Terbit *online* pada laman : http://teknosi.fti.unand.ac.id/

**Jurnal Nasional Teknologi dan Sistem Informasi**

| ISSN (Print) 2460-3465 | ISSN (Online) 2476-8812 |

Artikel Peneltian

# Analisis Komparatif Kinerja Llama Murni, Rag Native, dan Rag Fine-Tuning

*Nindya Desyana* [a], *Ardytha Luthfiarta* [b*]

[a,b] *Universitas Dian Nuswantoro, Jalan Imam Bonjol No. 207, Kota Semarang Kode Pos. 50131, Indonesia*

INFORMASI ARTIKEL

*Sejarah Artikel:*
Diterima Redaksi: 29 Oktober 2025
Revisi Akhir: 18 Desember 2025
Diterbitkan *Online*: 28 Desember 2025

KATA KUNCI

Chatbot Llama,
Retrieval-Augmented-Generation (RAG),
Langchain,
Fine-tuningrag

KORESPONDENSI

E-mail: ardytha.luthfiarta@dsn.dinus.ac.id*

**A B S T R A C T**

Penelitian ini mengatasi masalah halusinasi pada Large Language Models (LLM) seperti LLaMA ketika menjawab kueri domain spesifik. Tujuan penelitian adalah membandingkan kinerja tiga arsitektur chatbot: LLaMA murni**,** Retrieval-Augmented Generation (RAG) berbasis LangChain (RAG Native**)**, dan RAG Fine-Tuning. Implementasi RAG pada penelitian ini menggunakan framework LangChain sebagai sistem penghubung antara model LLaMA dan sumber pengetahuan eksternal (vector database). Framework ini menyediakan pipeline *retriever-reader* yang memungkinkan integrasi antara model bahasa dan data kontekstual melalui embedding serta pencarian vektor. Metode evaluasi dilakukan secara kuantitatif menggunakan metrik ROUGE-L dan BLEU pada dataset studi kasus. Hasil penelitian menunjukkan peningkatan kinerja yang progresif: arsitektur LLaMA murni (baseline) memperoleh skor ROUGE-L sebesar 46.48, implementasi RAG Native (LangChain) meningkat menjadi 61.42, dan model RAG Fine-Tuning (LangChain Optimized) mencapai kinerja tertinggi dengan skor 83.06**.** Penelitian ini menyimpulkan bahwa integrasi arsitektur RAG melalui framework LangChain secara signifikan meningkatkan akurasi respons chatbot, dan proses fine-tuning pada konfigurasi RAG merupakan langkah optimasi krusial untuk mencapai performa terbaik pada domain spesifik.

## 1. PENDAHULUAN

Perkembangan *Large Language Models* (LLM) [1] telah mentransformasi kapabilitas pemrosesan bahasa alami. Model-model ini, yang dilatih pada data tekstual skala masif, berfungsi sebagai mesin generasi teks yang canggih, mampu menjawab pertanyaan, dan menjadi fondasi bagi agen konversasional (chatbot) yang adaptif dan interaktif. Namun, di balik kemampuan impresif tersebut, LLM memiliki keterbatasan fundamental yang inheren, yakni ketergantungannya pada data pelatihan yang bersifat statis. Kondisi ini menyebabkan LLM rentan menghasilkan keluaran yang tidak faktual, sebuah fenomena yang dikenal luas sebagai halusinasi, terutama saat dihadapkan pada kueri yang bersifat spesifik domain atau memerlukan data terkini.

Sebagai solusi dominan untuk mengatasi keterbatasan ini, arsitektur *Retrieval-Augmented Generation* (RAG) telah diadopsi secara luas. Pendekatan RAG, sebagaimana pertama kali diperkenalkan oleh Lewis et al. [2], secara fundamental melandaskan proses generasi respons pada sumber pengetahuan eksternal. Dalam penelitian ini, implementasi dasar RAG yang disebut sebagai RAG *Native* dikembangkan secara manual tanpa menggunakan *framework* pihak ketiga seperti LangChain. Arsitektur *native* ini mengimplementasikan *pipeline retriever-reader* standar, di mana sistem mengambil dokumen relevan dari basis data vektor dan menyajikannya secara langsung sebagai konteks bagi LLM untuk menghasilkan jawaban.

Meskipun implementasi RAG Native (tanpa *framework*) terbukti memberikan peningkatan kinerja dibanding model murni, arsitektur dasar ini seringkali masih beroperasi pada level suboptimal. Literatur terkini, seperti dalam berbagai artikel survei[3], telah mengidentifikasi kesenjangan ini dengan mengkategorikan implementasi RAG menjadi *Naive RAG* (setara dengan RAG Native dalam penelitian ini) dan *Advanced RAG*. Kategori *Advanced RAG* berfokus pada optimasi *pipeline retrieval* untuk mengatasi kegagalan dalam relevansi konteks.

https://doi.org/10.25077/TEKNOSI.v11i3.2025.263-272

Sejalan dengan konsep *Advanced RAG*, penelitian ini mengimplementasikan arsitektur kedua yang disebut RAG *Fine-Tuning*. Berbeda dengan RAG Native, arsitektur ini secara spesifik memanfaatkan *framework* LangChain untuk orkestrasi *pipeline* yang lebih terstruktur dan kompleks. Penggunaan LangChain [4] memungkinkan implementasi teknik optimasi lanjutan, sejalan dengan praktik yang diusulkan dalam literatur [5], yang menekankan pentingnya modul *reranking* atau *query expansion* untuk meningkatkan presisi *retrieval*. Dalam studi ini, *RAG Fine-Tuning* didefinisikan sebagai optimalisasi *system-level* yang difasilitasi LangChain, mencakup *Query Processing*, *Multi-Source Data Retrieval*, *Advanced Vector Search [6]* dengan *Intelligence Scoring*, serta *Response Validation Layer*.

Untuk mengatasi kendala tersebut, penelitian ini mengusulkan pendekatan *RAG Fine-Tuning*, yaitu optimalisasi arsitektur RAG melalui penyesuaian konfigurasi *pipeline* tanpa melakukan *model fine-tuning* langsung terhadap LLM. Pendekatan ini mencakup beberapa aspek, seperti *Query Processing & Topic Filtering*, *Multi-Source Data Retrieval*, *Advanced Vector Search [6]* dengan *Intelligence Scoring*, serta penerapan *Anti-Hallucination System* dan *Response Validation Layer*. Optimalisasi ini diimplementasikan menggunakan *framework LangChain*, yang berfungsi sebagai pengelola *pipeline* antara *retriever*, *vector store*, dan *model inference* dari *Ollama*. Melalui *LangChain*, sistem dapat melakukan *text chunking*, *query expansion*, dan *context aggregation* secara otomatis, sehingga meningkatkan relevansi hasil pencarian terhadap kueri pengguna.

Studi ini difokuskan pada kasus nyata, yaitu pengembangan *chatbot* layanan informasi untuk Televisi Universitas Dian Nuswantoro (*TVKU)*, yang menggunakan sumber data statis (file *JSON* tentang profil, layanan, dan media sosial *TVKU*) serta data dinamis (*API TVKU* yang menyediakan informasi aktual). Tiga arsitektur dibandingkan secara kuantitatif, yaitu:

1. *Llama murni (baseline)* — tanpa *retrieval* eksternal.
2. *Llama + RAG Native* — dengan *pipeline* dasar tanpa *framework*.
3. *Llama + RAG Fine-Tuning* — dengan optimasi menggunakan *framework* LangChain.

