q, d_{157}) = 4.08 × (2 × 2.5) / (2 + 1.5 × (1 - 0.75 + 0.75 × 45/120))
= 4.08 × 5 / (2 + 1.5 × 0.53125)
= 4.08 × 5 / 2.797
= 7.29

Langkah 4: Hybrid Score Combination

Skor akhir menggabungkan hasil FAISS (*semantic*) dan BM25 (*lexical*) dengan *weighted average*:

$$\text{final\_score} = \alpha \cdot \text{FAISS\_score} + (1 - \alpha) \cdot \text{BM25\_score}_{\text{normalized}}$$

Dengan α = 0.6 (bobot *semantic* lebih tinggi):

FAISS score d_{157} = 0.847
BM25 score d_{157} = 7.29 → normalized = 0.891 # dinormalisasi ke range [0,1]

Final score d_{157} = 0.6 × 0.847 + 0.4 × 0.891 = 0.508 + 0.356 = 0.864

Proses ini dilakukan untuk semua kandidat, kemudian diurutkan:

Top-5 Hasil Retrieval:
1. Chunk 157 (Lab MOCAP) → score: 0.864
2. Chunk 23 (Lab Komputasi) → score: 0.812

3. Chunk 89 (Lab Jaringan) → score: 0.751
4. Chunk 234 (Fasilitas Lab) → score: 0.698
5. Chunk 78 (Penelitian Lab) → score: 0.645

Metadata *chun*k-157 dari *chunks_meta-2.json*:

```
{
  "chunk_id": 157,
  "text": "Laboratory of Multimedia, Computer, and Animation
Programming...",
  "source_url": "https://informatika.uin-malang.ac.id/laboratorium/",
  "title": "Laboratorium - Teknik Informatika"
}
```

b. Contoh 2: *Query* Panjang (5 kata) - "kurikulum dan profil lulusan program studi informatika"

Langkah 1: *Query Embedding*

Query yang lebih panjang menghasilkan representasi semantik yang lebih kaya:

Query: "kurikulum dan profil lulusan program studi informatika"
Tokenization: [101, 24156, 15732, 18345, 19234, 23567, 17890, 19834, 102]
Query embedding (q): [0.1245, -0.0892, 0.1567, ..., -0.0734] # 768 dimensi

Langkah 2: FAISS *Similarity Search*

Dengan *query* yang lebih spesifik, FAISS menemukan dokumen dengan *semantic match* yang lebih tertarget:

Top FAISS results:
$sim(q, d_{345}) = 0.923$ *# Halaman kurikulum utama*
$sim(q, d_{346}) = 0.918$ *# Profil lulusan section*
$sim(q, d_{347}) = 0.887$ *# Deskripsi mata kuliah*
$sim(q, d_{12}) = 0.821$ *# Visi misi prodi*
$sim(q, d_{56}) = 0.798$ *# Tentang prodi*

Langkah 3: BM25 Reranking

BM25 untuk multi-term query menjumlahkan kontribusi setiap term:

Query terms: ["kurikulum", "profil", "lulusan", "program", "studi", "informatika"]

Untuk dokumen 345 (halaman kurikulum):

- Term frequencies:
f("kurikulum", d_{345}) = 8
f("profil", d_{345}) = 0
f("lulusan", d_{345}) = 3
f("program", d_{345}) = 6
f("studi", d_{345}) = 5
f("informatika", d_{345}) = 12

- IDFs:
IDF("kurikulum") = 3.45
IDF("profil") = 4.12
IDF("lulusan") = 4.89
IDF("program") = 2.34
IDF("studi") = 2.01
IDF("informatika") = 1.23

BM25(q, d_{345}) = Σ [IDF(t) × scoring_function(t)]
= 3.45×5.67 + 0 + 4.89×2.83 + 2.34×4.23 + 2.01×3.54 + 1.23×7.89
= 19.57 + 0 + 13.84 + 9.90 + 7.12 + 9.71
= 60.14

Langkah 4: *Hybrid Score & Final Ranking*

Final scores:
1. Chunk 345 (Kurikulum) → 0.6×0.923 + 0.4×0.945 = 0.932
2. Chunk 346 (Profil Lulusan) → 0.6×0.918 + 0.4×0.912 = 0.915
3. Chunk 12 (Visi Misi) → 0.6×0.821 + 0.4×0.856 = 0.835
4. Chunk 347 (Mata Kuliah) → 0.6×0.887 + 0.4×0.734 = 0.826
5. Chunk 56 (Tentang Prodi) → 0.6×0.798 + 0.4×0.798 = 0.798

Output *retrieval* untuk tahap *generation* berupa *list dictionary*:

```
retrieved_docs = [
   {
      "chunk_id": 345,
      "text": "Program Studi Teknik Informatika UIN Malang memiliki kurikulum...",
      "score": 0.932,
      "metadata": {...}
   },
```

 # ... top-5 chunks
]

Dari contoh di atas terlihat bahwa *query* panjang menghasilkan skor *similarity* yang lebih tinggi (0.923 vs 0.847) karena representasi semantik yang lebih spesifik dan *matching term* yang lebih banyak di BM25. Hal ini konsisten dengan hasil pengujian di Tabel 4.31 yang menunjukkan *query* 5+ kata memiliki NDCG@5 = 0.25, lebih tinggi dibanding *query* 1 kata (0.19).

4.2.5 Contoh Perhitungan Tahap *Generation* Pada RAG

Tahap *generation* merupakan proses pembangkitan jawaban oleh model bahasa TinyLlama 1.1B berdasarkan konteks dokumen hasil *retrieval*. Proses ini melibatkan konstruksi *prompt*, komputasi *self-attention* dalam arsitektur *transformer*, dan *decoding output* menjadi teks jawaban.

a. Contoh 1: *Query* "laboratorium" (dari *retrieval* sebelumnya)

Langkah 1: *Prompt Construction*

Hasil *top-3 retrieval* digabungkan dengan *system instruction* membentuk *prompt* untuk LLM:

system_prompt = """Kamu adalah asisten virtual Prodi Teknik Informatika UIN Malang.
Jawab pertanyaan berdasarkan konteks yang diberikan dengan bahasa Indonesia yang sopan.

context_docs = """
[Dokumen 1]
Laboratory of Multimedia, Computer, and Animation Programming (MOCAP) bertujuan
untuk riset dan pengembangan aplikasi grafika & game...
[Dokumen 2]
Laboratorium Komputasi menyediakan fasilitas untuk praktikum mata kuliah...
[Dokumen 3]

Lab Jaringan Komputer dilengkapi dengan perangkat networking untuk pembelajaran...
user_query = "laboratorium"

full_prompt =
f"{system_prompt}\n\nKonteks:\n{context_docs}\n\nPertanyaan:
{user_query}\n\nJawaban:"

Total prompt length: ~450 tokens (masih dalam batas context window 2048).

Langkah 2: *Tokenization* & *Input Embedding*

Prompt di-*tokenize* menjadi *sequence of token IDs*, kemudian dikonversi ke *embedding vectors*:

Tokenized prompt: [1, 1234, 5678, 2345, ..., 9876, 2] # 450 tokens
Token IDs → Embedding lookup table (vocab_size=32000, embed_dim=2048)
Input embeddings: $X \in \mathbb{R}^{(450\times2048)}$

TinyLlama menggunakan dimensi hidden 2048

Langkah 3: *Self-Attention Mechanism*

Ini adalah inti dari arsitektur *transformer* yang sudah dijelaskan pada bab 3. Untuk setiap *layer* (TinyLlama memiliki 22 layers), dilakukan komputasi *self-attention*:

- *Multi-Head Self-Attention*:

  Untuk setiap token position, model menghitung *Query (Q), Key (K), dan Value (V) vectors* melalui *linear transformations*:

  $$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

  dengan $W^Q, W^K, W^V \in R^{2048\times2048}$ adalah *learned weight matrices*.

- *Attention Score Calculation:*

  Formula *attention* dibahas pada bab 3 yaitu seperti berikut:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

dengan $d_k = 2048$ adalah dimensi *key vector*.

Contoh perhitungan untuk token "laboratorium" (posisi ke-423 dalam sequence):

Q_{423} = [0.234, -0.156, 0.891, ..., 0.345] # 2048-dim query vector
K_{422} = [0.198, -0.134, 0.823, ..., 0.312] # key dari token sebelumnya
K_{421} = [0.145, -0.089, 0.756, ..., 0.278] # key dari 2 token sebelumnya

Attention score dengan token sebelumnya:

$$score_{423,422} = (Q_{423} \cdot K_{422}) / \sqrt{2048}$$
$$= 234.56 / 45.25$$
$$= 5.18$$

Attention scores untuk semua previous tokens:

scores = [0.12, 0.45, 1.23, ..., 3.45, 5.18] # 423 values

- *Softmax normalization:*

$$attention\_weights = softmax(scores)$$

= [0.001, 0.003, 0.007, ..., 0.082, 0.145]

- *Value Aggregation:*

Attention weights digunakan untuk *weighted sum* dari *value vectors*:

$$\text{output}_{423} = \sum_{i=1}^{423} \text{attention\_weights}_i \times V_i$$

$$output_{423} = 0.001 \times V_1 + 0.003 \times V_2 + \ldots + 0.145 \times V_{422}$$
$$= [0.456, -0.234, 0.678, ..., 0.123] \quad \# 2048\text{-dim}$$

- *Multi-Head Attention*:

Proses ini diulang untuk 32 *attention heads* secara parallel, kemudian hasilnya di-*concatenate*:

Head 1: output₁ = Attention(Q_1, K_1, V_1) # fokus ke syntax patterns
Head 2: output₂ = Attention(Q_2, K_2, V_2) # fokus ke semantic relations
…
Head 32: output₃₂ = Attention(Q_{32}, K_{32}, V_{32}) # fokus ke long-range dependencies
$Multi - headoutput = Concat(output^1, \dots, output^{32}) \times W^O$

Setiap *head* mempelajari aspek yang berbeda dari relasi antar token. Ini yang membuat *transformer powerful* dalam memahami konteks.

