IMPLEMENTASI *RETRIEVAL AUGMENTED GENERATION* PADA *INFORMATION RETRIEVAL AND RESPONSE SYSTEM* (STUDI KASUS : ANARKISME EMMA GOLDMAN) MENGGUNAKAN LANGCHAIN

LAPORAN TUGAS AKHIR

Laporan ini disusun untuk memenuhi salah satu syarat memperoleh

Gelar Sarjana Strata 1 (S1) pada Program Studi Teknik Informatika Fakultas Tkenologi Industri Universitas Islam Sultan Agung

Disusun Oleh :

Nama	: Winda Setyaningsih
NIM	: 32602000113
Program Studi	: Teknik Informatika

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM SULTAN AGUNG

SEMARANG

2025

FINAL PROJECT

IMPLEMETATION RETRIEVAL AUGMENTED GENERATION ON INFORMATION RETRIEVAL AND RESPONSE SYSTEM (CASE STUDY : ANARCHISM BY EMMA GOLDMAN) USING LANGCHAIN

Proposed to complete the requirement to obtain a bachelor's degree (S1) At Informatics Engineering Department of Industrial Technology Faculty

Sultan Agung Islamic University

Arranged By :

WINDA SETYANINGSIH

NIM 32602000113

MAJORING OF INFORMATICS ENGINEERING

INDUSTRIAL TECHNOLOGY FACULTY

SULTAN AGUNG ISLAMIC UNIVERSITY

SEMARANG

2025

LEMBAR PENGESAHAN
TUGAS AKHIR

IMPLEMENTASI RETRIEVAL AUGMENTED GENERATION PADA INFORMATION RETRIEVAL AND RESPONSE SYSTEM (STUDI KASUS ANARKISME EMMA GOLDMAN) MENGGUNAKAN LANGCHAIN

WINDA SETYANINGSIH
NIM 32602000113

Telah dipertahankan di depan tim penguji ujian sarjana tugas akhir
Program Studi Teknik Informatika
Universitas Islam Sultan Agung
Pada tanggal : 25 Februari 2025

TIM PENGUJI UJIAN SARJANA :

Moch Taufik, ST. MIT
NIDN. 0622037505
24-04-2025 .....
(Ketua Penguji)

Andi Riyansyah, ST, M.Kom
NIDN. 0609108802
24-04-2025 .....
(Anggota Penguji)

Mustafa, ST, MM, M.Kom
NIDN. 0623117703
25-04-2025 .....
(Pembimbing)

Semarang, 25 Mei 2025

Mengetahui,
Kaprodi Teknik Informatika
Universitas Islam Sultan Agung

Moch Taufik, ST. MIT
NIDN. 0622037505

SURAT PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan dibawah ini :

Nama	: Winda Setyaningsih
NIM	: 32602000113
Judul Tugas Akhir	: Implementasi Retrieval Augmented Generation Pada Information Retrieval and Response System (studi kasus Anarkisme Emma Goldman)

Dengan bahwa ini saya menyatakan bahwa judul dan isi Tugas Akhir yang saya buat dalam rangka menyelesaikan Pendidikan Strata Satu (S1) Teknik Informatika tersebut adalah asli dan belum pernah diangkat, ditulis ataupun dipublikasikan oleh siapapun baik keseluruhan maupun sebagian, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka, dan apbila di kemudian hari ternyata terbukti bahwa judul Tugas Akhir tersebut pernah diangkat, ditulis ataupun dipublikasikan, maka saya bersedia dikenakan sanksi akademis. Demikian surat pernyataan ini saya buat dengan sadar dan penuh tanggung jawab.

Semarang, 25 Mei 2025

Yang Menyatakan,

Winda Setyaningsih

PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH

Saya yang bertanda tangan dibawah ini :

Nama : Winda Setyaningsih

NIM : 32602000113

Program Studi : Teknik Informatika

Fakultas : Teknologi industri

Alamat Asal : Blora, Jawa Tengah

Dengan ini menyatakan Karya Ilmiah berupa Tugas akhir dengan Judul : Implementasi Retrieval Augmented Generation Pada Information Retrieval and Response System (studi kasus Anarkisme Emma Goldman) Menggunakan Langchain

Menyetujui menjadi hak milik Universitas Islam Sultan Agung serta memberikan Hak bebas Royalti Non-Eksklusif untuk disimpan, dialihmediakan, dikelola dan pangkalan data dan dipublikasikan diinternet dan media lain untuk kepentingan akademis selama tetap menyantumkan nama penulis sebagai pemilik hak cipta. Pernyataan ini saya buat dengan sungguh-sungguh. Apabila dikemudian hari terbukti ada pelanggaran Hak Cipta/Plagiarisme dalam karya ilmiah ini, maka segala bentuk tuntutan hukum yang timbul akan saya tanggung secara pribadi tanpa melibatkan Universitas Islam Sultan agung.

Semarang, 25 Mei 2025

Yang menyatakan,

Winda Setyaningsih

KATA PENGANTAR

Dengan mengucap syukur alhamdulillah atas kehadirat Allah SWT yang telah memberikan rahmat dan karunianya kepada penulis, sehingga dapat menyelesaikan Tugas Akhir dengan judul "Implementasi *Retrieval Augmented Generation* pada *Information Retrieval and Response System* (Studi Kasus : Anarkisme Emma Goldman) menggunakan LangChain" ini untuk memenuhi salah satu syarat menyelesaikan studi serta dalam rangka memperoleh gelar sarjana (S-1) Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Islam Sultan Agung Semarang.

Tugas Akhir ini dapat diselesaikan atas dukungan beberapa pihak baik berupa materiil maupun non materiil, oleh karena itu penulis ingin mengucapkan terima kasih kepada :

1. Orang tua penulis yang telah memberikan dukungan baik secara materi maupun emosional.
2. Dosen pembimbing penulis Bapak Mustafa, S.T., M.M., M.Kom yang telah meluangkan waktu untuk memberikan ilmu serta saran
3. Rektor UNISSULA Bapak Prof. Dr. H. Gunarto, S.H., M.H yang telah mengizinkan penulis menyelesaikan studi di kampus ini.
4. Dekan Fakultas Teknologi Industri Ibu Dr. Novi Marlyana, S.T., M.T
5. Teman-teman penulis yang telah memberikan dukungan berupa kritik dan saran yang tidak dapat penulis sebutkan satu persatu.

Dengan segala kerendahan hati penulis menyadari bahwa penulisan laporan ini masih jauh dari kata sempurna, oleh karena itu penulis mengharap kritik dan saran untuk membantu laporan ini menjadi lebih baik.

Semarang, 25 Februari 2025

Winda Setyaningsih

DAFTAR ISI

HALAMAN JUDUL i
LEMBAR PENGESAHAN iii
LEMBAR PERNYATAAN KEASLIAN TUGAS AKHIR iv
LEMBAR PERSETUJUAN PUBLIKASI KARYA ILMIAH v
KATA PENGANTAR vi
DAFTAR ISI vii
DAFTAR GAMBAR ix
DAFTAR TABEL x
ABSTRAK xi
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Perumusan Masalah 1
1.3 Pembatasan Masalah 2
1.4 Tujuan 2
1.5 Manfaat 2
1.6 Sistematika Penulisan 2
BAB II TINJAUAN PUSTAKA DAN DASAR TEORI 4
2.1 Tinjauan Pustaka 4
2.2 Dasar Teori 8
2.2.1 Natural Language Processing 8
2.2.2 Information Rterieval and Response System 9
2.2.3 LangChain 9
2.2.4 Gemini 14
2.2.5 Matrix ROUGE dan BLEU 14
2.2.6 Cosine Similarity 15
BAB III METODE PENELITIAN 17
3.1 Deskripsi Sistem 17

3.2 Metode Penelitian 17
BAB IV HASIL DAN ANALISIS PENELITIAN 25
4.1 Hasil Penelitian 25
4.1.1 Dokumen 25
4.1.2 Pengolahan Dokumen 26
4.1.3 Setting Model LLM 26
4.1.4 Chunking 27
4.1.5 Embedding 28
4.1.6 Pembuatan *Vector Store* ChromaDB 28
4.1.7 Pembuatan Objek Koneksi dengan *Vector Store* 28
4.1.8 Pembuatan *Retriever* 28
4.1.9 Pembuatan *Prompt Engineering* 29
4.1.10 Pemanggilan *Chain* RAG 29
4.1.11 Response 30
4.2 Analisa Penelitian 30
4.3 Evaluasi 32
BAB V KESIMPULAN DAN SARAN 35
5.1 Kesimpulan 35
5.2 Saran 35
DAFTAR PUSTAKA 36