Dengan demikian, penelitian ini tidak hanya mengonfirmasi bahwa RAG lebih baik daripada *baseline*, tetapi secara spesifik memberikan bukti empiris—menggunakan metrik ROUGE-L dan BLEU—mengenai dampak krusial dari optimasi *pipeline* RAG yang terstruktur dalam meningkatkan akurasi faktual, melampaui apa yang dicapai oleh arsitektur RAG Native.

## 2. METODE

Untuk mengevaluasi kinerja secara komprehensif, penelitian ini membandingkan tiga arsitektur model yang berbeda. Model 1**,** Llama Murni (*Baseline*), berfungsi sebagai titik acuan dasar, mewakili kinerja LLM (*Large Language Model*) tanpa augmentasi data eksternal. Model 2, Llama + RAG Native, mengimplementasikan arsitektur *Retrieval-Augmented Generation* (RAG) di mana *pipeline* (seperti pemroses kueri, *retriever*, dan LLM) dibangun dan dihubungkan secara manual dari dasar (*from scratch*). Model 3, Llama + RAG Fine-Tuning, juga memanfaatkan RAG namun dirancang untuk alur kerja yang lebih kompleks dan teroptimasi.

Pada penelitian ini proses pengembangan, *prototyping* awal, dan implementasi model dilakukan pada *local device*. Perangkat keras yang digunakan untuk pengembangan ini adalah sebuah MacBook Pro (13-inch, 2020) yang dilengkapi dengan prosesor 2 GHz Quad-Core Intel Core i5, memori 16 GB 3733 MHz LPDDR4X, dan grafis terintegrasi Intel Iris Plus Graphics 1536 MB, yang beroperasi pada sistem operasi macOS 14.0. Lingkungan lokal ini digunakan untuk iterasi dan pembangunan *framework* aplikasi. Namun, untuk memastikan objektivitas, standarisasi sumber daya komputasi, dan menghindari bias kinerja perangkat, proses evaluasi kuantitatif terhadap ketiga arsitektur model (Llama Murni, RAG Native, dan RAG Fine-Tuning) dilakukan pada *cloud* Google Colaboratory (Colab). Penggunaan Google Colab memungkinkan alokasi sumber daya (CPU/GPU) yang konsisten dan terukur untuk setiap eksekusi model terhadap dataset evaluasi, sehingga perbandingan skor metrik ROUGE-L [7], [8] dan BLEU[9] yang dihasilkan bersifat adil dan dapat direproduksi.

Dari sisi perangkat lunak, arsitektur sistem dibangun menggunakan tumpukan teknologi modern. Bahasa pemrograman utama yang digunakan adalah TypeScript, yang dipilih karena kapabilitas *static type-checking* untuk menjaga integritas kode, sementara Node.js berfungsi sebagai *runtime* sisi server. Antarmuka pengguna dibangun menggunakan React 18, dengan Next.js 14 (App Router) dimanfaatkan sebagai *framework full-stack* untuk mengelola *server-side rendering* dan integrasi rute API.

Untuk *engine* inferensi model bahasa, penelitian ini memanfaatkan Ollama[10]. Justifikasi pemilihan Ollama adalah kemampuannya untuk menyederhanakan proses *deployment* dan eksekusi LLM *open-source* secara lokal. Ollama membungkus kompleksitas *setup* inferensi dan menyediakan *endpoint* API yang stabil, sehingga memungkinkan Llama 3 8B [10], [11] versi terkuantisasi (Q4_0) dapat berjalan secara efisien pada perangkat keras pengembangan yang telah disebutkan. Pendekatan ini krusial untuk memfasilitasi iterasi cepat dan menguji kelayakan arsitektur pada lingkungan komputasi *consumer-grade*.

Dua *library* spesialis menjadi krusial dalam implementasi RAG. Pertama, ChromaDB dipilih sebagai basis data vektor (*vector database*) [1], [12]. Alasan utama pemilihannya adalah sifatnya yang *open-source* dan arsitekturnya yang berfokus pada kemudahan pengembang (*developer-friendly*), dirancang untuk implementasi lokal yang cepat. Berbeda dengan solusi *vector store* berskala produksi seperti Milvus [1], [12], [13] yang dirancang untuk skalabilitas horizontal dan *throughput* tinggi dalam lingkungan terdistribusi, arsitektur *standalone* ChromaDB [12] lebih ideal untuk skenario penelitian dan purwarupa. ChromaDB memungkinkan *setup* minimal dan reproduktifitas yang mudah, yang sangat sesuai dengan fokus penelitian ini pada perbandingan *pipeline*, bukan pada *benchmark* skalabilitas infrastruktur.

Kedua, *framework* LangChain [4] digunakan secara eksklusif untuk orkestrasi arsitektur Model 3 (RAG Fine-Tuning). LangChain berfungsi sebagai *framework* abstraksi modular yang

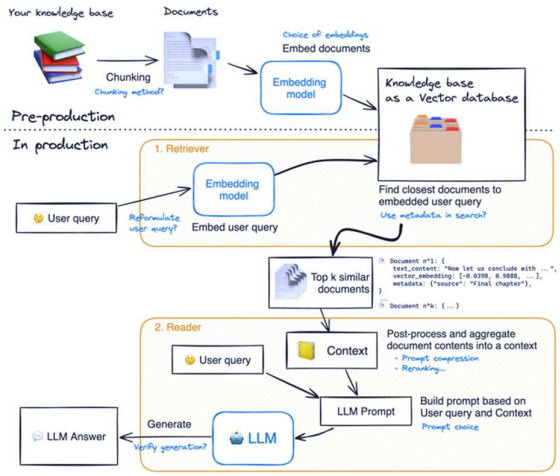

https://doi.org/10.25077/TEKNOSI.v11i3.2025.263-272

menyatukan berbagai komponen *pipeline* RAG. Justifikasi penggunaannya adalah untuk memfasilitasi desain *pipeline* yang kompleks; alih-alih membangun interkoneksi manual antara pemroses kueri, *retriever*, dan LLM (seperti yang dilakukan pada Model 2 RAG Native), LangChain menyediakan "rantai" (*chains*) dan agen yang telah teruji. Penggunaan LangChain memungkinkan penelitian ini untuk fokus pada desain logis dan optimasi alur kerja RAG yang canggih, alih-alih pada kompleksitas implementasi *low-level* dari setiap interaksi komponen.

Penelitian ini menggunakan metodologi kuantitatif eksperimental untuk membandingkan kinerja dari tiga arsitektur chatbot yang berbeda. Tahapan penelitian mencakup perancangan arsitektur, implementasi system, pengumpulan data evaluasi, dan analisis kinerja menggunakan metrik standar.

## *2.1. Arsitektur Model*

Tiga arsitektur model diimplementasikan dan dibandingkan dalam penelitian ini. Ketiganya menggunakan model dasar LLaMA namun dengan konfigurasi *retrieval* yang berbeda.