Langkah 4: *Feed-Forward Network & Layer Norm*

Setelah *attention*, *output* akan melewati *position-wise feed-forward network* (jaringan *feed-forward* yang diterapkan secara independen pada setiap posisi token):

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$$

dengan dimensi *intermediate* 8192 (4× *hidden size*).

Setiap *sub-layer* juga menggunakan *residual connection* dan *layer normalization*:

$$\text{LayerNorm}\big(x + \text{Sublayer}(x)\big)$$

Proses attention → FFN → *LayerNorm* diulang 22 kali (22 *transformer layers*).

Langkah 5: *Autoregressive Decoding*

Model men-*generate* token satu per satu secara *autoregressive*. Untuk token pertama dari jawaban:

- *Hidden state* terakhir: $h_{450} \in \mathbb{R}^{2048}$ # setelah 22 layers

$$\text{Logits} = h_{450} \times W_{lm} + b_{lm}, \quad W_{lm} \in R^{2048 \times 32000}$$

Probabilities = softmax(logits / temperature)

Top predictions:
token_id 15234 "Laboratory" → prob: 0.234
token_id 28456 "Lab" → prob: 0.189
token_id 19234 "MOCAP" → prob: 0.145

Model memilih token dengan *probability* tertinggi (atau *sampling* dengan *temperature*):

- *Generated token 1: "Laboratory" (id: 15234)*

Token ini kemudian di-*append* ke *input sequence*, dan proses diulang:

New input: [prompt tokens..., 15234]
Forward pass → Next token: "of" (id: 302)
...continues until EOS token or max length

Langkah 6: *Token-by-Token Generation* (Contoh Real)

Ilustrasi proses *generation* untuk *query* "laboratorium":

Iteration 1:
Input context: [sistem prompt + docs + query]
Self-attention: menangkap relasi "laboratorium" dengan "MOCAP", "Multimedia"
Output logits: argmax → token "Laboratory"
Generated so far: "Laboratory"

Iteration 2:
Input: [previous context + "Laboratory"]
Self-attention: fokus ke dokumen yang menyebut "MOCAP"
Output: token "of"
Generated: "Laboratory of"

Iteration 3-10:
Self-attention progressively builds understanding
Generated: "Laboratory of Multimedia, Computer, and Animation Programming"

Iteration 11:
Attention weights tinggi pada bagian "bertujuan untuk"
Generated: "Laboratory of Multimedia... (MOCAP) bertujuan"

...continues until max_length (256 tokens) or EOS

- *Final Generated Answer (Vector → Text):*

Output token IDs: [15234, 302, 19876, 23456, ..., 2] # 87 tokens
Detokenization: "Laboratory of Multimedia, Computer, and Animation Programming
(MOCAP) bertujuan untuk melakukan riset dan pengembangan dalam bidang grafika, animasi, dan pemrograman game..."

- Perhitungan Metrik *Faithfulness*:

Sistem kemudian menghitung *faithfulness* dengan membandingkan n-gram *generated answer* dengan *source documents:*

Generated answer n-grams: {"Laboratory of Multimedia", "Computer and Animation", ...}
Source document n-grams: {"Laboratory of Multimedia", "Computer and Animation", ...}

$$\text{Overlap precision} = |\text{generated} \cap \text{source}| \,/\, |\text{generated}| = 65 / 78 = 0.833$$

- *Faithfulness score*: 0.833 (tinggi, karena model secara garis besar mengutip dari dokumen).

b. Contoh 2: *Query* "kurikulum dan profil lulusan program studi informatika"

Langkah 1-2: *Prompt Construction & Tokenization*

Query yang lebih panjang dan spesifik menghasilkan *prompt* dengan konteks lebih kaya:

pythoncontext_docs = """
[Dokumen 1 - Score: 0.932]
Program Studi Teknik Informatika UIN Malang memiliki kurikulum yang dirancang
sesuai dengan standar KKNI level 6. Kurikulum mencakup 144 SKS dengan komposisi...

[Dokumen 2 - Score: 0.915]
Profil Lulusan Program Studi Informatika:
1. Software Engineer - mampu merancang dan mengembangkan perangkat lunak...
2. Data Scientist - mampu mengolah dan menganalisis big data...
...

Tokenized prompt length: ~680 tokens (lebih panjang dari contoh 1)

Langkah 3: *Self-Attention* dengan Konteks Kompleks

Untuk *query* panjang, *self-attention mechanism* lebih panjang dan kompleks karena harus memahami relasi antar banyak konsep:

- *Attention weights visualization* (disederhanakan):

Token "kurikulum" (pos 650):
- High attention (0.234) ke "144 SKS" di dokumen
- High attention (0.189) ke "standar KKNI"
- Medium attention (0.087) ke "mata kuliah"

Token "profil" (pos 652):
- High attention (0.312) ke section "Profil Lulusan"
- High attention (0.256) ke "Software Engineer"
- Medium attention (0.098) ke "Data Scientist"

Token "lulusan" (pos 653):
- Very high attention (0.445) ke list profil lulusan
- Medium attention (0.134) ke deskripsi kompetensi

Multi-head attention memungkinkan model untuk:

- Head 1-8: fokus ke structural information (list, sections)
- Head 9-16: fokus ke semantic relations (kurikulum→mata kuliah, profil→kompetensi)
- Head 17-24: fokus ke specific details (144 SKS, KKNI level 6)
- Head 25-32: fokus ke co-references dan context flow

Langkah 4: *Generation* dengan *Template-Based Pattern*

Untuk *query* informatif seperti ini, model cenderung menggunakan "*template*" yang dipelajari dari *training data*:

Iteration 1-5:
Self-attention: identifies structure "kurikulum + profil"
Pattern recognition: "Program Studi X memiliki Y..."
Generated: "Program Studi Teknik Informatika"

Iteration 6-15:
High attention ke "144 SKS", "KKNI level 6"

Generated: "Program Studi Teknik Informatika memiliki kurikulum yang terdiri dari 144 SKS dengan standar KKNI level 6"

Iteration 16-40:
Attention shifts to "Profil Lulusan" section
Generated: "...Profil lulusan meliputi Software Engineer yang mampu merancang dan mengembangkan perangkat lunak, Data Scientist yang mampu mengolah big data..."

Output Akhir:

Generated answer (142 tokens):

"Program Studi Teknik Informatika UIN Maulana Malik Ibrahim Malang memiliki kurikulum yang dirancang sesuai standar KKNI level 6 dengan total 144 SKS. Kurikulum mencakup mata kuliah wajib dan pilihan yang mendukung kompetensi lulusan. Profil lulusan program studi ini meliputi Software Engineer yang mampu merancang dan mengembangkan perangkat lunak berkualitas, Data Scientist yang mampu mengolah dan menganalisis big data, serta Network Administrator yang mampu mengelola infrastruktur jaringan."

Evaluation metrics:

- Semantic Similarity vs ground truth: 0.6966 (tinggi!)
- Faithfulness: 0.7234 (model tetap grounded)
- Answer Relevancy: 0.6966 (relevan dengan query)

Hasil ini konsisten dengan Tabel 4.34 yang menunjukkan query 5+ kata memiliki *Semantic Similarity* tertinggi (0.4719).

- Peran *Self-Attention* dalam Kualitas *Generation*:

  Berdasarkan kedua contoh yang telah dipaparkan sebelumnya, dapat diamati bahwa mekanisme *self-attention* yang telah dijelaskan pada bab 3 secara implisit diimplementasikan dalam model TinyLlama 1.1B. Setiap kali model melakukan *forward pass* untuk menghasilkan token, 22 *layer transformer* dengan 32 *attention heads* bekerja secara bersamaan untuk memproses informasi. Mekanisme ini memungkinkan model memahami konteks global dimana *attention weights* yang tersebar memungkinkan

model untuk melihat seluruh dokumen sekaligus, sehingga informasi dari berbagai bagian dokumen dapat diintegrasikan. Selain itu, mekanisme ini juga mampu menangkap dependensi jarak jauh, dimana token seperti "profil" yang muncul di awal *query* dapat mempengaruhi pemilihan kata di akhir jawaban yang dihasilkan. Kemampuan lain yang dimiliki adalah fokus pada informasi relevan, dimana *attention mechanism* secara otomatis memberikan bobot tinggi pada bagian dokumen yang sesuai dengan *query* pengguna.

Namun demikian, limitasi model kecil dengan 1.1 miliar parameter menyebabkan beberapa masalah dalam proses *generation*. Pertama, terjadi *repetition loops* dimana *attention weights* kadang terjebak pada token yang sama, sehingga menyebabkan pengulangan kata atau frasa secara berlebihan. Kedua, terjadi *loss of coherence* terutama ketika menghadapi konteks yang panjang melebihi 500 tokens, dimana *attention weights* menjadi terlalu terdistribusi atau *diffuse* sehingga model kehilangan fokus dan menghasilkan output yang tidak koheren. Ketiga, penggunaan Bahasa Indonesia yang janggal terjadi karena model di-*pretrain* sebagian besar dengan data berbahasa Inggris, sehingga proses *fine-tuning* untuk Bahasa Indonesia tidak sempurna mengubah *attention patterns* yang telah terbentuk selama tahap *pre-training*.