DAFTAR GAMBAR

Gambar 2. 1 contoh chains sederhana 10
Gambar 2. 2 agent terkoneksi dengan model open AI 13

Gambar 3. 1 flowchar alur penelitian 17
Gambar 3. 2 pemodelan Information Retrieval and Response System 18
Gambar 3. 3 Gambar Dokumen dalam folder sample_data 21
Gambar 3. 4 Gambar Input nilai Question 21
Gambar 3. 5 Gambar Loading and Splitting dokumen 21
Gambar 3. 6 Gambar halaman 46 22
Gambar 3. 7 Gambar chunk ke 98 22
Gambar 3. 8 Gambar chunk ke 677 22
Gambar 3. 9 gambar direktori lokal bernama chroma_ 23
Gambar 3. 10 Gambar pemrosesan input question 23
Gambar 3. 11 Gambar input question dan dokumen retriever 23
Gambar 3. 12 Gambar hasil generasi model 24

Gambar 4. 1 cover e-book “Ini Bukan Revolusiku” 25
Gambar 4. 2 gambar halaman 1 e-book “Ini Bukan Revolusiku” 25
Gambar 4. 3 gambar halaman 46 dari dokumen 26
Gambar 4. 4 gambar proses setting model LLM 26
Gambar 4. 5 gambar proses chunking 27
Gambar 4. 6 gambar chunks ke 98 dengan pemisah \n 27
Gambar 4. 7 proses embedding 28
Gambar 4. 8 output response 30
Gambar 4. 9 dokumen 1 dan 2 31
Gambar 4. 10 dokumen 3 31
Gambar 4. 11 response 1 dokumen 31
Gambar 4. 12 response 10 dokumen 32
Gambar 4. 13 hasil evaluasi 32

DAFTAR TABEL

Tabel 1 1 Rujukan Penelitian 8

ABSTRAK

Perkembangan *Large Language Model* (LLM) yang sangat pesat mampu menyediakan beragam informasi yang memudahkan aktivitas manusia hampir di setiap aspek kehidupan. Halusinasi yang disebabkan oleh kurangnya konteks serta terbatasnya data latih menjadi evaluasi yang sangat krusial untuk LLM. Penelitian ini mengembangkan model *Retrieval Augmented Generation* pada *Information Retrieval and Response System* yang menggabungkan kemampuan LLM dengan pengetahuan eksternal berupa *e-book* untuk meminimalisir halusinasi oleh LLM. Dengan memanfaatkan beberapa fitur dalam LangChain seperti *prompt engineering* dan *wrapper library* memudahkan proses generasi *output*.

Kata Kunci : *Large Language Model*, Halusinasi, RAG, LangChain.

ABSTRACT

The rapid development of Large Language Models (LLMs) enables the provision of diverse information that facilitates human activities across various aspects of life. However, hallucinations caused by a lack of context and limited training data are critical evaluations for LLMs. This research develops model on a Information Retrieval and Response System that combines the capabilities of LLMs with external knowledge in the form of e-books to minimize hallucinations. By leveraging features such as prompt engineering and wrapper libraries in LangChain, the output generation process becomes more straightforward.

Keywords: Large Language Model, Hallucination, RAG, LangChain

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kemunculan *Large Language Model* (LLM) dengan kemampuan pemrosesan bahasa alami atau biasa disebut *Natural Language Processing* (NLP) telah memicu berbagai perkembangan sistem pengambilan informasi dan respon (IRR) seperti chatbot, sistem pencarian, sistem tanya jawab, dll. *Large Language Model* sendiri memiliki pengetahuan yang diperoleh dari pelatihan berbagai *corpus* dalam skala besar seperti *The Pile* dan *OpenWebTextCorpus*(Roy *et al.,* 2024). Namun, realita yang terjadi pertanyaan dari pengguna LLM tidak selalu dapat dijawab karena informasi dalam data latih terbatas pada tahun tertentu. Selain itu, LLM tidak mampu memberikan informasi yang berasal dari sumber tertutup yang memerlukan akses melalui akun individu maupun lembaga, beberapa faktor tersebut ditambah kurangnya kemampuan untuk memperbaiki konteks menyebabkan LLM mengalami halusinasi (Neupane *et al.,* 2024)

Untuk mengatasi tantangan tersebut, salah satu solusinya adalah menggabungkan dokumen berupa *e-book*, jurnal, atau bentuk lain pengetahuan tertulis yang berasal sumber yang tidak dapat diakses oleh LLM pada *Information Retrieval and Response System* sebagai pengetahuan eksternal yang bisa diakses LLM untuk menghasilkan jawaban yang kontekstual serta meminimalisir halusinasi. Dalam penelitian ini penulis mengembangkan model *Retrieval Augmented Generation* pada *Information Retrieval and Response System* yang berfokus pada literatur Anarkisme dari Emma Goldman dengan memanfaatkan *framework* LangChain untuk membangun *Retrieval Augmented Generation* yang mampu memberikan jawaban kontekstual walaupun dengan skor *similarity* yang rendah.

1.2 Perumusan Masalah

Bagaimana mengimplementasikan *Retrieval Augmented Generation* pada *Information Retrieval and Response System* menggunakan LangChain?

1.3 Pembatasan Masalah

Pembatasan masalah pada penelitian ini antara lain :

1. Penelitian ini difokuskan untuk menyajikan informasi dari *e-book* Ini Bukan Revolusiku, buku terjemahan dengan judul asli *Anarchism and Other Essays* karya Emma Goldman.
2. Penelitian ini hanya membahas implementasi *Retrieval Augmented Generation* pada *Information Retrieval and Response System* menggunakan LangChain.

1.4 Tujuan

Penelitian ini bertujuan untuk mengimplementasikan *Retrieval Augmented Generation* pada *Information Retrieval and Response System* menggunakan LangChain untuk menyajikan informasi dari *e-book* terjemahan *Anarchism and Other Essays* karya Emma Goldman.

1.5 Manfaat

Berdasarkan permasalahan dan tujuan yang ada, penelitian ini diharapkan dapat memberikan manfaat berupa penyajian informasi yang kontekstual melalui penggabungan pengetahuan eksternal sebagai sumber pengetahuan tambahan untuk LLM dalam menghasilkan jawaban.

1.6 Sistematika Penulisan

Berikut sistematika penulisan yang akan dipakai dalam penyusunan laporan tugas akhir ini :

BAB I : PENDAHULUAN

Bab pertama berisi latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian serta sistematika penulisan.

BAB II : TINJAUAN PUSTAKA DAN DASAR TEORI

Bab kedua berisi rujukan yang relevan dengan topik penelitian serta dasar teori tentang metode dan teknologi yang digunakan dalam pembangunan sistem.

BAB III : METODE PENELITIAN

Bab ketiga menjelaskan tahapan proses penelitian mulai dari pemuatan dokumen hingga pengolahan teks

BAB IV : HASIL DAN ANALISIS PENELITIAN

Bab keempat memaparkan hasil penelitian berupa output yang dihasilkan LLM terhadap pertanyaan yang diberikan.

BAB V : KESIMPULAN DAN SARAN

Bab kelima berisi kesimpulan dari proses penelitian dan saran yang diberikan untuk memperbaiki penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Kemunculan *Large Language Model* (LLM) dengan kemampuan pemrosesan bahasa alami atau biasa disebut *Natural Language Processing* (NLP) telah memicu berbagai perkembangan sistem pengambilan informasi dan respon (IRR) seperti chatbot, sistem pencarian, sistem tanya jawab, dll. *Large Language Model* sendiri memiliki pengetahuan yang diperoleh dari pelatihan berbagai *corpus* dalam skala besar seperti *The Pile* dan *OpenWebTextCorpus* (Roy *et al.*, 2024). Namun, realita yang terjadi pertanyaan dari pengguna LLM tidak selalu dapat dijawab karena informasi dalam data latih terbatas pada tahun tertentu. Selain itu, LLM tidak mampu memberikan informasi yang berasal dari sumber tertutup yang memerlukan akses melalui akun individu maupun lembaga, beberapa faktor tersebut ditambah kurangnya kemampuan untuk memperbaiki konteks menyebabkan LLM mengalami halusinasi. (Neupane *et al.*, 2024)

Information-Retrieval and Response (IRR) merupakan sistem yang mampu melakukan penyimpanan, pengambilan informasi yang dibekali dengan kemampuan pemrosesan bahasa alami dari *Large Language Model* (LLM) sehingga mampu melakukan pencarian informasi dan penalaran serta dirancang untuk menjawab pertanyaan yang diajukan oleh pengguna secara otomatis. *Pipeline* dari sistem tanya jawab ini mencakup input pertanyaan dari pengguna, pencarian informasi yang relevan dengan konteks *input* dan respon (Roy et al., 2024).