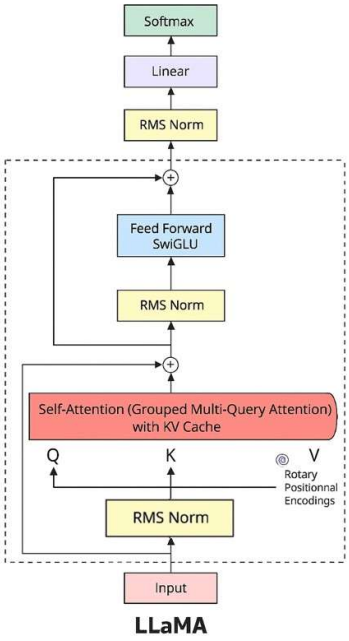



Gambar 1. Arsitektur Chatbot

### *2.1.1. Model 1: Llama Murni (*Baseline*)*

Model bahasa dasar (LLM) yang digunakan sebagai model generatif inti dalam ketiga konfigurasi arsitektur adalah LLaMA 3 seperti terlihat pada gambar 2. Model ini merupakan generasi ketiga dari famili model bahasa Llama yang dikembangkan oleh Meta AI dan dirilis secara terbuka untuk komunitas riset serta komersial. Pemilihan Llama 3 didasarkan pada kinerjanya yang *state-of-the-art* di antara model *open-source* yang tersedia pada saat penelitian ini dilakukan, yang menawarkan kapabilitas pemahaman konteks dan generasi teks yang lebih unggul (superior) dibandingkan pendahulunya (seperti Llama 2 [14]). Varian spesifik yang diimplementasikan adalah LLaMA 3 8B[10], yakni model dengan 8 miliar parameter. Justifikasi utama pemilihan varian 8B—dibandingkan varian yang lebih besar seperti 70B—adalah kesesuaiannya dengan tujuan penelitian, yaitu menguji arsitektur yang efisien dan dapat diimplementasikan pada perangkat keras *consumer-grade*, sebagaimana telah diperinci pada spesifikasi lingkungan implementasi (MacBook Pro 16 GB RAM).

Untuk mendukung tujuan efisiensi tersebu**t,** model ini tidak dijalankan dalam mode parameter penuh (*full parameter*, misalnya FP16 atau BF16) yang membutuhkan alokasi memori sekitar 16 GB hanya untuk bobot model. Sebaliknya, model diinferensi menggunakan versi terkuantisasi dengan format Q4_0, yang dimuat melalui *framework* Ollama. Kuantisasi Q4_0 (kuantisasi 4-bit) secara signifikan mengurangi jejak memori (*memory footprint*) model menjadi sekitar 4-5 GB. Pendekatan kuantisasi ini krusial untuk mencapai efisiensi sumber daya. Format Q4_0 dipilih karena merupakan standar industri yang menawarkan keseimbangan optimal antara reduksi ukuran model secara agresif dan degradasi kinerja (akurasi) yang m inimal, sehingga memungkinkan inferensi lokal yang efektif pada perangkat keras dengan VRAM/RAM terbatas.




Gambar 2. Arsitektur LLaMA

### *2.1.2. Model 2: Llama + RAG Native*

Arsitektur yang diimplementasikan pada penelitian ini merupakan Native Retrieval-Augmented Generation (RAG), yaitu sistem generatif yang menggabungkan proses pencarian informasi (*retrieval*) dan pembangunan jawaban (*generation*) secara langsung tanpa menggunakan framework pihak ketiga seperti LangChain. Pendekatan ini dipilih karena memiliki keunggulan dalam hal kinerja, efisiensi sumber daya, serta fleksibilitas dalam pengendalian logika sistem.

Secara umum, RAG berfungsi untuk meningkatkan akurasi *language model* dengan menambahkan tahap pengambilan konteks dari sumber eksternal sebelum proses generasi dilakukan. Dengan demikian, model tidak hanya mengandalkan data pelatihan internal, tetapi juga dapat mengakses informasi terkini

dari *knowledge base*. Arsitektur Native RAG yang diimplementasikan terdiri dari beberapa komponen utama, yaitu:

1. User Query Interface Pengguna mengajukan pertanyaan atau perintah melalui antarmuka chatbot pada situs TVKU. Pertanyaan yang diterima dikirim ke server *backend* untuk diproses lebih lanjut
2. Topic Filtering Module Modul ini berfungsi untuk memeriksa apakah pertanyaan yang diajukan relevan dengan domain TVKU. Pertanyaan di luar topik (seperti cuaca, hiburan umum, atau hal yang tidak berkaitan dengan televisi kampus) akan ditolak dengan pesan penolakan otomatis. Tahapan ini penting untuk membatasi ruang lingkup model agar hanya beroperasi pada domain tertentu (*domain-specific constraint*).
3. Embedding Generation [15] (Ollama API) Pertanyaan pengguna diubah menjadi representasi vektor menggunakan model nomic-embed-text yang dijalankan melalui API Ollama. Model ini menghasilkan embedding berdimensi tinggi yang merepresentasikan makna semantik dari teks input.
4. Vector Retrieval (ChromaDB) Embedding hasil konversi digunakan untuk melakukan pencarian semantic [16] (*semantic similarity search*) pada basis data vektor ChromaDB, yang berisi potongan dokumen (*document chunks*) terkait profil, program, jadwal acara, dan berita TVKU. Proses ini menghasilkan beberapa dokumen paling relevan berdasarkan nilai *cosine similarity*.
5. Dynamic Context Construction Selain hasil pencarian dari ChromaDB, sistem juga melakukan pemanggilan data dinamis dari API resmi TVKU secara *real-time on-demand* seperti https://apidev.tvku.tv/api/berita,https://apidev.tvku.tv/api/program, dan https://apidev.tvku.tv/api/jadwal. Data hasil pemanggilan ini digabungkan dengan konteks statis dari basis pengetahuan untuk membentuk konteks akhir yang kaya informasi.
6. Response Generation (LLM via Ollama) Seluruh konteks (baik statis maupun dinamis) disusun dalam format prompt terstruktur dan dikirim ke Large Language Model (Llama 3) melalui endpoint /api/generate pada Ollama. Model kemudian menghasilkan jawaban yang faktual, kontekstual, dan sesuai dengan gaya bahasa chatbot TVKU.
7. Output Layer Hasil akhir jawaban ditampilkan kembali kepada pengguna melalui antarmuka web, dengan opsi dukungan suara (*voice-to-text*) sebagai fitur tambahan untuk meningkatkan interaktivitas.

Metrik *cosine similarity* dalam konteks ini berfungsi untuk mengkuantifikasi tingkat relevansi semantik antara dua representasi vektor dalam hal ini, vektor kueri pengguna $(\vec{q})$ dan vektor *document chunk* $(\vec{d})$ di dalam ruang vektor berdimensi tinggi. Justifikasi pemilihan *cosine similarity* [17]dibandingkan metrik jarak lain (seperti *Euclidean distance*) [18] adalah keunggulannya dalam menangani data tekstual. *Cosine similarity* berfokus murni pada orientasi (sudut) antar vektor dan mengabaikan magnitudo (panjang) vektor. Hal ini krusial, karena dalam pemrosesan bahasa alami, magnitudo vektor dapat sangat bervariasi (misalnya, kueri yang pendek versus dokumen yang panjang), namun orientasi vektorlah yang secara efektif merepresentasikan kesamaan topik atau makna semantik.