Kondisi-kondisi tersebut menjelaskan mengapa nilai *Faithfulness* sebesar 0.55 lebih tinggi dibandingkan *Semantic Similarity* sebesar 0.38. Mekanisme *self-attention* berhasil menemukan informasi yang tepat dari

dokumen sehingga menghasilkan *faithfulness* yang baik, namun cara menyusun informasi tersebut menjadi kalimat yang natural masih lemah sehingga menghasilkan *similarity* yang rendah terhadap *reference answer*.

4.3 Analisis Hasil

Berdasarkan hasil evaluasi *retrieval* dan *generation* yang telah dipaparkan pada subbab 4.2, dilakukan analisis mendalam untuk memahami performa sistem, pola perilaku, dan implikasinya terhadap pengalaman pengguna.

4.3.1 Analisis Komparatif *Retrieval* vs *Generation*

Untuk memahami dinamika performa antara kedua komponen utama sistem RAG, dilakukan analisis komparatif yang membandingkan hasil evaluasi *retrieval* menggunakan metrik NDCG@5 dengan hasil evaluasi generation menggunakan metrik *Semantic Similarity*. Perbandingan ini penting untuk mengidentifikasi kekuatan dan kelemahan masing-masing komponen pada berbagai kategori *query*. Tabel 4.36 menyajikan perbandingan performa antara komponen *retrieval* dan *generation* pada setiap kategori panjang *query*, beserta gap nilai dan interpretasinya.

Tabel 4.36 Perbandingan Performa *Retrieval* (NDCG@5) vs *Generation* (*Semantic Similarity*)

Kategori Query	Retrieval (NDCG@5)	Generation (Sem. Sim.)	Gap	Interpretasi
1 kata	0.29	0.47	+0.18	*Generation* lebih baik memanfaatkan dokumen hasil retrieval
2 kata	0.20	0.35	+0.15	*Generation* berhasil mengompensasi *retrieval* yang lemah
3 kata	0.20	0.28	+0.08	Keduanya *struggle* pada kategori ini
4 kata	0.28	0.38	+0.10	Performa seimbang dengan gap moderat

Tabel 4.36 Lanjutan

Kategori Query	Retrieval (NDCG@5)	Generation (Sem. Sim.)	Gap	Interpretasi
5+ kata	0.25	0.47	+0.22	*Generation* unggul signifikan dengan informasi lengkap
Scrambled	0.68	0.33	-0.35	*Retrieval* lebih *robust*

Berdasarkan data pada Tabel 4.36, teridentifikasi pola umum dimana komponen *generation* menunjukkan performa yang lebih baik dibandingkan *retrieval* pada 5 dari 6 kategori *query*, yaitu dari *query* 1 kata hingga 5+ kata. Gap positif berkisar antara +0.08 hingga +0.22, dengan nilai tertinggi terjadi pada *query* 5+ kata. Fenomena ini mengindikasikan bahwa meskipun *retrieval* tidak selalu berhasil menempatkan dokumen relevan pada *ranking* teratas yang tercermin dari nilai NDCG yang rendah, model *language generation* TinyLlama 1.1B masih mampu mengekstrak dan menyintesis informasi yang dibutuhkan dari dokumen-dokumen yang berhasil di-*retrieve*. Temuan ini menunjukkan efektivitas pendekatan RAG dimana kelemahan pada tahap *retrieval* dapat dikompensasi oleh kemampuan *generative model* untuk memahami konteks dari *multiple* dokumen, bukan hanya bergantung pada dokumen yang berada di posisi teratas hasil *retrieval*.

Anomali menarik ditemukan pada kategori *query scrambled* yang menunjukkan pola berlawanan dengan gap 130enyusun sebesar -0.35. Pada kategori ini, *retrieval* sangat unggul dengan nilai NDCG@5 sebesar 0.68 yang merupakan nilai tertinggi dari semua kategori, namun *generation* menunjukkan performa lemah dengan *Semantic Similarity* hanya 0.33. Kekuatan *retrieval* pada *query scrambled* dapat dijelaskan melalui karakteristik komponen BM25 dalam sistem *hybrid retrieval* yang melakukan *keyword matching* secara 130enyusun130c130 *per-term*. *Query* seperti “java komunitas mahasiswa akademik

prodi struktur" mengandung banyak kata kunci penting, dan BM25 tidak terganggu oleh urutan kata sehingga tetap menemukan dokumen relevan dengan *overlap* kata kunci yang tinggi. Kombinasi dengan FAISS yang menangkap konteks 131enyusun keseluruhan menghasilkan performa *retrieval* yang sangat baik.

Sebaliknya, kelemahan *generation* pada *query scrambled* terjadi karena *model language* TinyLlama 1.1B bergantung pada struktur 131enyusun131c untuk memahami *intent query*. *Query* tanpa struktur gramatikal yang jelas membuat model mengalami kebingungan mengenai informasi apa yang harus diekstrak dari dokumen. Model cenderung melakukan fallback ke strategi copy-paste dari dokumen tanpa melakukan sintesis yang bermakna, sehingga menghasilkan Semantic Similarity yang rendah meskipun nilai *Faithfulness* tinggi mencapai 0.68. Implikasi praktis dari temuan ini adalah untuk *query* yang tidak terstruktur atau mengandung *noise*, sistem *retrieval* sangat dapat diandalkan untuk menemukan dokumen relevan, namun pengguna mungkin perlu membaca dokumen sumber secara langsung alih-alih mengandalkan jawaban yang di-*generate* oleh sistem.

Temuan penting lainnya adalah gap terbesar terjadi pada *query* dengan 5+ kata yang mencapai +0.22, dimana *generation* jauh lebih unggul dibandingkan *retrieval*. *Query* panjang memberikan konteks lengkap yang memudahkan *model language* untuk memahami *intent* pengguna dengan lebih baik. Meskipun *retrieval* hanya menunjukkan performa moderat dengan NDCG@5 sebesar 0.25, model *generation* mampu menyintesis informasi dari *multiple* dokumen dengan baik. Hal ini terjadi karena model *language* modern termasuk TinyLlama 1.1B dilatih untuk

memproses *prompt* panjang dengan konteks yang kaya, sehingga menghasilkan *Semantic Similarity* tertinggi sebesar 0.47 pada kategori *query* ini.

4.3.2 Analisis Performa *Retrieval*

Hasil pengujian *retrieval* menunjukkan pola yang cukup menarik. Sistem *hybrid* yang menggabungkan FAISS (*semantic search*) dan BM25 (*lexical search*) berhasil mencapai *Recall*@10 sebesar 1.0278, yang artinya sistem tidak hanya menemukan dokumen *ground truth*, tapi juga dokumen relevan tambahan yang tidak terdaftar secara manual. Ini membuktikan bahwa pendekatan *hybrid* memang efektif untuk menangkap relevansi dari berbagai aspek, baik makna 132enyusun maupun kecocokan kata kunci.

Adapun yang cukup mengejutkan adalah performa terbaik justru muncul di kategori *query* acak (*scrambled*). *Query* yang tidak terstruktur seperti "java komunitas mahasiswa akademik prodi struktur" malah menghasilkan MRR@5 = 0.58 dan *Recall*@10 = 1.83, jauh lebih tinggi dari kategori lain. Ini terjadi karena BM25 bekerja dengan cara *term-independent matching*, jadi urutan kata tidak terlalu berpengaruh. Selama *keyword*-nya ada banyak dan tersebar di berbagai dokumen, sistem bisa *matching* dengan baik. FAISS juga membantu menangkap konteks 132enyusun secara keseluruhan meskipun strukturnya acak.

Sebaliknya, *query* pendek (1-2 kata) justru mengalami kesulitan di *ranking* teratas. MRR@3 untuk *query* 1 kata Cuma 0.10, yang berarti dokumen relevan jarang muncul di top-3 meskipun *Recall*@10-nya tinggi (1.20). Ini masalah ambiguitas, kata seperti "komunitas" atau "kurikulum" terlalu *general* sehingga

banyak dokumen yang sesuai, tetapi sistem kesulitan menentukan mana yang paling relevan.

Query 3-4 kata menunjukkan performa paling stabil dan *balance*. Ini bagian bagusnya karena cukup spesifik untuk mengurangi ambiguitas, tetapi tidak terlalu panjang hingga terlalu spesifik. Untuk *use case* chatbot akademik, kategori ini paling realistis karena mahasiswa biasanya bertanya dengan format seperti "visi misi prodi" atau "ketua program studi informatika".

4.3.3 Analisis Performa *Generation*

Performa *generation* secara keseluruhan berada di level moderat dengan *Semantic Similarity* dan *Answer Relevancy* masing-masing 0.38, sementara *Faithfulness* mencapai 0.55. Gap antara *faithfulness* dan *semantic similarity* ini menjadi temuan penting, artinya model dapat mengambil informasi yang benar dari dokumen (*faithful*), tapi cara menyampaikannya kurang natural dan tidak mirip dengan jawaban ideal yang ditulis manusia.

Pola ini konsisten dengan karakteristik model kecil seperti TinyLlama 1.1B. Model dengan 1.1 miliar parameter ini memang punya keterbatasan dalam memahami konteks kompleks dan menghasilkan teks yang koheren dalam bahasa Indonesia. Yang sering terjadi adalah *repetition loops*, model *stuck* mengulang-ulang frasa yang sama, atau terjadi *degenerative output* di mana jawabannya jadi tidak sesuai sama sekali.

Query 3 kata menunjukkan performa terburuk dengan *Semantic Similarity* hanya di angka 0.28. Ini 133enyusu karena seharusnya 3 kata itu optimal seperti yang terlihat di *retrieval*. Ternyata masalahnya terdapat pada *context window*

pressure. Dengan 3 dokumen masing-masing sekitar 1200 karakter ditambah *system prompt* dan *query*, total token mendekati batas 2048. TinyLlama menjadi "tertekan" dan outputnya jadi membingungkan atau bahkan *fallback* ke *template* jawaban seperti "informasi tersebut tidak tersedia dalam *database*" padahal dokumennya jelas ada.