Bagi Emma Goldman, anarkisme adalah filosofi dari sebuah tatanan sosial baru, yang berdasarkan pada kebebasan yang tak terikat pada hukum buatan manusia. Anarkisme sendiri merupakan bentuk perlawanan serta penolakan terhadap segala bentuk dominasi yang berpotensi mereduksi daya individu untuk membuat pilihan atas hidupnya. Jika dikorelasikan dengan realitas sosial di Indonesia hari ini, hegemoni negara yang dijalankan oleh pemerintah terpilih menyebabkan banyak kekacauan mulai dari pembantaian di Papua, komersialisasi pendidikan, kapitalisasi

agama, privatisasi informasi publik, dll. Emma skeptis terhadap revolusi lantaran berkaca pada Revolusi Bolshevik dimana perpindahan tangan kediktatoran menambah kompleks masalah struktural yang disebabkan oleh kapitalisme, begitupun di Indonesia, dari tahun ke tahun tidak ada pemimpin yang benar-benar ingin memperbaiki Indonesia, masalah struktural seperti kemiskinan, eksklusifitas pendidikan, bukan membaik justru semakin memburuk.

Dalam buku Dayak Mardaheka, stigma bahwa orang Dayak itu pemalas tidak terlepas dari konstruksi sejarah yang dibangun oleh Kapten Daniel Beeckman yang merupakan pemimpin *East India Company* yang berlayar dari Inggris untuk membuka kembali jalur perdagangan setelah sebelumnya pernah mengalami konflik dengan Kesultanan Banjarmasin. Beeckman menggambarkan Dayak Ngaju yang ia tulis "Byajo" sebagai orang-orang yang cenderung pemalas, membenci industri dan perdagangan, serta hidup hanya dari perampokan dan kemurahan hati tetangga mereka. Dalam sebuah esai *Colonization and Identity* (2003) yang ditulis oleh Chris Kortright menyatakan bahwa untuk membenarkan penjajahan atas suatu bangsa, citra perlu dibuat agar penaklukan itu masuk akal dan kelak menjadi identitas bagi mereka yang terjajah, sama halnya dengan stigma bahwa orang Dayak pemalas merupakan bentuk pembenaran atas penjajahan yang dilakukan oleh Daniel Beeckman tahun 1718 silam (Bima Satria Putra, no date).

Kutipan diatas berasal dari *e-book* berjudul Dayak Mardaheka yang ditulis oleh Bima Satria Putra. Untuk mendapatkan *e-book* tersebut memerlukan akses individu melalui *google drive* yang mengharuskan memiliki akun gmail terlebih dahulu. Ketika penulis mencoba bertanya pada LLM Claude AI apakah mengetahui isi buku Dayak Mardaheka dan jawaban yang diberikan adalah buku Dayak Mardaheka ditulis oleh Tjilik Riwut. Jawaban ini jelas tidak tepat karena penulis buku tersebut bukan Tjilik Riwut melainkan Bima Satria Putra. Halusinasi yang terjadi pada LLM terjadi akibat dari data latih yang terbatas pada tahun tertentu sehingga jawaban yang dihasilkan bisa saja keliru.

Retrieval Augmented Generation hadir sebagai salah satu solusi untuk mengatasi halusinasi LLM akibat data latih yang terbatas pada tahun tertentu dan pengetahuan

tertulis yang memerlukan akun individu maupun lembaga untuk dapat mengaksesnya. RAG secara sinergis menggabungkan pengetahuan eksternal berupa dokumen yang telah melalui tahap preprocessing, chunking, embedding kemudian disimpan dalam database dalam bentuk vektor dengan pengetahuan internal LLM untuk menghasilkan jawaban yang relevan dengan pertanyaan melalui uji skor kemiripan (Wang *et al.*, 2024)

LangChain merupakan salah satu *framework* yang memiliki berbagai fitur seperti *wrapper library* yang dapat dimanfaatkan untuk membangun *Information Retrieval and Response System* berbasis RAG (Mavroudis, no date). Intergrasi dengan API yang kompleks serta adanya biaya untuk penggunaan model LLM dan embedding tertentu mampu disederhanakan oleh *wrapper* GoogleGenerativeAI agar dapat menggunakan produk dari *platform* Google AI Studio dan Vertex AI secara efisien dan gratis. *Loader* dan *splitter* dokumen yang mengadopsi dari *library* pypdf2 memungkin pemotongan dokumen perhalaman untuk memudahkan proses *chunking*. Dengan *wrapper* NLTKTextSplitter yang berbasis pada *library* NLTK, proses *chunking* dengan ukuran sesuai kebutuhan pengguna dan penambahan *overlap* untuk menjaga konteks tiap *chunk* dapat meningkatkan akurasi retrieval. *Pipeline prompt engineering* dan *context window* membantu model LLM untuk menghasilkan jawaban sesuai pesan sistem dengan mengambil informasi yang sesuai konteks pertanyaan. Proses *embedding* pertanyaan serta penghitungan kemiripan antara vektor pertanyaan dan dokumen menggunakan *Cosine Similarity* yang sudah secara *default* terjadi dalam Chroma.

No	Judul Penelitian	Ringkasan Isi
1.	QA-RAG : Leveraging Question and Answer-based Retrieved Chunk Re-Formatting for Improving Response Quality During Retrieval-augmented Generation.	Penelitian mengevaluasi metode pendekatan pada dataset RAG Bechmark menggunakan metrics evaluasi standar dan menghasilkan metode baru yang disebut QA-RAG yang dapat meningkatkan kualitas

		jawaban yang dihasilkan RAG. (Roy *et al.*, 2024)
2.	RAGAS : Automated Evaluation of Retrieval Augmented Generation.	Penelitian ini merupakan uraian evaluasi otomatis untuk metode Retrieval Augmented Generation. (Es *et al.*, 2023)
3.	Retrieval-Augmented Generation for Knowledge Intensive NLP Task	Penelitian merupakan inovasi model RAG pada bagian memori parametriknya adalah model seq2seq yang telah dilatih sebelumnya dan memori non parametriknya adalah indeks vektor dari Wikipedia.
4.	Searching for Best Practice in Retrieval-Augmented Generation.	Penelitian ini membahas inovasi metode RAG dengan Teknik pengambilan multimodal dapat meningkatkan kemampuan menjawab pertanyaan tentang input visual mempercepat pembuatan konten multimodal menggunakan strategi pengambilan sebagai generasi. (Wang *et al.*, 2024)
5.	A Methodology Combining Cosine Similarity with Classifier for Text Classification.	Penelitian ini memperkenalkan metodologi untuk klasifikasi teks dengan kombinasi Cosine Similarity untuk meningkatkan akurasi dari klasifikasi konvensional. (Park, Hong and Kim, 2020)
6.	Comparative Study and Framework for Automated Summariser Evaluation :	Penelitian ini mengurai tentang proses penilaian kemiripan konteks asli dalam dokumen dengan teks

	LangChain and Hybrid Alghorithms.	ringkasan model menggunakan framework LangChain dan ringkasan yang dihasilkan oleh manusia. (Lakshmi, no date)
7.	Revolutionizing Mental Health Care through LangChain : A Journey with Large Language Model.	Penelitian memperkenalkan chatbot Bernama MindGuide yang berfungsi sebagai asisten kesehatan mental bagi individu yang mencari bimbingan dan dukungan dalam area-area kritis. (Singh *et al.*, no date a)
8.	E-book Generative AI with LangChain : Building Large Language Model (apps) with Python, ChatGPT and Others LLMs.	Buku ini berisi tentang bagaimana membangun sebuah LLM bagi pemula dan dilengkapi source code.
9.	Out of The BLEU : How Should We Assess Quality of The Code Generation Model	Penelitian menguraikan aplikasi enam metriks evaluasi BLEU, ROUGE-L, METEOR, ChrF, CodeBLEU dan RUBY pada model generasi.
10.	Re-evaluating Automatic Summarization with BLEU and 192 Shades of ROUGE.	Penelitian ini mengurai analisis terhadap metode metriks evaluasi.

Tabel 1 1 Rujukan Penelitian

2.2 Dasar Teori

2.2.1 Natural Language Processing

Natural Language Processing (NLP) merupakan salah satu cabang kecerdasan buatan (AI) yang memiliki tugas spesifik memahami dan menafsirkan Bahasa manusia. NLP sendiri memiliki dua komponen utama yaitu *Natural Language Understanding* yang berfokus untuk mengerti Bahasa manusia dan *Natural*

Language Generation yang mampu menghasilkan format data bahasa alami. NLP juga berisi kumpulan metode dan algoritma yang memungkinkan komputer berinteraksi dan memahami bahasa manusia dengan cara yang tidak hanya bermakna namun juga relevan secara kontekstual (Takale, 2024).