Secara matematis, *cosine similarity* [17] menghitung nilai kosinus $(\theta)$ dari sudut yang terbentuk di antara kedua vektor tersebut. Perhitungan ini dilakukan dengan mengambil hasil perkalian titik (*dot product*) dari dua vektor, kemudian membagi hasil tersebut dengan perkalian dari magnitudo (panjang atau norma $L_2$) masing-masing vektor. Skor yang dihasilkan berkisar antara -1 (berlawanan total) hingga 1 (identik total), di mana skor yang lebih mendekati 1 menunjukkan relevansi semantik yang lebih tinggi. Rumus *cosine similarity* adalah sebagai berikut :

$$\text{Cosine} = \frac{\left(\overrightarrow{(q)} \cdot \overrightarrow{(d)}\right)}{\left(|\overrightarrow{(q)}||\overrightarrow{(d)}|\right)} \quad (1)$$

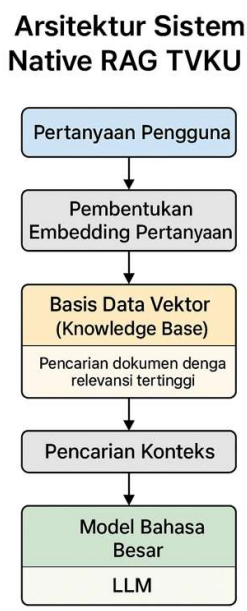
Gambar 3 memaparkan arsitektur Model 2: Llama + RAG Native, yang mewakili implementasi RAG manual (*from scratch*) dalam penelitian ini. Model ini sengaja dirancang tanpa *framework* orkestrasi (seperti LangChain) untuk dua tujuan utama: pertama, untuk mengukur dampak kinerja fundamental dari RAG (dibandingkan Model 1 Baseline), dan kedua, untuk berfungsi sebagai titik perbandingan (*benchmark*) terhadap arsitektur terorkestrasi canggih (Model 3).

Alur kerja sistem ini adalah sebagai berikut:

1. Proses dimulai ketika Pertanyaan Pengguna diterima.
2. Kueri ini segera ditransformasi melalui Pembentukan Embedding Pertanyaan untuk menghasilkan representasi vektor.
3. Vektor kueri kemudian digunakan untuk menantang Basis Data Vektor (*Knowledge Base*), yang diimplementasikan menggunakan ChromaDB. Proses ini melakukan Pencarian dokumen dengan relevansi tertinggi untuk mengidentifikasi pengetahuan faktual yang paling sesuai dari data TVKU.
4. Hasil dari Pencarian Konteks (dokumen yang relevan) diambil dan digabungkan secara manual dengan kueri pengguna asli untuk membentuk *prompt* yang diperkaya.
5. Terakhir, gabungan teks ini (konteks + kueri) dikirim ke Model Bahasa Besar (LLM), yaitu Llama 3, yang menghasilkan jawaban akhir yang didasarkan pada konteks yang diberikan.

Setiap panah dalam diagram ini mewakili interkoneksi komponen yang dibangun secara *native* dalam kode aplikasi.


https://doi.org/10.25077/TEKNOSI.v11i3.2025.263-272




Gambar 3. Arsitektur LLaMA + RAG Native

*2.1.3. Model 3: Llama + RAG Fine-Tuning*

Arsitektur ketiga merupakan pengembangan dari *Retrieval-Augmented Generation* (RAG) dengan pendekatan *system-level tuning*, bukan *weight fine-tuning* pada model LLM. Pada arsitektur ini, model *Llama 3* tetap digunakan dalam bentuk *pre-trained*, sedangkan yang dioptimasi adalah komponen-komponen sistem RAG yang mengelilinginya.

Optimasi dilakukan pada tahap *retrieval*, *reranking*, *validation*, dan *prompt configuration* untuk meningkatkan efisiensi sistem dalam menangani data domain TVKU. Framework *LangChain* digunakan untuk mengatur alur kerja *retrieval* dan *context construction*, sementara *ChromaDB* berfungsi sebagai *vector store* untuk penyimpanan hasil *embedding* dari proses *chunking* dokumen statis. Proses tuning dilakukan melalui tiga lapisan utama:

1. RAG Architecture Tuning meliputi *multi-source data retrieval* (API, JSON, dan *vector store*), *query expansion* dengan sinonim, algoritma *custom re-ranking* berbasis *intelligence scoring*, serta sistem *anti-hallucination* dan *topic filtering* yang dirancang khusus untuk domain TVKU.
2. RAG Configuration Tuning mengatur parameter penting seperti jumlah hasil maksimal (*maxResults* = 12), ambang relevansi (*relevanceThreshold* = 0.6), *diversity search* (*MMR* = true), serta *confidence threshold* untuk validasi jawaban otomatis.
3. Ollama Model Parameter Tuning Menyesuaikan parameter inferensi seperti *temperature* (0.3), *topK* (10), *topP* (0.5), dan *numCtx* (4096) guna mengoptimalkan akurasi tanpa melakukan *fine-tuning* pada *weight* model.

Secara umum, alur kerja *RAG Fine-Tuning* dapat dijelaskan sebagai berikut:

1. Pengguna mengirimkan *query* melalui antarmuka chatbot.
2. *Query* difilter oleh modul *topic filtering* untuk memastikan relevansi dengan domain TVKU.
3. *LangChain Retriever* melakukan perluasan *query* (melalui sinonim atau parafrasa) dan melakukan pencarian ke *ChromaDB* serta *API TVKU*.
4. Hasil pencarian diberi skor ulang menggunakan algoritma *intelligence scoring* untuk memilih konteks paling relevan.
5. Lapisan *validation* melakukan verifikasi fakta dan penyaringan *hallucination*.
6. Konteks yang lolos validasi disusun menjadi *prompt terstruktur* dan dikirim ke *Llama 3* melalui *Ollama API*.
7. Hasil jawaban diolah pada *response layer* dan ditampilkan kepada pengguna.

Penetapan nilai spesifik untuk parameter-parameter tersebut tidak dilakukan secara arbitrer, melainkan didasarkan pada metodologi yang menggabungkan praktik terbaik dari literatur dan pengujian empiris yang disesuaikan dengan domain penelitian.