Terdapat temuan menarik, yaitu pada *query* >5 kata justru menunjukkan skor *Semantic Similarity* tertinggi (0.47). Hipotesisnya, *query* panjang memberikan lebih banyak *clue* dan konteks yang membuat *prompt* lebih terarah. Model menjadi lebih fokus dan berhasil melakukan *generation* berdasarkan *template* dengan lebih baik. Contohnya *query* "kurikulum dan profil lulusan program studi informatika" bisa menghasilkan *Semantic Similarity* sampai 0.69.

Untuk *scrambled query*, *faithfulness*-nya bahkan paling tinggi (0.68) tetapi *semantic similarity*-nya rendah di angka (0.33). Ini terjadi karena *retrieval* berhasil mendapatkan dokumen yang sangat relevan karena *overlap* pada *keyword* tinggi, kemudian model cenderung *copy-paste* dari dokumen tanpa benar-benar memahami maksud *query*. Strategi bertahan ini sebenarnya cukup efektif untuk model kecil dan lebih baik mengutip dokumen yang benar daripada mengarang dan membuat jawaban sendiri.

4.3.4 Analisis Keterkaitan *Retrieval* dan *Generation*

Gambar 4.10 menunjukkan bahwa performa *retrieval* yang tinggi tidak selalu menjamin *generation* yang bagus. Kategori *scrambled* adalah contoh paling jelas, dimana NDCG@5 mencapai angka 0.68 (*retrieval excellent*), tetapi *Semantic*

Similarity-nya malah turun ke 0.33 (*generation poor*). Gap sebesar -0.35 ini menunjukkan *bottleneck* ada di tahap *generation*, bukan pada tahap *retrieval*.

Sebaliknya, kategori 1 kata dan 5+ kata menunjukkan peningkatan yang positif yaitu (+0.18 dan +0.22). Artinya meskipun dokumen yang di-*retrieve* belum sempurna, model *generation* berhasil "memperbaiki" *output* dengan cara 135enyusun ulang informasi dari beberapa dokumen atau menambahkan konteks dari pemahaman bahasanya sendiri.

Pola ini mengonfirmasi bahwa dalam arsitektur RAG, kedua komponen harus dioptimasi secara terpisah tapi juga dipertimbangkan sebagai satu kesatuan *pipeline*. *Retrieval* yang bagus dapat berujung sia-sia apabila *generation* lemah, dan *generation* yang kuat tetap butuh input berkualitas dari *retrieval*.

4.3.5 Limitasi dan Tantangan *Generation*

Dari seluruh data hasil pengujian dapat dilihat bahwa meskipun pada beberapa bagian *generation* memberikan hasil yang bagus tetap ditemukan beberapa kekurangan dan keterbatasan. Adapun beberapa limitasi utama yang ditemukan selama pengujian sistem adalah:

4.3.3.1 Keterbatasan Model Generatif Kecil

TinyLlama 1.1B dipilih karena pertimbangan efisiensi, bisa jalan di hardware terbatas tanpa perlu GPU *high-end*. Tetapi pilihan ini tantangannya jelas: model ini *struggle* dengan konteks panjang, sering terjadi *stuck* di *repetition loops*, dan kurang bagus dalam bahasa Indonesia. Contoh nyata adalah jawaban untuk *query* "ketua prodi" yang hasilnya "HANYA merujuk padan dengan HANYA

merujuk padan dengan..." berulang-ulang. Ini *degenerative output* yang tidak informatif sama sekali.

4.3.3.2 Context Window Limitation

Dengan maksimal 2048 tokens, sistem harus melakukan *trim* dokumen hasil *retrieval*. Untuk *query* yang butuh informasi dari banyak bagian dokumen, konteks yang diteruskan ke model jadi tidak lengkap dan sempurna. Ini terutama masalah di *query* 3-4 kata yang spesifik tapi butuh informasi detail.

4.3.3.3 *Ambiguitas Query* Pendek

Query 1-2 kata sangat ambiguous. "Komunitas" bisa merujuk ke Fun Java, GDSC, ETH0, atau komunitas lainnya. Sistem tidak punya mekanisme *clarification asking*, chatbot langsung menjawab berdasarkan dokumen top-1 yang belum tentu yang dimaksud user.

4.3.3.4 *Over-specification* di *Query* Panjang

Meskipun query 5+ kata menunjukkan *semantic similarity* tertinggi, ada risiko *over-specification*. Kalau user terlalu detail tapi dokumen tidak ada yang exact match dengan semua keyword, sistem bisa miss dokumen yang sebenarnya relevan.

4.3.3.5 Tidak Ada Mekanisme *Fallback*

Ketika model tidak yakin atau tidak menemukan informasi yang tepat, sistem tidak punya strategi *fallback* yang baik. Kadang model tetap memaksa jawab

(risiko *hallucination*), kadang *fallback* ke template "informasi tidak tersedia" padahal terdapat di dokumen.

4.3.6 Implikasi Penggunaan Model Kecil (TinyLlama 1.1B)

Keputusan untuk menggunakan TinyLlama 1.1B adalah atas dasar pertimbangan antara performa dan efisiensi. Pada konteks prodi yang *resource*-nya terbatas dan prioritasnya adalah kelayakan sistem yang bisa jalan di *local machine*, maka pilihan ini masuk akal.

Dari hasil pengujian yang telah dilakukan menunjukkan bahwa model kecil sebenarnya cukup memiliki kapasitas untuk tugas tertentu, terutama yang sifatnya *retrieval-heavy* dan tidak butuh *generation* yang terlalu kreatif. *Faithfulness* 0.55 membuktikan model bisa tetap *grounded* ke dokumen, ini kebiasaan yang diinginkan untuk pencarian informasi chatbot.

Adapun yang kemudian menjadi masalah adalah kualitas bahasa output. Model pre-trained ini kebanyakan dilatih dengan data bahasa Inggris, jadi saat *prompting* dalam bahasa Indonesia, hasilnya sering janggal atau repetitif. Kedepannya, apabila terdapat *resource* lebih, dapat dilakukan *upgrade* ke model yang lebih besar (misalnya 3B atau 7B parameter) atau model yang secara spesifik dilatih untuk bahasa Indonesia bisa dan secara signifikan dapat meningkatkan skor *semantic similarity* dan *answer relevancy*.

Tetapi harus diakui juga, untuk demonstrasi awal sistem RAG di lingkungan akademik dengan hardware terbatas, TinyLlama 1.1B sudah menunjukkan bahwa konsep RAG itu dapat bertahan dan bisa meneruskan nilai informasi meskipun dengan limitasi yang ada.

4.4 Integrasi Islam dan Maqasid Syariah

Pengembangan *Chatbot Informasi Akademik Teknik Informatika Berbasis Retrieval-Augmented Generation (RAG)* tidak hanya dinilai dari aspek teknis, tetapi juga dianalisis dari perspektif nilai-nilai Islam, khususnya dalam kerangka Maqasid Syariah. Maqasid Syariah adalah prinsip-prinsip utama yang bertujuan mewujudkan *kemaslahatan (maslahah)* dan mencegah *kerusakan (mafsadah)* dalam kehidupan manusia. Lima tujuan utama (al-kulliyat al-khams) yang relevan dalam penelitian ini adalah:

1. *Hifz al-'Aql* (Menjaga Akal)
2. *Hifz al-'Ilm* (Menjaga dan Mengembangkan Ilmu)
3. *Hifz al-Nafs* (Menjaga Jiwa dan Kedisiplinan Mahasiswa)
4. *Hifz al-Din* (Integritas, kejujuran, dan etika pemanfaatan pengetahuan)

Integrasi nilai-nilai tersebut memberikan dasar filosofis dan moral dalam pengembangan teknologi di lingkungan UIN Maulana Malik Ibrahim Malang, sehingga inovasi digital tidak lepas dari orientasi kebermanfaatan dan tanggung jawab etis.

4.2.6 *Hifz al-'Aql* (Menjaga dan Mengoptimalkan Akal)

Penelitian ini bertujuan membantu mahasiswa memperoleh informasi akademik dengan cepat, akurat, dan terstruktur. Sistem chatbot berbasis RAG mengurangi *cognitive overload* mahasiswa saat mencari informasi manual pada website yang strukturnya kompleks. Allah berfirman:

هَلْ يَسْتَوِى الَّذِيْنَ يَعْلَمُوْنَ وَالَّذِيْنَ لَا يَعْلَمُوْنَۗ

"...apakah sama orang-orang yang mengetahui (hak-hak Allah) dengan orang-orang yang tidak mengetahui (hak-hak Allah)?" (QS. Az-Zumar: 9)

Pada NU Online dijelaskan bahwa ayat ini menegaskan pentingnya memudahkan akses terhadap ilmu pengetahuan. Chatbot ini menghilangkan hambatan informasi dan membantu mahasiswa menggunakan akal secara optimal dalam proses akademik. Adapun relevansinya dengan sistem adalah sebagai berikut:

- Relevansi dengan sistem
- Menyediakan informasi akademik tanpa ambiguitas.
- Mengurangi kebingungan akibat informasi tercecer.
- Mendukung proses belajar dan pengambilan keputusan akademik.

4.2.7 *Hifz al-'Ilm* (Menjaga Ilmu dan Keaslian Informasi)

Salah satu isu krusial dalam era AI adalah disinformasi dan halusinasi model. Dengan menerapkan *Retrieval-Augmented Generation*, penelitian ini mengarahkan AI untuk selalu menjawab berdasarkan dokumen resmi prodi, bukan berdasarkan spekulasi.