2.2.2 Information Rterieval and Response System

Information-Retrieval and Response (IRR) merupakan sistem yang mampu melakukan penyimpanan, pengambilan informasi serta dibekali dengan kemampuan pemrosesan bahasa alami dari *Large Language Model* (LLM) sehingga mampu melakukan pencarian informasi dan penalaran serta dirancang untuk menjawab pertanyaan yang diajukan oleh pengguna secara otomatis. *Pipeline* dari *Information Retrieval and Response System* ini mencakup *input* pertanyaan dari pengguna, pencarian informasi yang relevan dengan konteks *input* dan respon (Roy et al., 2024).

2.2.3 LangChain

LangChain merupakan kerangka kerja serbaguna yang menawarkan berbagai pendekatan modular untuk mempermudah proses pengembangan sistem aplikasi berbasis LLM. Dengan *LangChain* pengembang dapat menyederhanakan proses kompleks rancang bangun sistem seperti proses integrasi aman dengan API, *deployment*, dll, sehingga menjadikannya lebih terukur, aman dan sadar akan konteks (Mavroudis, no date).

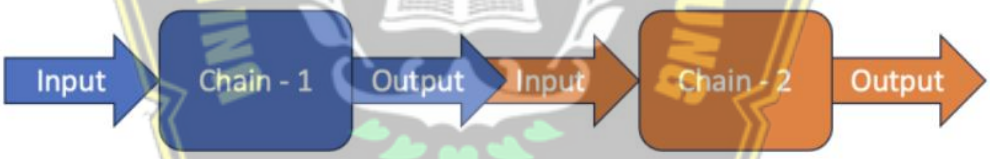
LangChain memiliki beberapa komponen antara lain :

1. *Chains*

Chains adalah rangkaian panggilan komponen dalam *pipeline* yang bisa digunakan kembali. Pengembang dapat menggabungkan beberapa panggilan LLM dan komponen lain secara berurutan untuk membuat aplikasi yang kompleks. Modularitas Rantai: Rantai memungkinkan Anda untuk memecah tugas yang kompleks menjadi komponen yang lebih kecil dan dapat diatur. Setiap komponen dapat dikembangkan dan diuji secara mandiri, membuatnya lebih mudah untuk

memelihara dan memperbarui aplikasi. Berikut beberapa keuntungan penggunaan *chains* :

1. Penyederhanaan: Dengan menggabungkan komponen-komponen ke dalam sebuah rantai, Anda dapat menyederhanakan implementasi keseluruhan aplikasi Anda. Rantai mengabstraksi kerumitan bekerja dengan komponen individu, menyediakan antarmuka tingkat tinggi untuk pengembang.
2. *Debugging*: Ketika muncul masalah dalam aplikasi Anda, rantai dapat membantu mengidentifikasi komponen yang bermasalah. Dengan mengisolasi rantai dan menguji setiap komponen secara individu, Anda dapat mengidentifikasi dan memecahkan kesalahan atau perilaku tak terduga.
3. Pemeliharaan: Rantai membuatnya lebih mudah untuk memperbarui atau mengganti komponen tertentu tanpa memengaruhi keseluruhan aplikasi. Jika versi baru dari komponen tersedia atau jika Anda ingin beralih ke komponen yang berbeda.



Gambar 2. 1 contoh *chains* sederhana

Untuk membangun sebuah rantai, Anda cukup menggabungkan komponen yang diinginkan dalam urutan yang harus dieksekusi. Setiap komponen dalam rantai menerima *output* dari komponen sebelumnya sebagai input, memungkinkan aliran data dan interaksi yang mulus dengan model Bahasa.(Singh *et al.*, no date b)

Berikut beberapa komponen chains yang digunakan dalam penelitian ini :

- *NLTKTextSplitter*

  NLTKTextSlitter adalah salah satu komponen pemrosesan teks yang digunakan LangChain, komponen ini mengadopsi *library* NLTK dengan berbagai fungsi seperti mengubah daftar dokumen secara asinkron, menyusun dokumen dari daftar teks, memisahkan teks menggunakan

tokenizer HuggingFace dan TikToken Encoder untuk menghitung panjangnya, memotong dokumen menjadi bagian yang lebih kecil *(chunk)*.

- *PyPDFLoader*

  PyPDFLoader adalah komponen pemrosesan teks yang berfungsi untuk mengekstrak informasi berupa teks dari dokumen yang diunggah untuk kemudian diproses oleh komponen lain. Komponen ini diadopsi dari *library* pypdf2.

- *Embeddings Model*

  Embedding salah satu komponen dalam pemrosesan teks dengan fungsi mengubah chunk menjadi vektor.

Dalam membangun sebuah *chains* ada beberapa metode yang dapat digunakan salah satunya adalah *LangChain Expression Language* (LCEL). LCEL memungkinkan pengembang untuk membangun *chains* yang kompleks dengan cara yang mudah. Berikut komponen dalam LCEL :

- *Runnables*

  Runnables merupakan blok bangunan dasar dalam LCEL, Runnables mewakili komponen yang dapat dijalankan dalam *chains*. *Runnables* dapat melakukan transformasi data, pemanggilan model atau fungsi lain yang dibutuhkan dalam proses alur kerja. Runnable memiliki beberapa fungsi seperti *invoke() , stream(), batch().*

  invoke() merupakan kode untuk menjalankan *Runnables* secara sinkron dan cocok untuk tugas yang memerlukan respon secara *real-time,* kode ini akan menunggu hingga proses dalam *chains* selesai sebelum melanjutkan ke proses selanjutnya. Kode ini sering digunakan untuk tugas seperti meringkas dokumen, menjawab pertanyaan dan melakukan analisis sentimen.

  stream() merupakan kode untuk menjalankan *Runnables* secara asinkron, kode ini akan memberikan hasil berupa generator yang melibatkan *user* dalam proses interaktif seperti aplikasi *streaming* teks.

batch() merupakan kode utnuk menjalankan *Runnables* pada sekumpulan input secara bersamaan sehingga kode ini memungkinkan pemrosesan banyak *input* secara efisien daripada satu-persatu.

- Operator *pipe* "|"

  Operator ini digunakan *Runnables* untuk membuat alur *chains*, ini memungkinkan pengembang untuk membuat alur yang terstruktur.

- *RunnablePassthrough*

  RunnablePassthrough merupakan bagian dari *Runnable* yang memiliki tugas spesifik meneruskan *input* tanpa memodifikasinya. *RunnablesPassthrough* biasanya menggunakan fungsi *assign()* untuk menambah atau memodifikasi kunci dalam *input dictionary*.

- *Prompt Templates*

  Promp Templates digunakan untuk membuat *prompt* yang dinamis dan dapat diatur sesuai dengan kebutuhan pengembang. Dalam *Prompt Templates* dapat dimasukan pesan atau instruksi yang digunakan sebagai acuan dalam memberikan jawaban oleh model LLM.

- LLM

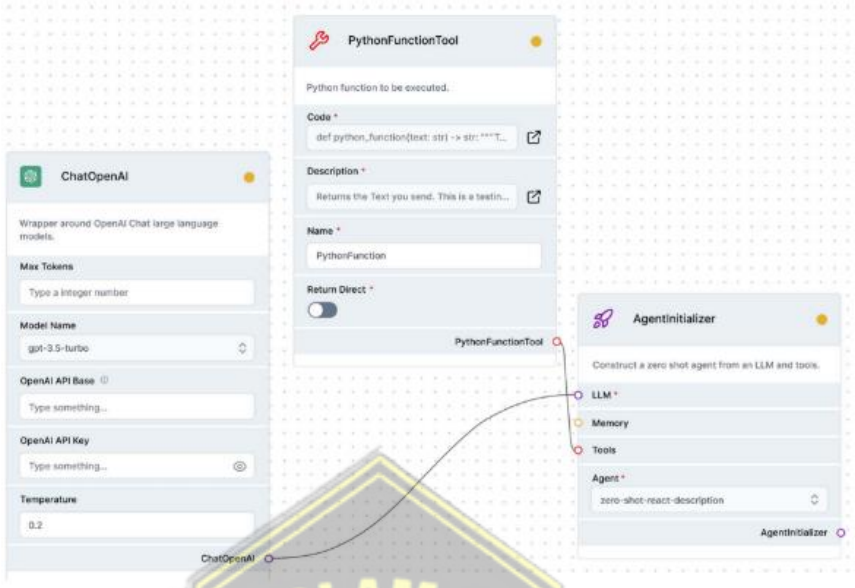
  LLM merupakan komponen inti dari banyak *chains* dalam LangChain, LLM digunakan untuk memproses *input* berupa konteks yang berisi informasi teks dari dokumen untuk kemudian menghasilkan *output* teks.