Pada Ollama Model Parameter Tuning, nilai temperature (0.3) diatur rendah secara intensional untuk memprioritaskan konsistensi faktual (determinisme) dan mengurangi probabilitas halusinasi, yang merupakan syarat krusial untuk chatbot domain spesifik. Selain itu, parameter *sampling* Top-K (10) dan Top-P (0.5) juga disesuaikan. Top-K adalah teknik yang membatasi pilihan *token* berikutnya hanya pada K (dalam hal ini, 10) *token* dengan probabilitas tertinggi. Sementara itu, Top-**P**, yang juga dikenal sebagai *Nucleus Sampling* , secara dinamis membatasi pilihan pada himpunan *token* terkecil yang jumlah probabilitas kumulatifnya melebihi nilai P (dalam hal ini, 0.5). Konfigurasi topK=10 dan topP=0.5 ini dipilih sebagai kombinasi optimal untuk model Llama 3 8B, yang bertujuan menyaring *token* berkualitas rendah (*noise*) sambil tetap mempertahankan akurasi dan menghindari keluaran yang terlalu acak atau tidak koheren. Nilai numCtx (4096) ditetapkan agar sesuai dengan batas maksimal *context window* yang didukung oleh versi model Llama 3 yang digunakan.

Untuk RAG Configuration Tuning, nilai-nilai tersebut sebagian besar merupakan hasil dari pengujian empiris pada *dataset* TVKU. maxResults (12) ditetapkan sebagai titik keseimbangan (*trade-off*) optimal antara cakupan konteks (*contextual coverage*) dan latensi pemrosesan. Nilai ini dianggap memadai untuk menampung hasil dari *query expansion* tanpa membebani *context window* 4096 token. Relevance Threshold (0.6) diidentifikasi sebagai "sweet spot" selama validasi pengujian menunjukkan bahwa ambang batas yang lebih rendah (misalnya, 0.5) memasukkan terlalu banyak *noise* (informasi tidak relevan), sedangkan ambang batas yang lebih tinggi (misalnya, 0.7) terbukti terlalu restriktif. Nilai 0.6 ini merepresentasikan keseimbangan presisi-*recall* terbaik untuk *vector store* pada domain TVKU. Penggunaan diversity search (MMR = true) juga merupakan keputusan intensional untuk memastikan bahwa 12 hasil yang diambil tidak hanya relevan tetapi juga beragam, sehingga memberikan konteks yang lebih kaya bagi LLM.

Tabel 1. Komponen dan Fokus Optimasi pada Arsitektur RAG Fine-Tuning

| Komponen | Fokus Optimasi |
|---|---|
| Query Processing | Penyaringan & perluasan kueri |
| Multi-Source Retrieval | Integrasi API, JSON, dan *Vector Store* |
| Vector Search | *Re-ranking & relevance scoring* |
| Validation Layer | *Anti-hallucination* & verifikasi fakta |
| Ollama Configuration | Parameter inferensi model |
| RAG Configuration | Tuning parameter *retrieval* & context |

Setelah dilakukan identifikasi terhadap komponen-komponen kunci serta fokus optimasi pada arsitektur RAG Fine-Tuning sebagaimana telah dirinci pada Tabel 1, interkoneksi dan alur kerja antar komponen tersebut kemudian divisualisasikan secara komprehensif dalam bentuk diagram arsitektur sistem pada Gambar 4. Diagram dibawah mengilustrasikan alur pemrosesan sekuensial yang dimulai dari penerimaan pertanyaan pengguna, dilanjutkan dengan tahap pemrosesan dan penyaringan kueri untuk memastikan relevansi topik dan melakukan *parsing* semantik. Hasilnya kemudian digunakan untuk melakukan pencarian pada basis data vektor, diikuti dengan tahap pencarian konteks untuk menentukan fragmen informasi yang paling relevan terhadap *prompt*. Sebelum generasi respons, sistem melewati lapisan anti-halusinasi untuk validasi dan *cross-checking*, dan akhirnya, konteks yang telah divalidasi digunakan dalam tahap konfigurasi model Ollama dan penyetelan *prompt* untuk menghasilkan jawaban akhir.


Gambar 4. Arsitektur LLaMA + RAG Fine-Tuning

## 2.2. *Dataset dan Skenario Evaluasi*

Untuk mengevaluasi kinerja sistem Retrieval-Augmented Generation (RAG) pada lingkungan TVKU, digunakan sebuah dataset internal yang disusun secara terstruktur dan teridiri atas 10 pertanyaan uji yang dirancang untuk mencakup berbagai . topik seputar TVKU. Dataset ini berisi data tekstual yang berasal dari beberapa sumber utama, yaitu:

1. Dokumen statis, meliputi:
   a. *tentangTVKU.json* (informasi profil lembaga)
   b. *Company_Profile_TVKU_2025_web.pdf* (profil resmi perusahaan)
   c. *Draft Struktur Organisasi 2025.pdf* (struktur organisasi terbaru)
2. Sumber dinamis (API), meliputi:
   a. https://apidev.tvku.tv/api/berita – data berita terkini
   b. https://apidev.tvku.tv/api/program – data program telev
   c. https://apidev.tvku.tv/api/jadwal – data jadwal acara.

Dataset ini disusun dalam bentuk *knowledge base* yang mencakup berbagai topik seputar TVKU seperti visi dan misi, program acara, rate card, jadwal siaran, serta berita terbaru. Setiap entri data direpresentasikan sebagai pasangan *pertanyaan–jawaban ideal* (*question–ground truth*) yang digunakan untuk mengevaluasi keluaran sistem. Dataset evaluasi kemudian dibagi ke dalam tiga file CSV terpisah, yaitu:

1. evaluasi_llama_saja.csv – hasil keluaran sistem non-RAG (LLM murni),
2. evaluasi_rag_native.csv – hasil keluaran LangChain-RAG dengan arsitektur *retrieval* standar,
3. evaluasi_rag_tuned.csv – hasil keluaran Native-RAG dengan optimasi *system-level tuning*.

Masing-masing berkas berisi 10 entri pertanyaan yang sama, namun dengan perbedaan pada kolom Answer dan Context sesuai hasil keluaran dari tiap model. Format struktur dataset dijaga tetap konsisten dengan empat kolom utama: struktur dataset evaluasi hasil pengujian model

Tabel 2. Struktur Dataset Evaluasi Sistem RAG TVKU

| Kolom | Keterangan |
|---|---|
| Question | Pertanyaan dari pengguna |
| Answer | Jawaban yang dihasilkan model |
| Contexts | Potongan dokumen (chunks) |
| Ground_truth | Jawaban ideal |

## 2.3. *Metrik Evaluasi*

Kinerja model dievaluasi secara kuantitatif menggunakan dua metrik standar dalam evaluasi teks, yaitu ROUGE dan BLEU. Kedua metrik ini dipilih karena mampu mengukur tingkat kesamaan semantik dan struktur antara jawaban yang dihasilkan model (*answer*) dengan jawaban acuan (*ground truth*).

### *2.3.1. ROUGE (Recall-Oriented Understudy for Gisting Evaluation)*

Metrik ROUGE, yang pertama kali diperkenalkan oleh Chin-Yew Lin [7], adalah seperangkat metrik yang berfungsi untuk mengevaluasi *automatic summarization* (peringkasan otomatis) dan *machine translation* (penerjemahan mesin). Metrik ini berfokus pada pengukuran kelengkapan informasi (*recall*) dengan cara menghitung tingkat tumpang tindih (*overlap*) *n-gram*


https://doi.org/10.25077/TEKNOSI.v11i3.2025.263-272

antara teks yang dihasilkan (kandidat) dan satu atau lebih teks referensi (jawaban ideal).