عَنْ الْمُغِيرَةِ رَضِيَ اللَّهُ عَنْهُ قَالَ سَمِعْتُ النَّبِيَّ صَلَّى اللَّهُ عَلَيْهِ وَسَلَّمَ يَقُولُ إِنَّ كَذِبًا عَلَيَّ لَيْسَ كَكَذِبٍ عَلَى أَحَدٍ مَنْ كَذَبَ عَلَيَّ مُتَعَمِّدًا فَلْيَتَبَوَّأْ مَقْعَدَهُ مِنْ النَّارِ

'Dari al-Mughirah Radhiyallahu anhu, dia berkata, "Aku mendengar Rasûlullâh Shallallahu alaihi wa sallam bersabda, "Sesungguhnya berdusta atasku tidak seperti berdusta atas orang yang lain. Barangsiapa berdusta atasku dengan sengaja, maka hendaklah dia mengambil tempat tinggalnya di neraka"'. [HR. Al-Bukhâri, no. 1229]

(HR. Al-Bukhâri dalam *Shahih Bukhari Muslim (Al-Lu'lu' Wal Marjan)*, Kitab al-Janâiz: 23, Kitab Jenazah, bab ke-34, bab hal-hal yang dibenci dari meratapi orang yang telah meninggal dunia, hadis no. 4) (Fuad Abdul Baqi, 2017).

Hadis ini menjadi landasan etis untuk mencegah penyampaian informasi palsu atau mengada-ada (taḥrīf). Dalam konteks teknologi informasi, hadis ini menuntut sistem berbasis AI untuk tidak mengarang jawaban (hallucination), tidak menyebarkan data akademik yang salah, dan hanya menyampaikan informasi berdasarkan sumber yang valid dan terverifikasi. Prinsip kejujuran ini sejalan dengan konsep ṣidq (kejujuran) dan amānah (amanah) dalam Islam yang mewajibkan setiap penyampai informasi untuk memastikan kebenaran dan akurasi pesan yang disampaikan.

Relevansi hadis ini dengan sistem chatbot RAG yang dikembangkan tercermin dalam tiga aspek implementasi. Pertama, instruksi sistem (system prompt) pada TinyLlama menetapkan larangan tegas untuk menjawab tanpa merujuk pada konteks dokumen yang di-retrieve, dengan perintah eksplisit "JANGAN MENGARANG" sebagai guardrail untuk mencegah hallucination. Kedua, setiap jawaban yang dihasilkan sistem disertai source attribution berupa URL dokumen sumber yang memungkinkan pengguna melakukan verifikasi langsung, menjamin transparansi dan akuntabilitas informasi. Ketiga, metrik faithfulness dalam evaluasi sistem mengukur tingkat groundedness jawaban terhadap dokumen konteks, dengan hasil 0.549 yang mengkonfirmasi bahwa mayoritas jawaban tetap setia pada sumber dan tidak mengada-ada informasi yang tidak terdapat dalam dokumen. Dengan demikian, sistem tidak hanya memenuhi

standar teknis akurasi informasi, tetapi juga mengimplementasikan nilai kejujuran Islam dalam penyampaian informasi akademik kepada mahasiswa.

4.2.8 *Hifz al-Nafs* (Menjaga Kenyamanan, Ketenangan, dan Kesiapan Mahasiswa)

Mahasiswa sering mengalami stres ketika harus mencari informasi penting menjelang PKLI, KRS, remedial, skripsi, dan seminar hasil. Chatbot ini membantu mengurangi tekanan psikologis karena informasi menjadi lebih mudah ditemukan, tersedia kapan pun, dan tidak lagi bergantung pada jam kantor atau admin. Allah berfirman:

....يُرِيْدُ اللّٰهُ بِكُمُ الْيُسْرَ وَلَا يُرِيْدُ بِكُمُ الْعُسْرَۖ

"Allah menghendaki kemudahan bagimu dan tidak menghendaki kesukaran" (QS. Al-Baqarah: 185).

Dikutip dari NU Online, dari ayat tersebut, Allah itu senantiasa memudahkan hambanya di setiap keadaan yang dihadapi oleh hambanya. Dari sini dapat dipahami bahwa sistem chatbot juga sejalan dengan konsep memudahkan dimana kehadiran sistem chatbot dapat menghemat waktu pencarian informasi yang sebelumnya memakan puluhan menit, mengurangi potensi miskomunikasi antara mahasiswa dan pihak prodi, dan menjaga ketenangan jiwa mahasiswa dalam menjalani proses akademik.

4.2.9 *Hifz al-Din* (Integritas, Kejujuran, dan Etika Teknologi)

Integritas adalah aspek penting dalam pengembangan teknologi, khususnya di institusi Islam. Penelitian ini menjaga nilai-nilai kejujuran akademik: model

dilarang membuat data yang tidak ada, dimana sistem diarahkan untuk berkata "tidak tahu" jika konteks tidak tersedia, dan seluruh proses pengembangan dilakukan dengan niat memberikan kemaslahatan bagi civitas akademika.

يَٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُواْ ٱتَّقُواْ ٱللَّهَ وَقُولُواْ قَوْلًا سَدِيدًا

"Wahai orang-orang yang beriman, bertakwalah kepada Allah dan berkatalah dengan perkataan yang benar." (QS. Al-Ahzab: 70).

Rasulullah juga bersabda:

عَنْ تَمِيمٍ الدَّارِيِّ أَنَّ النَّبِيَّ صَلَّى اللهُ عَلَيْهِ وَسَلَّمَ، قَالَ: «الدِّينُ النَّصِيحَةُ» قُلْنَا: لِمَنْ؟ قَالَ: «لِلَّهِ وَلِكِتَابِهِ وَلِرَسُولِهِ وَلِأَئِمَّةِ الْمُسْلِمِينَ وَعَامَّتِهِمْ»

Bersumber dari Tamim Ad-Dari bahwa Nabi SAW bersabda, "Agama adalah nasihat." Kami (sahabat Nabi) bertanya, "Untuk siapa?" Beliau menjawab, "Untuk Allah, Kitab, Rasul, para pemimpin muslimin dan mereka secara umum." Hadits ini juga diriwayatkan oleh Bukhari, Abu Dawud, Tirmidzi, Nasa'i, Syafi'i, Ahmad, Darimi, Ibnu Hibban, Thabrani dan masih ada yang lainnya.

Hadis Rasulullah ﷺ yang menyatakan bahwa *"Agama adalah nasihat"* (**الدِّينُ النَّصِيحَةُ**) memberikan landasan moral bahwa setiap bentuk interaksi, termasuk dalam ranah teknologi, seyogianya dilandasi kejujuran, integritas, serta komitmen untuk menghadirkan manfaat yang benar bagi umat. Ketika para sahabat menanyakan kepada siapa nasihat itu ditujukan, Rasulullah ﷺ menegaskan bahwa nasihat mencakup hubungan dengan Allah, Kitab-Nya, Rasul-Nya, para pemimpin, dan seluruh masyarakat muslim. Dengan demikian, makna nasihat dalam hadis ini tidak hanya terbatas pada tutur lisan, tetapi juga mencakup penyampaian informasi yang akurat, bertanggung jawab, serta menghindari penyimpangan dalam bentuk apa pun.

Dalam konteks penelitian ini, prinsip nasihat tersebut menjadi relevan ketika diterapkan pada sistem chatbot akademik berbasis Retrieval-Augmented Generation (RAG). Hal ini tampak melalui beberapa aspek berikut:

1. Chatbot sebagai sarana nasihat digital yang memberikan informasi benar bagi mahasiswa.

   Sebagaimana nasihat harus disampaikan dengan ketulusan dan kebenaran, chatbot dirancang untuk menyampaikan informasi akademik yang tepat, berbasis sumber resmi prodi. Hal ini membantu mahasiswa memperoleh pemahaman yang benar mengenai prosedur akademik, sehingga teknologi berfungsi sebagai bentuk "nasihat digital" yang selaras dengan nilai kejujuran dalam Islam.
2. Menghindari manipulasi informasi dan menjaga prinsip kejujuran ilmiah. Hadis tersebut menekankan pentingnya amanah dan kejujuran dalam menyampaikan informasi. Implementasi RAG pada chatbot memastikan bahwa jawaban diambil dari sumber yang valid, bukan hasil halusinasi model. Pendekatan ini sejalan dengan nilai *Hifz al-Dīn*, yaitu menjaga kemurnian nilai kebenaran, menghindari manipulasi data, serta menegakkan integritas dalam pelayanan informasi akademik.
3. Menjadi contoh penerapan etika Islam dalam teknologi modern. Ketika teknologi dikembangkan dengan menjunjung tinggi nilai ketelitian, kejujuran, dan kemanfaatan bagi masyarakat kampus, hal ini mencerminkan bentuk aktualisasi nasihat yang disebutkan dalam hadis. Chatbot bukan hanya

alat teknis, tetapi juga wujud komitmen etis untuk membantu civitas akademika dengan cara yang bertanggung jawab dan bernilai ibadah.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini berhasil merancang dan mengimplementasikan sistem chatbot berbasis *Retrieval-Augmented Generation* (RAG) untuk layanan informasi akademik Program Studi Teknik Informatika UIN Maulana Malik Ibrahim Malang. Sistem terdiri dari tiga komponen utama: tahap pengumpulan data yang mengekstrak 384 potongan dokumen (*chunk*) dari 350 halaman website prodi, sistem pencarian informasi menggunakan gabungan metode pencarian semantik dan pencarian kata kunci dengan bobot 0.25, serta sistem pembangkit jawaban menggunakan model bahasa TinyLlama 1.1B dengan kapasitas pemrosesan 2048 token. Implementasi menggunakan arsitektur layanan web berbasis *Application Programming Interface* (API) memungkinkan sistem berjalan mandiri pada perangkat dengan spesifikasi minimal (prosesor 4 core, memori 8GB) tanpa bergantung pada layanan komersial berbayar.