- *Output Parser*

  Output Parser digunakan untuk merapikan *ouput* LLM agar lebih terstruktur.

2. *Agents*

Agents adalah entitas perangkat lunak otonom yang mampu mengambil tindakan untuk mencapai tujuan dan menyelesaikan tugasnya. *Agents* dapat membuat sistem yang berinteraksi secara dinamis dengan pengguna sepanjang waktu.



Gambar 2. 2 agent terkoneksi dengan model open AI

Gambar tersebut memperlihatkan *Agents* terkoneksi dengan model OpenAI dan fungsi Python, proses ini menjelaskan bahwa *Agents* dapat memutuskan untuk menjalankan fungsi Python. Tiap *Agents* juga dapat memutuskan alat mana dan kapan menggunakannya.

3. *Memory*

Memory mnegacu pada keadaan bertahan antara eksekusi *Chain* atau *Agents*, penyimpanan *history context* pada *memory* meningkatkan koherensi dan relevansi respon LLM seiring berjalannya waktu. Dalam kerangka LangChain, memori implementasi dapat memiliki berbagai bentuk. Tipe *'buffer'* mendeskripsikan batasan memori berdasarkan jumlah pertukaran percakapan, sedangkan tipe 'token' menerapkan batasan yang bergantung pada jumlah token. Jenis 'memori ringkasan' mencakup ringkasan token yang diabstraksi ketika ambang batas tertentu terlampaui. Selain jenis memori ini, pengembang memiliki kebebasan untuk mengarsipkan seluruh percakapan dalam database konvensional atau penyimpanan nilai kunci. Hal ini dapat berguna untuk tujuan audit atau untuk meningkatkan kinerja sistem melalui analisis reflektif dari interaksi sebelumnya. (Topsakal and Akinci, 2023)

4. *Tools*

Tools menyediakan berbagai antar muka untuk *Agents* yang berfungsi sebagai penghubung dengan pelayanan eksternal seperti pemanggilan API, kueri basis data, kode python, dll.

2.2.4 Gemini

Gemini merupakan salah satu model yang dirilis oleh Google pada Desember 2023 dengan kemampuan penalaran, pemahaman lintas data seperti audio, teks, dan foto. Versi terbesarnya, Gemini Ultra mampu menetapkan hasil baru di 30 tolok ukur yang mencakup Bahasa, pengkodean, penalaran dan tugas multimodal seperti MMMU (*Massive Multi-discipline Multimodal*). Dalam penelitian ini LLM yang digunakan adalah Gemini 1.5-pro-latest.

2.2.5 Matrix ROUGE dan BLEU

ROUGE atau *Recall-Oriented Understudy for Gisting Evaluation* merupakan matrix yang digunakan untuk mengukur sebarapa baik informasi relevan dalam ringkasan yang dihasilkan oleh model dengan ringkasan referensi.

ROUGE-N digunakan untuk mengukur kesamaan antara ringkasan yang dihasilkan sistem dengan ringkasan referensi berdasarkan kemunculan n-gram.

$$ROUGE - N = \frac{\sum N - gram \in refCount_{match}(N - gram)}{\sum N - gram \in refCount(N - gram)} \quad (1)$$

ROUGE-L digunakan untuk mengukur kesamaan berdasarkan subskuensi terpanjang yang sama antara ringkasan sistem dan referensi.

$$ROUGE - L = \frac{LCS(ref, sys)}{total\ word\ in\ reference} \quad (2)$$

BLEU atau *Bilingual Evaluation Understudy* merupakan matrix yang digunakan untuk mengukur kemiripan hasil terjemahan model dengan referensi. Matrix BLEU

didasarkan pada ukuran presisi n-gram yang dimodifikasi dengan penalti panjang untuk kalimat yang dihasilkan model dari referensi.

$$BLEU = BP\,.exp\left(\sum_{n=1}^{N} w_n log p_n\right);BP = min\left(1, e^{1-r/c}\right), \qquad (3)$$

Dimana :

- BP (Bravery Penalty) adalah hukuman yang diterapkan pada kalimat yang dihasilkan model yang lebih pendek dari referensi.
- p_n adalah presisi yang tertimbang untuk n_gram
- w_n adalah bobot untuk kontribusi n_gram yang bervariasi dengan bobot standar adalah $w_1 = w_2 = w_3 = w_4 = \frac{1}{4}$

2.2.6 Cosine Similarity

Cosine similarity merupakan metrik yang luas digunakan dalam bidang pemrosesan bahasa alami, pengelompokan, dan sistem rekomendasi. Konsep utamanya berpusat pada representasi dokumen atau vektor kata dalam ruang vektor dimensi tinggi, di mana perbandingan antara vektor-vektor tersebut dilakukan melalui perhitungan kosinus sudut. Keuntungan dari *cosine similarity* adalah independensi dari skala dan dimensi data, sehingga cocok untuk aplikasi di mana perbedaan skala tidak relevan atau tidak diinginkan. *cosine similarity* mengukur kemiripan antara dua vektor dalam ruang vektor dengan menghitung kosinus sudut di antara mereka. Prosesnya melibatkan normalisasi vektor *input* terlebih dahulu, di mana setiap vektor dibagi dengan panjangnya, dan kemudian perhitungan kosinus sudut antara dua vektor tersebut dihitung.

Penghitungan kemiripan dua buah vektor dengan cara sebagai berikut :

Terdapat dua vektor yaitu A dan B :

$$\frac{A\,.\,B}{\|A\| \times \|B\|} \qquad (4)$$

Penjelasan :

$A\,.\,B$ adalah perkalian titik dari dua vektor $A\ dan\ B$

$||A||$ adalah panjang magnitude dari vektor A

$||B||$ adalag Panjang magnitude dari vektor B

Hasilnya adalah nilai yang berkisar dari -1 hingga 1, di mana nilai yang lebih tinggi menunjukkan kemiripan yang lebih besar antara dua vector (Park, Hong and Kim, 2020).

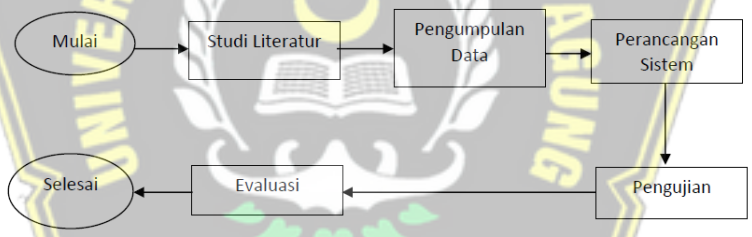
BAB III
METODE PENELITIAN

3.1 Deskripsi Sistem

Penelitian ini menggunakan *framework* LangChain untuk mengimplementasikan RAG pada *Information Retrieval and Response System* dengan model LLM Gemini-1.5-pro-latest yang mampu memahami dokumen terjemahan dan menghasilkan *output* yang kontekstual.

3.2 Metode Penelitian

Berikut *flowchat* yang digunakan dalam penelitian Implementasi *Retrieval Augmented Generation* untuk *Information Retrieval and Response System* Anarkisme Emma Goldman menggunakan LangChain :



Gambar 3. 1 flowchar alur penelitian

Gambar di atas menunjukan *flowchart* alur penelitian, tiap tahap dalam dalam alur penelitian akan dijelaskan sebagai berikut :

1. Studi Literatur

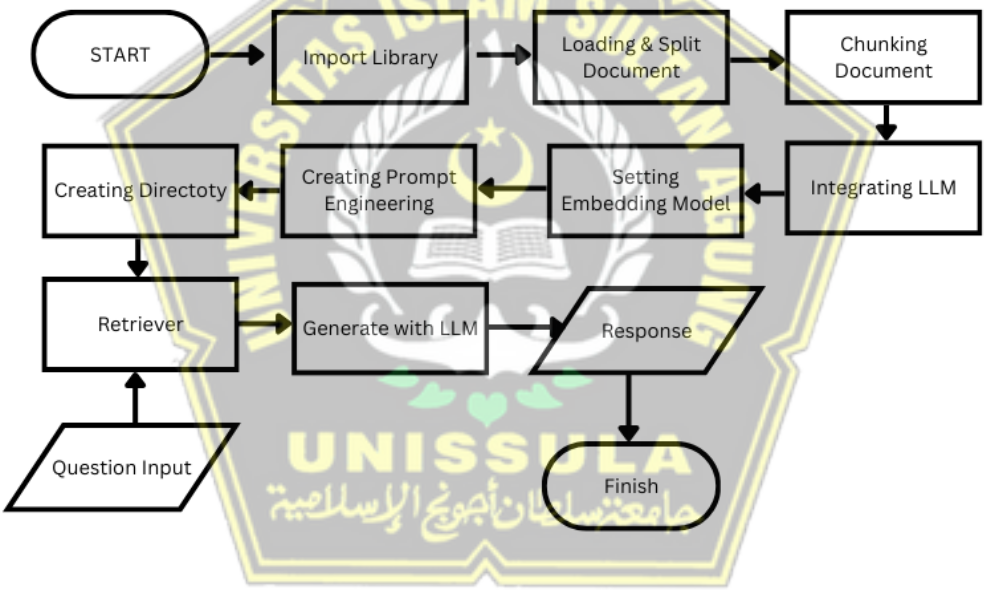
Studi Literature dilakukan dengan mengumpulkan referensi mengenai LLM, NLP, LangChain, *Cosine Similarity*, Gemini dari berbagai sumber mulai dari medium, google scholar guna memperkaya studi kasus dan rujukan model sistem yang akan dibuat oleh penulis.