*Library* evaluasi standar (rouge_score) yang digunakan dalam penelitian ini menghitung tiga skor. Pertama adalah ROUGE-L Recall ($R_{lcs}$), yang mengukur seberapa banyak referensi yang ditangkap oleh kandidat:

$$R_{lcs} = \frac{LCS(X,Y)}{m} \quad (2)$$

Kedua adalah ROUGE-L Precision ($P_{lcs}$), yang mengukur seberapa banyak kandidat yang relevan dengan referensi:

$$P_{lcs} = \frac{LCS(X,Y)}{n} \quad (3)$$

Ketiga adalah ROUGE-L F1-Score ($F_{lcs}$), yang merupakan rata-rata harmonik dari *Recall* dan *Precision*. Skor $F_{lcs}$) inilah yang digunakan sebagai metrik perbandingan utama dalam penelitian ini karena memberikan skor tunggal yang seimbang:

$$F_{lcs} = \frac{2 \cdot P_{lcs} \cdot R_{lcs}}{P_{lcs} + R_{lcs}} \quad (4)$$

Dalam penelitian ini digunakan varian ROUGE-L [7], [8], yang secara spesifik mengukur *Longest Common Subsequence* (LCS)[19] untuk menilai kesamaan faktual dan kesesuaian struktur kalimat. Sebagai metrik yang berorientasi pada *recall* (sesuai namanya), skor ROUGE-L ($R_{cls}$) dihitung dengan membagi panjang LCS dari teks kandidat (Y) dan teks referensi (X) dengan total panjang dari teks referensi (m).

Di mana $LCS(X,Y)$adalah panjang *subsequence* umum terpanjang antara referensi X dan kandidat Y, dan $m$ adalah jumlah kata dalam teks referensi X. Skor ROUGE yang tinggi menunjukkan bahwa informasi penting dari *ground truth* berhasil ditangkap dan disampaikan dengan baik oleh model.

*2.3.2. BLEU (Billingual Evaluation Understudy)*

Metrik ini digunakan untuk menilai presisi dan kelancaran linguistik dari jawaban yang dihasilkan. BLEU[9], [20], [21] menghitung jumlah *n-gram* yang cocok antara *answer* dan *ground truth*, dengan mempertimbangkan Brevity Penalty (BP) yaitu penalti yang diberikan jika jawaban terlalu pendek atau terlalu panjang dibandingkan referensi. Skor BLEU[9], [20], [21] yang tinggi mengindikasikan bahwa jawaban model tidak hanya relevan secara semantik, tetapi juga tersusun dengan baik dari sisi tata bahasa.

Secara matematis, skor BLEU dihitung sebagai rata-rata geometris (*geometric mean*) dari presisi *n-gram* yang telah dimodifikasi (biasanya dari 1-gram hingga 4-gram), yang kemudian dikalikan dengan sebuah *Brevity Penalty* (BP). umus BLEU direpresentasikan sebagai berikut:

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \quad (5)$$

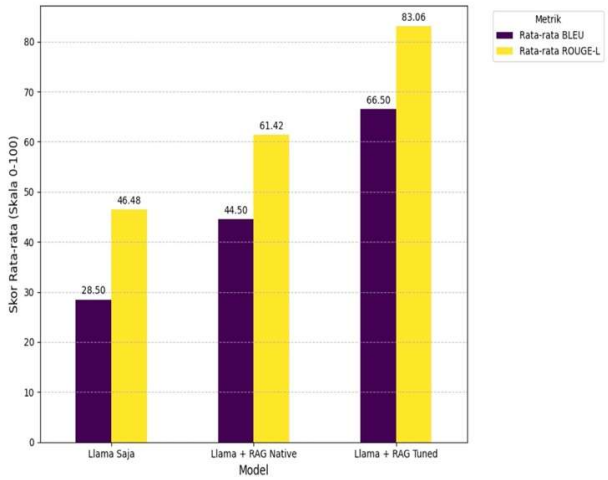
BP adalah Brevity Penalty (Penalti Kependekan). Ini adalah komponen krusial yang berfungsi untuk "menghukum" teks yang dihasilkan jika secara signifikan lebih pendek daripada teks referensi[4]. Penalti ini dihitung sebagai:

$$BP = \begin{cases} 1 & \text{jika c} > \text{r} \\ e^{1-\frac{r}{c}} & \text{jika c} \leq \text{r} \end{cases} \quad (6)$$

Skor BLEU yang tinggi (mendekati 1) mengindikasikan bahwa jawaban model tidak hanya relevan secara semantik, tetapi juga sangat mirip secara struktural (tata bahasa dan pilihan kata) dengan jawaban ideal (*ground truth*).

## 3. HASIL

Evaluasi kinerja dilakukan terhadap tiga arsitektur model: Llama Murni (*Baseline*), Llama + RAG Native, dan Llama + RAG *Fine-Tuning*. Pengukuran kinerja menggunakan dua metrik evaluasi utama, yaitu ROUGE-L untuk mengukur kelengkapan faktual dan BLEU untuk presisi linguistik. Hasil perbandingan skor rata-rata disajikan pada Gambar 5.




Gambar 5. Contoh Penulisan *Caption* pada gambar

Secara kuantitatif hasil menunjukkan adanya peningkatan kinerja yang konsisten dan signifikan pada setiap tahapan arsitektur:

1. Model Llama Murni (*Baseline*) mencatat skor terendah, dengan rata-rata ROUGE-L sebesar 46.48 dan BLEU sebesar 28.50.
2. Implementasi Llama + RAG Native berhasil meningkatkan kinerja secara signifikan, mencapai rata-rata ROUGE-L 61.42 dan BLEU 44.50.
3. Model Llama + RAG *Fine-Tuning* menunjukkan kinerja paling unggul, dengan rata-rata ROUGE-L mencapai 83.06 dan BLEU mencapai 66.50.

Temuan ini memperlihatkan adanya tren peningkatan yang jelas baik dalam akurasi faktual (ROUGE-L) maupun presisi linguistik (BLEU) seiring dengan penerapan mekanisme RAG dan optimalisasi konfigurasi sistem.

## 4. PEMBAHASAN

Temuan kuantitatif di atas kemudian diperkuat dengan analisis kualitatif terhadap dataset mentah (.csv) dari setiap model. Analisis ini bertujuan untuk menjelaskan faktor penyebab di balik perbedaan skor antar model.

### *4.1. Analisis Kinerja Llama Murni (*Baseline*)*

Model Llama Murni mencatat skor terendah (ROUGE-L 46.48), yang menegaskan hipotesis awal penelitian ini. Skor ini tidak nol karena model masih mampu menjawab kueri umum atau non-domain seperti sapaan ("Halo Dira") atau ungkapan terima kasih.

Namun, kegagalan utamanya terletak pada ketidak mampuannya menjawab pertanyaan yang bersifat domain spesifik. Analisis pada file evaluasi_llama_saja.csv menunjukkan bahwa untuk pertanyaan faktual seperti "Di mana alamat kantor TVKU?" atau "Siapa direktur utama tvku?", model secara konsisten gagal memberikan jawaban faktual dan hanya merespons dengan "Maaf, saya hanya bisa memberikan informasi seputar TVKU". Kegagalan dalam *knowledge retrieval* ini membuktikan bahwa LLM *baseline* tidak dapat diandalkan untuk aplikasi domain spesifik tanpa augmentasi pengetahuan.