Evaluasi komprehensif menggunakan 30 pertanyaan untuk pengujian pencarian (*retrieval*)dan 30 pertanyaan untuk pengujian pembangkit jawaban (*generation*) pada 6 kategori panjang pertanyaan mengungkap kinerja sistem yang beragam. Komponen pencarian informasi mencatat nilai kualitas peringkat sebesar 0.258 dan tingkat penemuan dokumen sebesar 1.028, dengan kinerja terbaik pada pertanyaan acak (nilai 0.68) yang menunjukkan ketahanan sistem terhadap variasi struktur pertanyaan. Komponen pembangkit jawaban mencatat kemiripan semantik (*Semantic Similarity*) 0.381 dan *Faithfulness* terhadap dokumen 0.549, dengan

kinerja optimal pada pertanyaan 5+ kata (0.472) dan titik lemah pada pertanyaan 3 kata (0.284). *Faithfulness* yang konsisten tinggi mengkonfirmasi model tetap mengacu pada dokumen sumber dan jarang mengada-ada informasi. Dari analisis perbandingan menunjukkan *generation* unggul di 5 dari 6 kategori dengan selisih positif 0.08 hingga 0.22, memvalidasi efektivitas pendekatan RAG dalam meningkatkan kinerja melalui penggabungan informasi dari berbagai dokumen.

Pengujian konsistensi dengan mengulang pertanyaan yang sama sebanyak 5 kali menghasilkan temuan penting bahwa sistem mampu menghasilkan jawaban yang identik dengan nilai variasi nol untuk pertanyaan faktual kritis yang telah dilindungi dengan mekanisme pengaman. Hal ini memvalidasi bahwa strategi mitigasi yang diimplementasikan berhasil mengatasi ketidakkonsistenan output yang diamati pada saat sidang. Limitasi yang teridentifikasi mencakup pengulangan kalimat sekitar 20%, kebocoran template instruksi sekitar 30%, dan kesulitan ekstraksi informasi spesifik seperti nama dan jabatan, yang merupakan karakteristik umum model bahasa berukuran kecil dengan keterbatasan pelacakan konteks. Limitasi ini berhasil dimitigasi melalui penyaringan pasca-pemrosesan dan solusi cadangan untuk pertanyaan kritis.

Sistem membuktikan kelayakan pendekatan RAG dengan model berukuran kecil untuk aplikasi spesifik domain di institusi pendidikan, memberikan kontribusi praktis berupa solusi aksesibilitas informasi 24/7 dengan biaya minimal serta kontribusi metodologis berupa identifikasi kondisi optimal (pertanyaan 5+ kata) dan titik lemah (pertanyaan 3 kata) yang dapat menjadi acuan penelitian serupa di masa depan.

5.2 Saran

Berdasarkan hasil penelitian, beberapa saran diajukan untuk pengembangan sistem dan penelitian lanjutan di bidang chatbot RAG untuk pendidikan:

1. Melakukan upgrade model *generation* menjadi prioritas utama untuk mengatasi limitasi yang teridentifikasi. TinyLlama 1.1B dengan *semantic similarity* 0.38 dan frekuensi repetition loops 20% menunjukkan keterbatasan signifikan model kecil. Penelitian selanjutnya dapat menggunakan model berkapasitas lebih besar seperti Llama-3-8B atau Qwen-7B yang memiliki kapasitas pemrosesan 4096+ token dan dukungan Bahasa Indonesia lebih baik, diperkirakan dapat meningkatkan kemiripan semantik menjadi 0.55-0.65 dan mengurangi keluaran yang bermasalah hingga di bawah 5%. Alternatif hemat biaya adalah penyesuaian model menggunakan teknik efisien dengan dataset spesifik akademik UIN Malang untuk meningkatkan kinerja tanpa melatih ulang seluruh model.
2. Optimasi tahap *preprocessing* dan *chunking* sangat krusial namun sering diabaikan. Penelitian ini menggunakan *chunking fixed-size* 150 karakter yang dapat memotong informasi penting di tengah paragraf atau kalimat. Penelitian selanjutnya dapat mengeksplorasi *semantic chunking* yang mempertimbangkan batas kalimat, paragraf, atau bagian dokumen, serta pengayaan metadata untuk setiap potongan dengan informasi seperti tanggal publikasi, kategori konten, dan hierarki struktur dokumen. Strategi pemotongan yang lebih baik dapat meningkatkan relevansi pencarian sebesar 15-20% dan mengurangi fragmentasi konteks yang menyebabkan kesulitan saat *generation* pada pertanyaan 3 kata

3. Implementasi mekanisme perangkingan ulang seperti (*reranker*) seperti jina-reranker-v2-base-multilingual sebagai tahap ketiga setelah pencarian semantik dan pencarian kata kunci dapat meningkatkan kualitas peringkat sebesar 15-25%, terutama untuk kategori pertanyaan 2-3 kata yang saat ini mencatat nilai rendah (0.20). Mekanisme ini melakukan penilaian yang lebih presisi dengan mempertimbangkan interaksi semantik antara pertanyaan dan dokumen, lebih *powerful* dibanding penilaian dua arah yang digunakan pada pencarian semantik.
4. Pembangunan pembangunan graf pengetahuan atau basis data terstruktur untuk menyimpan informasi faktual seperti nama dosen, jabatan, jadwal akademik, dan mata kuliah dapat meningkatkan akurasi pertanyaan faktual mendekati 95-100%. Pendekatan gabungan antara pencarian faktual terstruktur dan RAG untuk pertanyaan penjelasan dapat mengombinasikan presisi tinggi dengan fleksibilitas pemahaman bahasa natural.
5. Untuk *deployment production* sebaiknya dilengkapi *monitoring dashboard* untuk *tracking* pola pertanyaan (*query patterns*), rata-rata kemiripan semantik per kategori, frekuensi kesalahan, dan waktu respons. Mekanisme umpan balik dari pengguna penting untuk pengembangan berkelanjutan melalui pembelajaran aktif, di mana pertanyaan dengan umpan balik negatif diprioritaskan untuk diperbaiki baik melalui pembaruan basis pengetahuan maupun penyesuaian model.
6. Penelitian lanjutan dapat melakukan studi ablasi dengan membandingkan kinerja sistem pencarian murni, *generation* murni, dan RAG lengkap untuk

mengukur kontribusi setiap komponen secara eksplisit. Studi pengguna dengan mahasiswa sebagai pengguna akhir menggunakan skala kegunaan sistem juga diperlukan untuk mengukur kepuasan dan tingkat adopsi aktual yang tidak terukur melalui evaluasi offline.

Implementasi saran-saran ini secara bertahap dapat mengembangkan sistem menjadi solusi yang lebih mature untuk aksesibilitas informasi akademik di institusi pendidikan, sekaligus memberikan kontribusi metodologis bagi penelitian RAG di domain pendidikan Indonesia.

DAFTAR PUSTAKA

Abudrrohman R. (2024). *Uji Performa Chatbot Dengan Retrieval Augmented Generation Dan Model Gpt-4 Untuk Domain Taharah Berdasarkan Empat Imam Mazhab Fikih (Studi Kasus Kitab Rahmah Al Ummah Fi Ikhtilaf Al a'Immah)*.

Adamopoulou, E., & Moussiades, L. (2023). An Overview of Chatbot Technology Eleni. In *Pakistan Journal of Medical Sciences* (Vol. 39, Issue 2). Springer International Publishing. https://doi.org/10.12669/pjms.39.2.7653

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., & Jégou, H. (2025). *The Faiss library*. http://arxiv.org/abs/2401.08281

Elysia, S., Persada, D., & alan Taman Malaka Selatan No, J. (2024). Chatbot Berbasis Retrieval Augmented Generation (RAG) untuk Peningkatan Layanan Informasi Sekolah. *Jurnal Tifda*, *1*(2), 52–58. https://doi.org/10.70491/tifda.v1i2.52

Fuad Abdul Baqi, F. (2017). *SHAHIH BUKHARI MUSLIM (AL-LU'LU' WAL MARJAN)* (A. Firly Bassam Taqiy (ed.); 1st ed.). PT Elex Media Komputindo. http://pustaka-indo.blogspot.com

Guo, J., Fan, Y., Pang, L., Yang, L., Ai, Q., Zamani, H., Wu, C., Croft, W. B., & Cheng, X. (2020). A Deep Look into neural ranking models for information retrieval. *Information Processing and Management*, *57*(6). https://doi.org/10.1016/j.ipm.2019.102067

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. T. (2020). Dense passage retrieval for open-domain question answering. *EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 6769–6781. https://doi.org/10.18653/v1/2020.emnlp-main.550

Klesel, M., & Wittmann, H. F. (2025). Retrieval-Augmented Generation (RAG). In *Business and Information Systems Engineering*. Springer. https://doi.org/10.1007/s12599-025-00945-3

Knollmeyer, S., Caymazer, O., & Grossmann, D. (2025). Document GraphRAG: Knowledge Graph Enhanced Retrieval Augmented Generation for Document Question Answering Within the Manufacturing Domain. *Electronics (Switzerland)*, *14*(11). https://doi.org/10.3390/electronics14112102

Krisnawati, L. D., Mahastama, A. W., Haw, S. C., Ng, K. W., & Naveen, P. (2024). Indonesian-English Textual Similarity Detection Using Universal Sentence Encoder (USE) and Facebook AI Similarity Search (FAISS). *CommIT Journal*, *18*(2), 183–195. https://doi.org/10.21512/commit.v18i2.11274

Labadze, L., Grigolia, M., & Machaidze, L. (2024). Role of AI chatbots in

education : systematic literature review. *International Journal of Educational Technology in Higher Education*, *2023*, 1–17. https://doi.org/10.1186/s41239-023-00426-1

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W. T., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, *2020-Decem*.