2. Pengumpulan Data

Dokumen diperoleh dari *website archieve internet* yang merupakan sebuah platform yang berisi kumpulan *e-book* dengan berbagai disiplin ilmu dan lintas regional. *E-book* yang diperoleh yaitu Ini Bukan Revolusiku.

3. Perancangan Sistem

Berikut gambar *flowchart* alur *Information Retrieval and Response System* berbasis RAG menggunakan LangChain :



Gambar 3. 2 pemodelan Information Retrieval and Response System

Gambar merupakan pemodelan *Information Retrieval and Response System* anarkisme Emma Goldman berbasis RAG, berikut penjelasannya :

a. Import Library

Tahap pertama yaitu mengimport *library* yang dibutuhkan seperti *userdata* yang digunakan untuk menakses *api_key* tanpa perlu menuliskan langsung pada kode, *Google Generative AI* digunakan untuk integrasi dengan API Gemini, *NLTKTextSplitter* digunakan untuk chunking dokumen, *Google Generative AI*

Embedding digunakan untuk integrasi dengan model *embedding*, Chroma digunakan untuk penyimpanan vektor hasil *embedding*, *PyPDFLoader* digunakan untuk memuat dokumen dan membagi dokumen menjadi perhalaman, *SystemMessage* untuk memberi pesan kepada model, *ChatPromptTemplate* digunakan untuk menyusun *prompt* dalam format chat, *HumanMessagePromptTemplate* digunakan untuk memastikan model menjawab input user berdasarkan dokumen retrieval, *Markdown* digunakan untuk merapikan teks, *RunnablePassthrough* digunakan untuk meneruskan data tanpa memodifikasinya.

b. Loading & Split Document

Pada tahap ini *e-book* Ini Bukan Revolusiku diupload ke dalam *path* `/content/sample_data`. Selanjutnya dokumen dimuat dan dibagi menjadi per-halaman oleh *PyPDFLoader* lalu disimpan dalam variable `pages` .

c. Chunking Document

Pada tahap ini dokumen yang telah disimpan dalam *variable* `pages` dipotong menjadi bagian lebih kecil *(chunk)* menggunakan *NLTKTextSplitter* dengan ukuran 500 token (kalimat) per-chunk, untuk menjaga konteks tiap *chunk* ditambahkan *chunk ovelap* atau penambahan sisa token yang terpotong dari *chunk* sebelumnya ke dalam *chunk* selanjutnya dengan ukuran maksimal 100 token lalu hasil *chunking* disimpan dalam variable `chunks`.

d. Integrating LLM

Pada tahap ini dilakukan integrasi dengan LLM Gemini melalui *Google Generative AI* lalu menggunakan userdata untuk mengambil `api_key` tanpa perlu menuliskan pada baris kode.

e. Setting Embedding Model

Pada tahap ini dilakukan integrasi dengan model *embedding* `models/embedding-001` dari Vertex AI melalui *Google Generative AI Embedding*.

f. Creating Prompt Engineering

Pada tahap ini, `ChatPromptTemplate` digunakan untuk Menyusun prompt dalam format chat, `SystemMessage` digunakan untuk memberi instruksi ke model, `HumanMessagePromptTemplate` digunakan untuk memastikan model menjawab *input user* berdasarkan dokumen retrieval.

g. Creating Directory

Pada tahap ini dilakukan pembuatan *directory local* menggunakan ChromaDB bernama `./chroma_` lalu menambahkan variable `chunks` dan `embedding_model` yang berisi model embedding, lalu dilakukan embedding data di dalam variable chunk kemudian disimpan di dalam variable `db_connection`,. Selanjutnya variable `db_connection` dijadikan sebagai *retriever* dan bisa melakukan pencarian kemiripan dengan input query, `search_kwargs` berfungsi untuk mengatur jumlah dokumen yang diambil.

h. Question Input

Pada tahap ini *pipeline* RAG akan diaktifkan dengan *input* pertanyaan yang telah dimasukkan.

i. Retriever

Pada tahap ini *retriever* akan mencari dokumen yang relevan dalam vektor *database*, dokumen yang ditemukan akan digunakan sebagai *context* dalam *prompt*.

j. Generate with LLM

Selanjutnya chat template akan membentuk *prompt* yang menggabungkan *context* dan *question* lalu mengirimnya ke LLM.

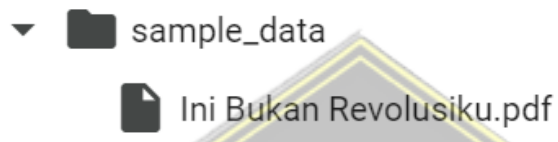
k. Respons

Setelah *prompt* dari chat template dikirim, LLM akan memberikan jawaban dengan format *Markdown* kepada *user*.

Berikut alur proses implementasi *Retrieval Augmented Generation* dalam *Information Retrieval and Response System* :

- ***Input***
  1. Upload Dokumen

     Dokumen diunggah pada folder *sample_data* yang tersedia di google colab.



Gambar 3. 3 Gambar Dokumen dalam folder sample_data

     Gambar diatas merupakan gambar dokumen yang sudah diunggah ke dalam folder *sample_data.*

  2. *Input Question*

     Pertanyaan dimasukan pada nilai question

```
({"question": "apa solusi dari anarkisme untuk melawan kapitalisme"})
```

Gambar 3. 4 Gambar Input nilai Question

     Gambar di atas merupakan gambar input nilai question

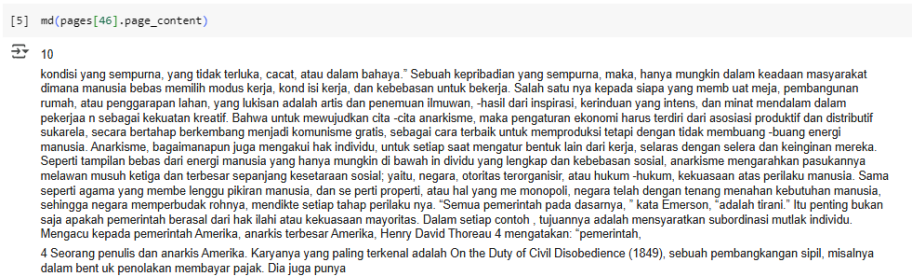
- **Proses**
  1. *Extract Text and Splitting Document*

     Informasi berupa teks dalam dokumen diekstrak lalu dipisah menjadi per halaman

```
loader = PyPDFLoader("/content/sample_data/Ini Bukan Revolusiku.pdf")
pages = loader.load_and_split()
```

Gambar 3. 5 Gambar Loading and Splitting dokumen

     Gambar di atas merupakan proses ekstraksi dan pemisahan dokumen menjadi perhalaman, lalu disimpan dalam variable `pages`.

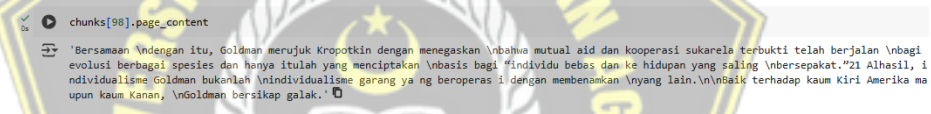


Gambar 3. 6 Gambar halaman 46

Gambar di atas merupakan gambar halaman 46 dalam variable `pages`.