### *4.2. Analisis Kinerja Llama + RAG Native*

Implementasi RAG *native* memberikan peningkatkan kinerja secara signifikan (ROUGE-L 61.42). Ini terjadi karena system *retriever* berhasil menyediakan konteks yang sebelumnya tidak dimiliki oleh model *baseline*. Model ini sekarang mampu menjawab pertanyaan faktual seperti "Apa visi dan misi TVKU?".

Namun, skor 61.42 menunjukkan bahwa model ini masih jauh dari sempurna. Analisis kualitatif pada file evaluasi_rag_native.csv mengungkap dua masalah kritis:

Dalam beberapa kasus, meskipun *retriever* berhasil menemukan konteks yang benar, model LLM gagal mematuhinya. Contoh paling jelas adalah pada kueri "Apa visi dan misi TVKU?". Konteks yang diberikan sudah benar, namun model berhalusinasi dan menjawab dengan menyebut "Universitas Tarumanegara", sebuah fakta yang sama sekali tidak ada dalam konteks.

Pada kueri topik seperti "Siapa presiden Indonesia saat ini?", *retriever* berhasil menemukan konteks yang menginstruksikan model untuk tidak menjawab ("Maaf, informasi…tidak tersedia…"). Namun, model Llama mengabaikan instruksi ini dan tetap menjawab ("Jawabannya adalah Joko Widodo ya") dari pengetahuan internalnya.

Temuan ini menunjukkan bahwa RAG *native* memang lebih baik dari *baseline*, namun kendalanya masih rendah karena model dasar Llama belum terlatih untuk "patuh" pada konteks yang disediakan.

### *4.3. Analisis Llama + RAG Fine-Tuning*

Model RAG *Fine-Tuning* menunjukkan kinerja tertinggi secara absolut (ROUGE-L 83.06 dan BLEU 66.50). peningkatan drastis dari RAG *native* ini membuktikan bahwa *fine-tuning* adalah Langkah optimalisasi yang krusial. Analisis menunjukkan bahwa *fine-tuning* berhasil menyelesaikan dua masalah utama yang ada ada RAG *native*:

Proses *fine-tuning* telah "melatih" model untuk lebih memprioritaskan informasi dari *prompt* konteks daripada pengetahuan internalnya. Hal ini secara efektif menghilangkan halusinasi kontekstual seperti kasus "Universitas Tarumanegara".

Model yang di-*fine-tuning* belajar untuk menjawab dengan format yang lebih ringkas, faktual, dan relevan, sesuai dengan format *ground_truth* pada data latihnya. Ini tidak hanya meningkatkan skor ROUGE-L (akurasi) tetapi juga menjelaskan lompatan besar pada skor BLEU (dari 44.50 ke 66.50), karena model mengurangi bas abasi konversasional ("ya!", "hehe") yang tidak perlu.

Secara keseluruhan, temuan ini mengkonfirmasi bahwa untuk membangun chatbot domain spesifik yang andal, kombinasi RAG dengan *fine-tuning* adalah arsitektur yang paling unggul, karena *fine-tuning* melatih LLM untuk menjadi "mesin perangkum" yang patuh dan efisien terhadap konteks yang diberikan oleh *retriever*.

## 5. KESIMPULAN

Penelitian ini bertujuan untuk menganalisis dan membandingkan secara kuantitatif kinerja dari tiga struktur chatbot: Llama Murni (*baseline*), Llama + RAG *Native*, dan Llama + RAG *Fine-Tuning*. Berdasarkan hasil evaluasi, dapat ditarik beberapa kesimpulan:

Llama Murni terbukti tidak memadai untuk kueri domain spesifik, mencatat skor terendah (ROUGE-L 46.48, BLEU 28.50). analisis kualitatif menunjukkan model ini secara konsisten gagal menjawab pertanyaan faktual karena ketiadaan konteks.

Implementasi Llama + RAG *Native* berhasil meningkatkan kinerja secara signifikan (ROUGE-L 61.42, BLEU 44.50). Ini membuktikan bahwa augmentasi konteks eksternal sangat penting. Namun, arsitektur ini masih menderita masalah keandalan yang kritis, seperti kontekstual (misalnya menyebut "Universitas Tarumanegara") dan kegagalan kepatuhan instruksi (misalnya, menjawab kueri diluar topik).

Llama + RAG *Fine-Tuning* secara definitive terbukti sebagai arsitektur yang paling unggul, mencapai skor tertinggi (ROUGE-L 83.06, BLEU 66.50). *Fine-tuning* tidak hanya menigkatkan kemampuan model untuk merangkum konteks dengan lebih akurat, tetapi juga secara efektif menyelesaikan masalah halusinasi dan kepatuhan instruksi yang ditemukan pada RAG *native*.

Secara keseluruhan, penelitian ini menyimpulkan bahwa untuk membangun chatbot domain spesifik yang andal dan akurat, arsitektur RAG *native* saja tidak cukup. Kombinasi RAG dengan *fine-tuning* adalah metode yang paling efektif untuk memastikan *grounding* faktual dan mengurangi halusinasi LLM.

## DAFTAR PUSTAKA

[1] A. Khan, Y. Wang, W. Zhang, Y. Tian, and M. T. Özsu, "LLM + Vector Data: Coupling of Large Language Models with Vector Data Management for Enhancing Data Science," *Proceedings - 2025 IEEE 41st*

 https://doi.org/10.25077/TEKNOSI.v11i3.2025.263-272

*International Conference on Data Engineering Workshops, ICDEW 2025*, pp. 93–96, 2025, doi: 10.1109/ICDEW67478.2025.00018.

[2] P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Apr. 2021, [Online]. Available: http://arxiv.org/abs/2005.11401

[3] Y. V Patel, V. Salvia, and I. Kumar, "A Survey on Retrieval-Augmented Generation: From Naive to Adaptive Approaches with Financial Insights."

[4] A. Kotiyal, J. Praveen Gujjar, M. S. Guru Prasad, R. M. Devadas, V. Hiremani, and P. Tangade, "Chat with PDF using LangChain Model," in *2nd IEEE International Conference on Advances in Information Technology, ICAIT 2024 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ICAIT61638.2024.10690817.

[5] K. R. Ong and W. P. Wong, "Optimizing Information Retrieval in RAG through Intelligent Reranking and Follow-Up Query Predictions," Sep. 03, 2025. doi: 10.21203/rs.3.rs-7506627/v1.

[6] S. Mathur and A. Chhabra, "Vector Search Algorithms: A Brief Survey," *Proceedings of the 4th International Conference on Ubiquitous Computing and Intelligent Information Systems, ICUIS 2024*, pp. 365–371, 2024, doi: 10.1109/ICUIS64676.2024.10866377.