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(4), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

Mcgrath, C., Farazouli, A., & Cerratto, T. (2025). Generative AI chatbots in higher education : a review of an emerging research area. *Higher Education*, *89*(6), 1533–1549. https://doi.org/10.1007/s10734-024-01288-w

Nur'aini, I. (2024). Sistem Chatbot Sebagai Layanan Informasi Kesehatan Mental Pada Remaja Menggunakan Metode Large Language Model (Llm). *Doctoral Dissertation, Universitas Islam Sultan Agung Semarang*, *15*(1), 37–48.

Nur Hakim, A. A., Murti, A. C., & Nindyasari, R. (2025). Implementasi Artificial Intelligence Dalam Sistem Pencarian Orang Hilang Dengan Face Recognition Studi Kasus Polres Kudus. *SKANIKA: Sistem Komputer Dan Teknik Informatika*, *8*(1), 168–180. https://doi.org/10.36080/skanika.v8i1.3334

Oreški, D., & Vlahek, D. (2024). Retrieval Augmented Generation in Large Language Models: Development of AI Chatbot for Student Support. In *CEUR Workshop Proceedings* (Vol. 3938, pp. 12–23). researchgate.net. https://www.researchgate.net/profile/Dino-Vlahek/publication/391918806_Retrieval_Augmented_Generation_in_Large_Language_Models_Development_of_AI_Chatbot_for_Student_Support/links/682d8a896b5a287c3042ef28/Retrieval-Augmented-Generation-in-Large-Language-Mo

Patil, R., Boit, S., Gudivada, V., & Nandigam, J. (2023). A Survey of Text Representation and Embedding Techniques in NLP. *IEEE Access*, *11*(March), 36120–36146. https://doi.org/10.1109/ACCESS.2023.3266377

Prasetyo, E. A. (2024). *Chatbot untuk Informasi Pembangunan Wilayah Kota Semarang menggunakan Metode Retrieval Augmented Generation (RAG)*. http://ecampus.poltekkes-medan.ac.id/jspui/handle/123456789/1726

Prastowo, I., Putro, H., Antoni, J., Adhitya, M. K., & Herawati, N. A. (2025). *Retrieval-Augmented Generation (RAG) Chatbot for Handling Customer Complaints in the Energy Sector*. *10*(2), 105–111.

Pratama, I. ichsanudin rachman, & Sisephaputra, B. (2024). Pengembangan Sistem Helpdesk Menggunakan Chatbot Dengan Metode Retrieval-augmented

Generation (Rag). *Journal of Informatics and Computer Science (JINACS)*, *6*(03), 696–710. https://doi.org/10.26740/jinacs.v6n03.p696-710

Pujiono, I., Agtyaputra, I. M., & Ruldeviyani, Y. (2024). Implementing Retrieval-Augmented Generation and Vector Databases for Chatbots in Public Services Agencies Context. *JITK (Jurnal Ilmu Pengetahuan Dan Teknologi Komputer)*, *10*(1), 216–223. https://doi.org/10.33480/jitk.v10i1.5572

Putro, I. P. H., Antoni, J., Adhitya, M. K., & ... (2025). Retrieval-Augmented Generation (RAG) Chatbot for Handling Customer Complaints in the Energy Sector. *Jurnal Infomedia ….* https://e-jurnal.pnl.ac.id/infomedia/article/view/7169

Ramadhan, T. I., Supriatman, A., & Kurniawan, T. R. (2024). Passage Retrieval untuk Question Answering Bahasa Indonesia Menggunakan BERT dan FAISS. *Jurnal Algoritma*, *21*(2), 156–163. https://doi.org/10.33364/algoritma/v.21-2.2100

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 3982–3992. https://doi.org/10.18653/v1/d19-1410

Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. In *Foundations and Trends in Information Retrieval* (Vol. 3, Issue 4). https://doi.org/10.1561/1500000019

Salsabila, S. K. (2025). *INTEGRASI CHATBOT BERBASIS LARGE LANGUANGE MODEL*.

Samudra, G., Zy, A. T., & Ermanto. (2025). Implementasi Retrieval Augmented Generation (RAG) Dalam Perancangan Chatbot Kesehatan Pencernaan. *JSAI (Journal Scientific and Applied Informatics)*, *8*(1), 181–188. https://doi.org/10.36085/jsai.v8i1.7678

Septri, V. (2025). *Rancang bangun chatbot menggunakan large language model sebagai sarana informasi skripsi skripsi*.

Shiri, A. (2004). Introduction to Modern Information Retrieval (2nd edition). *Library Review*, *53*(9), 462–463. https://doi.org/10.1108/00242530410565256

Siino, M., Tinnirello, I., & La Cascia, M. (2024). Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers. *Information Systems*, *121*(July 2023), 102342. https://doi.org/10.1016/j.is.2023.102342

Soliman, H., Kotte, H., Kravčík, M., Pengel, N., & Duong-Trung, N. (2025). Retrieval-Augmented Chatbots for Scalable Educational Support in Higher Education. *CEUR Workshop Proceedings*, *3994*(March), 22–31.

Stohr, C., Ou, A. W., & Malmstrom, H. (2024). *Computers and Education : Artificial Intelligence Perceptions and usage of AI chatbots among students in higher education across genders , academic levels and fields of study*. *7*(August 2023), 0–11. https://doi.org/10.1016/j.caeai.2024.100259

Sugiarto, R. W., Sokibi, P., Rizkiyah, P., Informatika, T., Informasi, T., Catur, U., & Cendekia, I. (2025). *Chatbot Layanan Akademik Calon Mahasiswa UCIC Menggunakan Metode Retrieval-Augmented Generation Retrieval : Mengambil dokumen relevan dari basis data menggunakan pencocokan vektor* . 246–249.

Vaswani, A., SHazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, L. (2017). Retention Is All You Need. *International Conference on Information and Knowledge Management, Proceedings*, *Nips*, 4752–4758. https://doi.org/10.1145/3583780.3615497

Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., & Wei, F. (2024). *Multilingual E5 Text Embeddings: A Technical Report*. http://arxiv.org/abs/2402.05672

Zhang, P., Zeng, G., Wang, T., & Lu, W. (2024). *TinyLlama: An Open-Source Small Language Model*. 1–10. http://arxiv.org/abs/2401.02385

Nahdlatul Ulama. (2025). Tafsir Tahlili – Quran Online NU. Diakses pada 2 Oktober 2025, dari https://quran.nu.or.id/

Airbyte. (n.d.). Tokenization vs. embeddings: Key differences explained. Airbyte. Retrieved October 2, 2025, from https://airbyte.com/data-engineering-resources/tokenization-vs-embeddings

LAMPIRAN-LAMPIRAN

Tabel A.1 Dataset Query untuk Pengujian Retrieval

ID	Kategori	Query	Ground Truth URL
Q1	1 kata	komunitas	https://informatika.uin-malang.ac.id/fun-java/ https://informatika.uin-malang.ac.id/etho/ https://informatika.uin-malang.ac.id/google-developer-student-club-dsc/ https://informatika.uin-malang.ac.id/data-science-enthusiast-dsc/
Q2	1 kata	kurikulum	https://informatika.uin-malang.ac.id/curriculum/
Q3	1 kata	dosen	https://informatika.uin-malang.ac.id/lecturer-and-staff/
Q4	1 kata	prestasi	https://informatika.uin-malang.ac.id/1st-winner-of-national-hackathon-competition-on-bug-bounty-2022/ https://informatika.uin-malang.ac.id/1st-winner-of-madinah-van-java-pencak-silat-championship/
Q5	1 kata	laboratorium	https://informatika.uin-malang.ac.id/undergraduate-s1/
Q6	2 kata	fun java	https://informatika.uin-malang.ac.id/fun-java/
Q7	2 kata	visi misi	https://informatika.uin-malang.ac.id/undergraduate-s1/
Q8	2 kata	profil prodi	https://informatika.uin-malang.ac.id/undergraduate-s1/
Q9	2 kata	eth0 komunitas	https://informatika.uin-malang.ac.id/etho/
Q10	2 kata	ketua prodi	https://informatika.uin-malang.ac.id/lecturer-and-staff/ https://informatika.uin-malang.ac.id/undergraduate-s1/
Q11	3 kata	kurikulum informatika uin	https://informatika.uin-malang.ac.id/curriculum/
Q12	3 kata	komunitas fun java	https://informatika.uin-malang.ac.id/fun-java/

ID	Kategori	Query	Ground Truth URL
Q13	3 kata	struktur organisasi prodi	https://informatika.uin-malang.ac.id/undergraduate-s1/
Q14	3 kata	sejarah prodi informatika	https://informatika.uin-malang.ac.id/undergraduate-s1/
Q15	3 kata	laboratorium intelligent system	https://informatika.uin-malang.ac.id/undergraduate-s1/
Q16	4 kata	persyaratan kelulusan informatika uin	https://informatika.uin-malang.ac.id/curriculum/
Q17	4 kata	ketua program studi informatika	https://informatika.uin-malang.ac.id/lecturer-and-staff/ https://informatika.uin-malang.ac.id/undergraduate-s1/
Q18	4 kata	tujuan pembelajaran program studi	https://informatika.uin-malang.ac.id/undergraduate-s1/
Q19	4 kata	mata kuliah wajib informatika	https://informatika.uin-malang.ac.id/curriculum/
Q20	4 kata	agenda kegiatan komunitas mahasiswa	https://informatika.uin-malang.ac.id/coordination-with-student-communities/ https://informatika.uin-malang.ac.id/fun-java/ https://informatika.uin-malang.ac.id/etho/
Q21	5+ kata	struktur organisasi jurusan teknik informatika uin	https://informatika.uin-malang.ac.id/undergraduate-s1/
Q22	5+ kata	mata kuliah wajib mahasiswa informatika uin	https://informatika.uin-malang.ac.id/curriculum/
Q23	5+ kata	persyaratan kelulusan program studi teknik informatika	https://informatika.uin-malang.ac.id/curriculum/
Q24	5+ kata	hubungan komunitas fun java dengan himpunan mahasiswa	https://informatika.uin-malang.ac.id/fun-java/ https://informatika.uin-malang.ac.id/coordination-with-student-communities/