2. *Chunking*

   Dokumen yang sudah dipisah menjadi per halaman akan dipisah lagi menjadi bagian yang lebih kecil (*chunk*) dengan ukuran 500 token per *chunk* lalu disimpan dalam variable bernama *chunks*

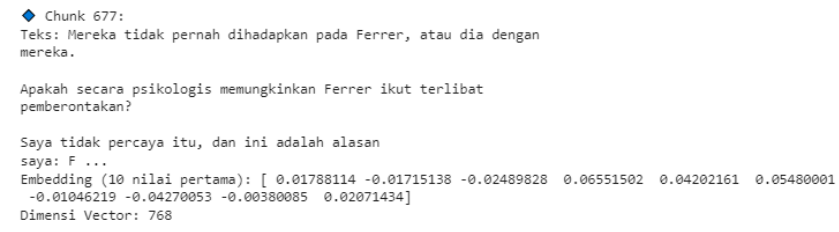


Gambar 3. 7 Gambar chunk ke 98

Gambar diatas merupakan gambar chunk ke 98

3. *Embedding*

   Embedding dilakukan oleh model *embedding* `models/embedding-001`

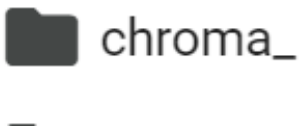


Gambar 3. 8 Gambar chunk ke 677

Gambar di atas merupakan gambar *chunk* ke 677 beserta representasi vektor dan dimensi vektornya

4. *Vector Store Database*

Embedding disimpan dalam direktori local yang dibuat mengunakan ChromaDB dengan nama `chroma_`



Gambar 3. 9 gambar direktori lokal bernama chroma_

Gambar di atas merupakan gambar direktori lokal yang digunakan untuk menyimpan vektor hasil *embedding*.

5. *Processing Input Question*

   Input Question akan melalui proses *embedding* melalui fungsi *invoke()* yang kemudian membawa *input question* ke dalam proses *embedding* dalam direktori lokal `chroma_` oleh *retriever*.

```
(context=lambda x: retriever.invoke(x["question"])[0].page_content, question=RunnablePassthrough())
```

Gambar 3. 10 Gambar pemrosesan input question

Gambar di atas merupakan gambar premroses *input question* oleh *retriever*

6. *Similarity Search*

   Proses pencarian kemiripan menggunakan *cosine similarity* dan terjadi di dalam `chroma_`

```
question = "siapa yang harus dilawan menurut anarkisme"
results = retriever.vectorstore.similarity_search_with_score(question, k=30)

# Menampilkan dokumen hasil retrieval + skor similarity
for i, (doc, score) in enumerate(results):
    print(f"\n🔷Dokumen {i+1}:")
    print(f"Skor Similarity: {score:.2f}")
    print(f"Teks: {doc.page_content}...")  # Hanya tampilkan 300 karakter pertama
    print("-" * 80)
```

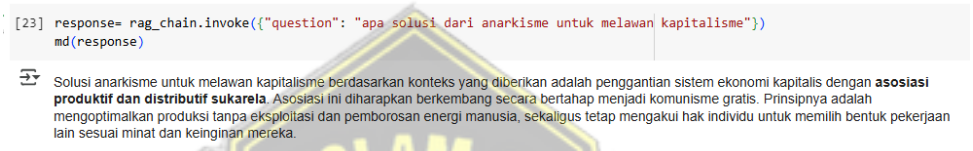
```
Skor Similarity: 0.42
Teks: Kemalasan
mental yang luas, begitu umum dalam masyarakat,
membuktikan bahwa hal ini memang benar.

Dari pada pergi ke
bagian bawah setiap ide yang diberikan, untuk memeriksa ke
asal-usulnya dan makna nya, kebanyakan orang lebih memilih
untuk mengutuk sama sekali,  atau bergantung pada beberapa
definisi dangkal atau merugikan, yang sebenarnya tidak penting.

Anarkisme mendesak manusia untuk berpikir, untuk
menyelidiki, menganalisis setiap proposisi; tapi supaya kapasitas...
--------------------------------------------------------------------------------
```

Gambar 3. 11 Gambar input question dan dokumen retriever

Gambar di atas merupakan gambar skor kemiripan antara *input question* dengan *chunk* dokumen

- ***Output***

Output dari proses tergantung *input question* yang diberikan. Berikut salah satu *output* yang dihasilkan oleh model :



Gambar 3. 12 Gambar hasil generasi model

Gambar di atas merupakan gambar response yang dihasilkan model

BAB IV

HASIL DAN ANALISIS PENELITIAN

4.1 Hasil Penelitian

4.1.1 Dokumen

Dokumen yang digunakan merupakan *e-book* terjemahan "Ini Bukan Revolusiku" dengan karya asli berjudul *"Anarchism and Other Essays"* karya Emma Goldman. Buku memiliki halaman berjumlah 240 halaman. *E-book* ini merupakan terjemahan dari karya yang berbahasa *Inggris* dengan pembendaharaan bahasa yang cukup kompleks.

Gambar 4. 1 cover e-book "Ini Bukan Revolusiku"

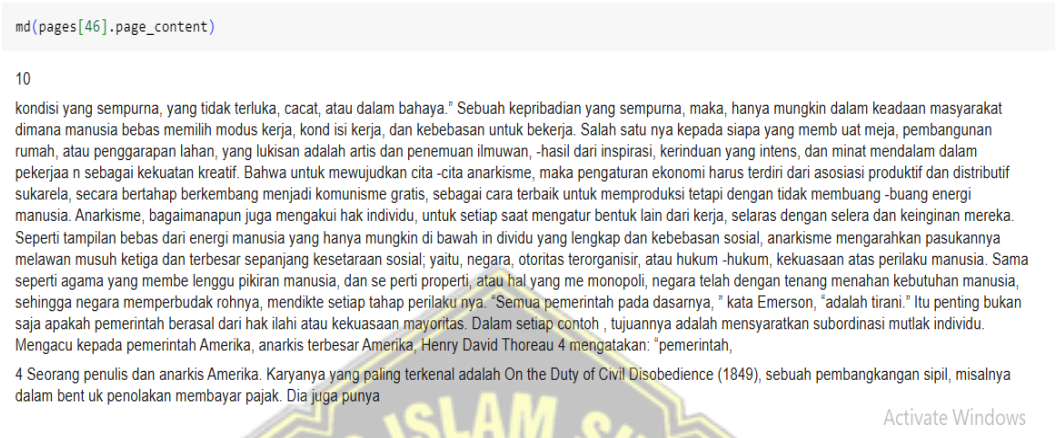
Gambar diatas merupakan gambar *cover e-book* "Ini Bukan Revolusiku" dalam bentuk PDF.

Gambar 4. 2 gambar halaman 1 e-book "Ini Bukan Revolusiku"

Gambar diatas merupakan gambar halaman 1 dari *e-book* Ini Bukan Revolusiku

4.1.2 Pengolahan Dokumen

Dokumen kemudian dimuat dan dipisah menjadi per halaman menggunakan *PyPDFLoader*.



Gambar 4. 3 gambar halaman 46 dari dokumen

Gambar di atas merupakan gambar halaman 46 dari *e-book* Ini Bukan Revolusiku

Tahap ini sangat penting karena memisahkan dokumen mejadi per halaman akan memudahkan proses pemotongan menjadi bagian yang lebih kecil atau *chunking*.

4.1.3 Setting Model LLM

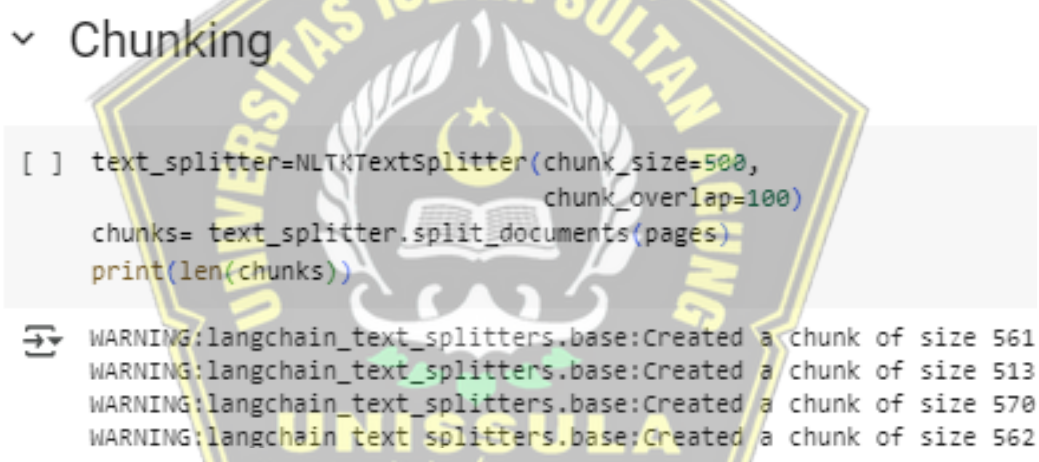
Tahap selanjutnya adalah *setting* model LLM, pada penelitian ini LLM yang digunakan adalah Gemini 1.5-pro-latest. *Setting* dilakukan dengan memanggil *API KEY* yang didapatkan dari *website* resmi *Google AI Studio* sebagai syarat penggunaan model LLM mereka.