[7] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," 2004. Accessed: Oct. 25, 2025. [Online]. Available: https://aclanthology.org/W04-1013/

[8] H. Rachmat, H. Riza, and T. F. Abidin, "Fine-Tuning Large Language Model (LLM) to Answer Basic Questions for Prospective New Students at Syiah Kuala University Using the Retrieval-Augmented Generation (RAG) Method," *2024 9th International Conference on Informatics and Computing, ICIC 2024*, 2024, doi: 10.1109/ICIC64337.2024.10956296.

[9] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a Method for Automatic Evaluation of Machine Translation."

[10] J. Gohil, H. L. Shifare, and M. Shukla, "Developing a User-Friendly Conversational AI Assistant for University Using Ollama and LLama3," *2025 International Conference on Data Science, Agents and Artificial Intelligence, ICDSAAI 2025*, 2025, doi: 10.1109/ICDSAAI65575.2025.11011878.

[11] AI@Meta, "Llama 3 Model Card," 2024. [Online]. Available: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[12] A. Sonkar, S. P. Singh, K. Sahu, A. Sahu, and S. Mishra, "Dynamic Query Handling with RAG Fusion for PDF-Based Knowledge Retrieval Systems," *2025 4th OPJU International Technology Conference on Smart Computing for Innovation and Advancement in Industry 5.0, OTCON 2025*, 2025, doi: 10.1109/OTCON65728.2025.11070378.

[13] X. Xie, H. Liu, W. Hou, and H. Huang, "A Brief Survey of Vector Databases," *2023 9th International Conference on Big Data and Information Analytics, BigDIA 2023 - Proceedings*, pp. 364–371, 2023, doi: 10.1109/BIGDIA60676.2023.10429609.

[14] S. Vakayil, D. Sujitha Juliet, J. Anitha, and S. Vakayil, "RAG-Based LLM Chatbot Using Llama-2," *ICDCS 2024 - 2024 7th International Conference on Devices, Circuits and Systems*, pp. 195–199, 2024, doi: 10.1109/ICDCS59278.2024.10561020.

[15] M. R. Putri, A. Y. Husodo, and B. Irmawati, "Simplification of Embedding Process in Retrieval Augmented Generation for Optimizing Question Answering Chatbot Model," *COMNETSAT 2024 - IEEE International Conference on Communication, Networks and Satellite*, pp. 665–670, 2024, doi: 10.1109/COMNETSAT63286.2024.10862926.

[16] U. Hasanah and B. P. Hartato, "Assessing Short Answers in Indonesian Using Semantic Text Similarity Method and Dynamic Corpus," *ICITEE 2020 - Proceedings of the 12th International Conference on Information Technology and Electrical Engineering*, pp. 312–316, Oct. 2020, doi: 10.1109/ICITEE49829.2020.9271696.

[17] N. Mardiana, R. D. Dana, Faisal, I. Farida, A. G. Azwar, and Nurwathi, "Similarity Measures Implementation on Face Authentication using Indonesian Citizen ID Card," *Proceeding of 2023 17th International Conference on Telecommunication Systems, Services, and Applications, TSSA 2023*, 2023, doi: 10.1109/TSSA59948.2023.10366880.

[18] G. Putro Dirgantoro, M. A. Soeleman, and C. Supriyanto, "Smoothing Weight Distance to Solve Euclidean Distance Measurement Problems in K-Nearest Neighbor Algorithm," *Proceedings - 2021 IEEE 5th International Conference on Information Technology, Information Systems and Electrical Engineering: Applying Data Science and Artificial Intelligence Technologies for Global Challenges During Pandemic Era, ICITISEE 2021*, pp. 294–298, 2021, doi: 10.1109/ICITISEE53823.2021.9655820.

[19] D. S. Sisodia, I. Nehapriyanka, and P. Amulya, "Longest common subsequence based multistage collaborative filtering for recommender systems," *Proceedings - 2020 21st International Arab Conference on Information Technology, ACIT 2020*, Nov. 2020, doi: 10.1109/ACIT50332.2020.9300068.

[20] P. Malik and A. S. Baghel, "An improvement in BLEU metric for English-Hindi machine translation evaluation," *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2016*, pp. 331–336, Jan. 2017, doi: 10.1109/CCAA.2016.7813740.

[21] J. Zhang, "Improving Genetic Algorithm-Based Automatic Machine Translation Models with Gated Recurrent Units and Bilingual Evaluation Understudy," pp. 12–16, Sep. 2025, doi: 10.1109/ICICR65456.2025.00010.

## NOMENKLATUR

| | |
|---|---|
| $R_lcs$ | Skor ROUGE-L Recall |
| $P_lcs$ | Skor ROUGE-L Precision |

| | |
|---|---|
| $F_lcs$ | Skor ROUGE-L F1-Score (metrik ROUGE utama yang dilaporkan) |
| $LCS(X,Y)$ | Panjang *Longest Common Subsequence* antara X dan Y |
| $X$ | Teks referensi (jawaban ideal atau *ground truth*) |
| $Y$ | Teks kandidat (jawaban yang dihasilkan oleh model) |
| $m$ | Panjang teks referensi X (dihitung dalam jumlah kata) |
| $n$ | (pada ROUGE-L) Panjang teks kandidat Y (dihitung dalam jumlah kata) |
| $BLEU$ | Skor akhir BLEU |
| $BP$ | *Brevity Penalty* (Penalti Kependekan) |
| $N$ | Orde *n-gram* maksimum yang dievaluasi (umumnya $N = 4$) |
| $n$ | (pada BLEU) Iterator orde *n-gram*, dari 1 hingga $N$ |
| $p_n$ | Skor presisi *n-gram* yang dimodifikasi untuk orde $n$ |
| $w_n$ | Bobot untuk skor $p_n$ (umumnya $1/N$) |
| $c$ | Panjang total teks kandidat (jawaban model) |
| $r$ | Panjang efektif teks referensi (jawaban ideal) |
| $exp$ | Fungsi eksponensial |
| $\overrightarrow{(q)}$ | Representasi vektor dari kueri pengguna |
| $\overrightarrow{(d)}$ | Representasi vektor dari *document chunk* (dokumen basis pengetahuan) |
| $\theta$ | Sudut (orientasi) antara $\vec{q}$ dan $\vec{d}$ |
| $\overrightarrow{(q)} \cdot \overrightarrow{(d)}$ | Perkalian titik (*dot product*) antara $\vec{q}$ dan $\vec{d}$ |
| $\left\|\overrightarrow{(q)}\right\|$ | Norma $L_2$ (magnitudo atau panjang) dari $\vec{q}$ |
| $\left\|\overrightarrow{(d)}\right\|$ | Norma $L_2$ (magnitudo atau panjang) dari $\vec{d}$ |

## BIODATA PENULIS

Nindya Desyana
Mahasiswi Program Studi Teknik Informatika S-1 Fakultas Ilmu Komputer Universitas Dian Nuswantoro, fokus peminatan pada *Data Science, Artificial Intelegence, Machine Learning.*

Ardytha Luthfiarta, M.Kom
Dosen Program Studi Teknik Informatka S-1 Fakultas Ilmu Komputer Universitas Dian Nuswantoro, fokus penelitian di bidang *Data Mining, Information Retrieval, Machine Learning, Deep Learning, NLP*.

 https://doi.org/10.25077/TEKNOSI.v11i3.2025.263-272