ID	Kategori	Query	Ground Truth URL
Q25	5+ kata	kurikulum dan profil lulusan program studi informatika	https://informatika.uin-malang.ac.id/curriculum/ https://informatika.uin-malang.ac.id/undergraduate-s1/
Q26	Scrambled	java komunitas mahasiswa akademik prodi struktur	https://informatika.uin-malang.ac.id/fun-java/ https://informatika.uin-malang.ac.id/undergraduate-s1/
Q27	Scrambled	jurusan akademik sks kuliah komunitas random	https://informatika.uin-malang.ac.id/curriculum/ https://informatika.uin-malang.ac.id/undergraduate-s1/ https://informatika.uin-malang.ac.id/fun-java/
Q28	Scrambled	komunitas informatika acak jumbled kata tidak urut	https://informatika.uin-malang.ac.id/fun-java/ https://informatika.uin-malang.ac.id/etho/ https://informatika.uin-malang.ac.id/google-developer-student-club-dsc/
Q29	Scrambled	kurikulum prodi informatika random kata tidak relevan	https://informatika.uin-malang.ac.id/curriculum/
Q30	Scrambled	fun java organisasi mahasiswa acak tidak jelas	https://informatika.uin-malang.ac.id/fun-java/

Tabel A.2 Dataset Query dan Reference Answer untuk Pengujian Generation

ID	Kategori	Query	Reference Answer
G1	1 kata	komunitas	Program Studi Informatika memiliki berbagai komunitas minat seperti Fun Java, ETH0, MOCAP, GDSC, dan DSE yang membantu mahasiswa mengembangkan kompetensi sesuai bidangnya.
G2	1 kata	kurikulum	Kurikulum Informatika UIN Malang memuat mata kuliah dasar informatika, mata kuliah keislaman, serta mata kuliah keahlian seperti pemrograman, basis data, jaringan komputer, dan kecerdasan buatan.

ID	Kategori	Query	Reference Answer
G3	1 kata	dosen	Dosen Program Studi Informatika terdiri dari tenaga pendidik profesional dengan keahlian di bidang pemrograman, jaringan, basis data, multimedia, sistem cerdas, serta bidang-bidang terkait lainnya..
G4	1 kata	laboratorium	Program Studi Informatika memiliki laboratorium yang mendukung pembelajaran seperti Laboratorium Programming, Intelligent System, Multimedia, dan Network Security
G5	1 kata	prestasi	Mahasiswa Informatika UIN Malang telah meraih berbagai prestasi di tingkat nasional maupun internasional dalam kompetisi pemrograman, robotika, dan hackathon.
G6	2 kata	fun java	Fun Java adalah komunitas pemrograman berorientasi objek yang membina mahasiswa dalam dasar-dasar Java dan menjadi fondasi bagi komunitas keahlian lainnya di Informatika UIN Malang.
G7	2 kata	visi misi	Visi Program Studi Informatika adalah menjadi program studi yang integratif dalam mengintegrasikan ilmu pengetahuan dan nilai-nilai Islam serta berdaya saing internasional. Misi nya adalah menghasilkan lulusan yang berkarakter ulul albab dan mengembangkan ilmu pengetahuan teknologi informasi yang relevan.
G8	2 kata	profil prodi	Program Studi Teknik Informatika UIN Maulana Malik Ibrahim Malang merupakan program sarjana di bawah Fakultas Sains dan Teknologi yang telah terakreditasi Unggul oleh LAM INFOKOM dan berfokus pada pengembangan teknologi informasi terintegrasi dengan nilai-nilai Islam.
G9	2 kata	ketua prodi	Ketua Program Studi Teknik Informatika UIN Malang adalah Supriyono, M.Kom.
G10	2 kata	eth0 komunitas	ETH0 adalah komunitas yang berfokus pada Linux dan open source untuk mengembangkan kemampuan mahasiswa di bidang sistem operasi dan keamanan jaringan.

ID	Kategori	Query	Reference Answer
G11	3 kata	kurikulum informatika uin	Kurikulum Informatika UIN Malang terdiri dari mata kuliah wajib prodi, mata kuliah pilihan, dan mata kuliah keislaman yang tersusun dalam 8 semester dengan total 144 SKS.
G12	3 kata	komunitas fun java	Komunitas Fun Java berfokus pada pembelajaran Object-Oriented Programming menggunakan bahasa Java untuk membekali mahasiswa dengan kemampuan dasar pemrograman yang kuat.
G13	3 kata	struktur organisasi prodi	Struktur organisasi Program Studi Informatika terdiri dari ketua program studi, sekretaris program studi, dosen, koordinator laboratorium, serta staff administrasi pendukung.
G14	3 kata	sejarah prodi informatika	Program Studi Informatika UIN Malang berkembang sebagai bagian dari Fakultas Sains dan Teknologi untuk memenuhi kebutuhan pendidikan teknologi informasi berbasis nilai-nilai Islami.
G15	3 kata	laboratorium intelligent system	Laboratorium Intelligent System fokus pada penelitian dan pembelajaran kecerdasan buatan, machine learning, dan sistem cerdas lainnya.
G16	4 kata	persyaratan kelulusan informatika uin	Kelulusan mahasiswa Informatika mensyaratkan pemenuhan 144 SKS kurikulum, penyelesaian Kerja Praktik, Tugas Akhir, serta memenuhi beban akademik dan administratif lainnya.
G17	4 kata	ketua program studi informatika	Ketua Program Studi Teknik Informatika UIN Maulana Malik Ibrahim Malang adalah Supriyono, M.Kom.
G18	4 kata	mata kuliah wajib informatika	Mata kuliah wajib Informatika mencakup pemrograman dasar, struktur data, basis data, jaringan komputer, rekayasa perangkat lunak, dan kecerdasan buatan.
G19	4 kata	tujuan pembelajaran program studi	Tujuan pembelajaran Program Studi Informatika adalah menghasilkan lulusan yang kompeten di bidang teknologi informasi, memiliki karakter Islami, dan mampu bersaing di tingkat global.

ID	Kategori	Query	Reference Answer
G20	4 kata	agenda kegiatan komunitas mahasiswa	Kegiatan komunitas mahasiswa meliputi workshop pemrograman, seminar teknologi, kompetisi IT, pelatihan keahlian, dan kegiatan pengembangan kompetensi lainnya.
G21	5+ kata	struktur organisasi jurusan teknik informatika uin	Struktur organisasi Program Studi Teknik Informatika UIN Malang meliputi ketua prodi Supriyono M.Kom, sekretaris prodi Shoffin Nahwa Utama M.T, dosen, koordinator laboratorium, dan staf administrasi.
G22	5+ kata	mata kuliah wajib mahasiswa informatika uin	Mata kuliah wajib mahasiswa Informatika UIN Malang mencakup pemrograman, algoritma, struktur data, basis data, jaringan komputer, rekayasa perangkat lunak, kecerdasan buatan, dan mata kuliah keislaman.
G23	5+ kata	persyaratan kelulusan program studi teknik informatika	Persyaratan kelulusan Program Studi Teknik Informatika adalah menyelesaikan 144 SKS, menempuh Kerja Praktik minimal 1 bulan, menyelesaikan Tugas Akhir, serta memenuhi syarat akademik dan administratif lainnya.
G24	5+ kata	hubungan komunitas fun java dengan himpunan mahasiswa	Komunitas Fun Java berkoordinasi dengan Himpunan Mahasiswa Jurusan untuk menyelenggarakan kegiatan pengembangan kompetensi pemrograman dan mendukung aktivitas akademik mahasiswa Informatika.
G25	5+ kata	kurikulum dan profil lulusan program studi informatika	Kurikulum Program Studi Informatika dirancang untuk menghasilkan lulusan dengan profil sebagai Software Engineer, Data Scientist, Network Engineer, dan IT Consultant yang kompeten dan berakhlak mulia.
G26	Scrambled	java komunitas mahasiswa akademik prodi struktur	Informatika memiliki komunitas seperti Fun Java yang fokus pada pengembangan kemampuan pemrograman Java bagi mahasiswa dan terintegrasi dengan struktur akademik prodi.
G27	Scrambled	jurusan akademik sks kuliah komunitas random	Program Studi Informatika memiliki kurikulum berbasis SKS yang terdiri dari mata kuliah dasar informatika, keahlian, dan kegiatan komunitas sebagai sarana pengembangan mahasiswa.

ID	Kategori	Query	Reference Answer
G28	Scrambled	komunitas informatika acak jumbled kata tidak urut	Komunitas di Informatika meliputi Fun Java, MOCAP, ETH0, GDSC, dan DSE yang membantu mahasiswa mengembangkan kompetensi sesuai minat masing-masing.
G29	Scrambled	kurikulum prodi informatika random kata tidak relevan	Kurikulum Program Studi Informatika disusun secara terstruktur dengan mata kuliah wajib, pilihan, dan kegiatan pendukung untuk mencapai standar kompetensi lulusan.
G30	Scrambled	fun java organisasi mahasiswa acak tidak jelas	Fun Java merupakan komunitas resmi Program Studi Informatika yang mengelola kegiatan pembelajaran pemrograman Java untuk mahasiswa.