Gambar 4. 4 gambar proses *setting model LLM*

Gambar di atas merupakan gambar proses *setting model LLM*

4.1.4 Chunking

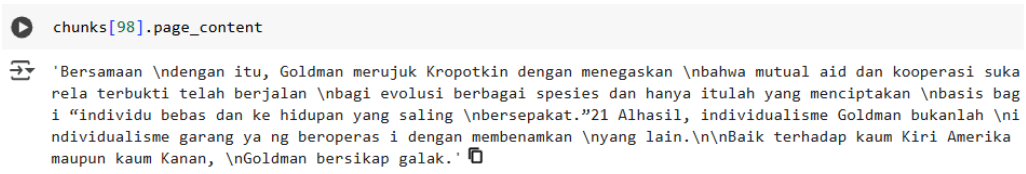
Pada tahap ini dilakukan pemecahan dokumen yang telah dibagi menjadi per halaman dengan ukuran 500 token untuk tiap pecahan. Proses chunking ini menggunakan *wrapper NLTKTextSplitter* yang mengadopsi model *library* NLTK. Skema yang digunakan adalah tokenisasi perkalimat atau biasa dikenal *sentence tokenizer*, model akan mencari tanda baca titik lalu memecahnya hingga mencapai ukuran 500 token. Untuk menjaga konteks tiap *chunk*, model diatur untuk melakukan ovelap atau penambahan potongan token dari *chunk* sebelumnya ke *chunk* setelahnya dengan jumlah 100 token.



Gambar 4. 5 gambar proses *chunking*

Gambar di atas merupakan gambar proses *chunking*

Chunking dilakukan dengan format diatas bertujuan untuk menjaga konteks tiap *chunk* ketika dilakukan proses *embedding*.

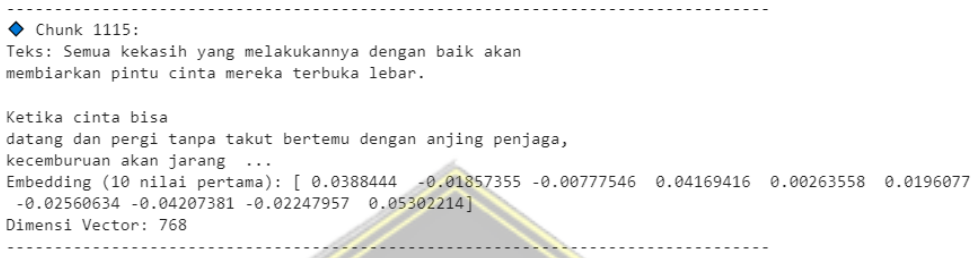


Gambar 4. 6 gambar chunks ke 98 dengan pemisah \n

Gambar di atas merupakan gambar *chunks* ke 98 dengan pemisah \n

4.1.5 Embedding

Pada tahap ini embedding dilakukan menggunakan salah satu model *embedding* modern dari Vertex AI yang dapat menangani proses *preprocessing* sederhana seperti perbedaan besar/kecil huruf. Selain itu objek *embedding* di sini adalah chunk bukan token individu yang ada di dalam tiap *chunk*.



Gambar 4. 7 proses embedding

Gambar di atas merupakan gambar proses embedding yang menampilkan *chunk* ke 1115 beserta vektor dan dimensi vektornya.

4.1.6 Pembuatan *Vector Store* ChromaDB

Pada tahap ini dilakukan pembuatan tempat penyimpanan vektor hasil dari *embedding* yaitu *directory* `./chroma_` menggunakan fungsi dari Pustaka Chroma `Chroma.from_documents` dalam *vector store* ini dilakukan proses *embedding* menggunakan model `models/embedding-001`.

4.1.7 Pembuatan Objek Koneksi dengan *Vector Store*

Pada tahap ini dilakukan pembuatan variable `db_connection` yang berisi konstruktor kelas dari Chroma memuat ulang direktori local yang telah dibuat sebelumnya `./chroma_` dalam direktori tersebut akan diisi oleh model *embedding* yang sebelumnya dan dijalankan fungsi embeddingnya, hal ini bertujuan untuk mengubah *input* yang masuk ke dalam variabale `db_connection` menjadi vektor yang berbaur dengan vektor *chunks* sebelumnya.

4.1.8 Pembuatan *Retriever*

Dalam *framework* Langchain, *retriever* bertugas untuk mencari dokumen yang berbentuk vektor yang relevan dengan *input* yang diberikan *user*. Pada tahap ini

variable `db_connection.as_retriever` diubah menjadi *retriever* dengan parameter jumlah dokumen yang bisa disesuaikan `search_kwargs={'k': 20}`

`search_kwargs` merupakan pengaturan tambahan ketika melakukan pencarian di basis data vektor, fungsi ini memungkinkan opsi seperti jumlah dokumen yang ingin dambil.

4.1.9 Pembuatan *Prompt Engineering*

Prompt engineering ini bertujuan untuk meminta LLM menjawab pertanyaan berdasarkan konteks yang diberikan. Dimulai dengan `ChatPromptTemplate.from_messages` yang digunakan untuk mendefinisikan peran dalam struktur percakapan missal sebagai pengguna, sistem atau yang lain. `SystemMessage(content="Kamu adalah QA sistem yang progresif.")` fungsi ini memberikan instruksi kepada LLM bahwa merupakan QA sistem yang progresif, instruksi ini merupakan konteks awal yang akan diikuti LLM hingga akhir program. `HumanMessagePromptTemplate.from_template` fungsi ini merupakan *promp template* pengguna yang dibuat dari *string multiline* dengan pesan `'''Jawab pertanyaan berikut berdasarkan konteks.`

```
    konteks: {context}
    pertanyaan: {question}
    jawaban: '''
```

{context} merupakan variable nilai aktual yang berasal dari *chunk* dari *retriever* yang relevan dengan *{question}* atau pertanyaan yang diberikan oleh *user*. Semua fungsi di atas disimpan dalam variable `chat_template`

4.1.10 Pemanggilan *Chain* RAG

Pada tahap ini dilakukan pemanggilan `rag_chain` atau rantai proses *retrieval augmented generation* mengunakan `RunnablePassthrough` dengan meneruskan *input* ke tiap langkah dalam *chain* tanpa mengubahnya sama sekali. *Chain* ini dibangun menggunakan operator | *(pipe)* yang menghubungkan komponen-komponen yang berbeda.

`| chat_template` mengkorelasikan *output* dari proses sebelumnya ke chat_template yang berisi *template prompt engineering* sebelumnya.

`| chat_models` mengkorelasikan hasil dari chat_template ke LLM yang ditentukan dalam chat_models

`| output_parsers` berisi hasil akhir dari output LLM

`RunnablePassthrough.assign` digunakan untuk menambahkan kunci dalam *input dictionary*, `context=lambda x: retriever.invoke(x["question"])[0].page_content`, fungsi lambda mengambil *input* x yang merupakan nilai *"question"* dalam *input*. `retriever.invoke(x["question"])` fungsi ini memanggil *retriever* dengan pertanyaan dari pengguna dalam nilai *"question"* dan secara implisit melakukan *embedding* dan pencarian kemiripan vektor didalam *directory* `./chroma_ .[0].page_content` fungsi ini bertujuan untuk menambah konteks dari halaman pertama dokumen dalam retriever. `question=RunnablePassthrough()` fungsi ini akan meneruskan kunci *"question"* tanpa memodifikasinya.

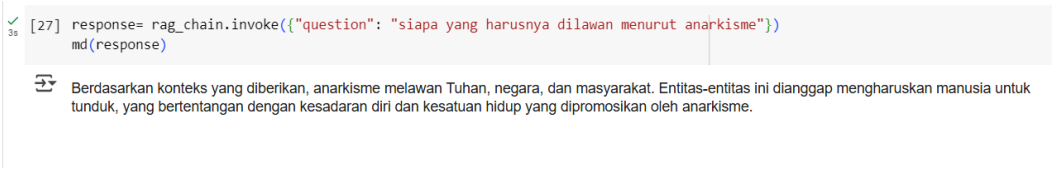
4.1.11 Response

```
response= rag_chain.invoke({"question": "siapa yang harusnya dilawan menurut anarkisme"})
```

`md(response)` *response* disini dihasilkan dari proses dalam *rag_chain* dengan memasukan nilai *"question"* dalam rantai pemrosesan tersebut.

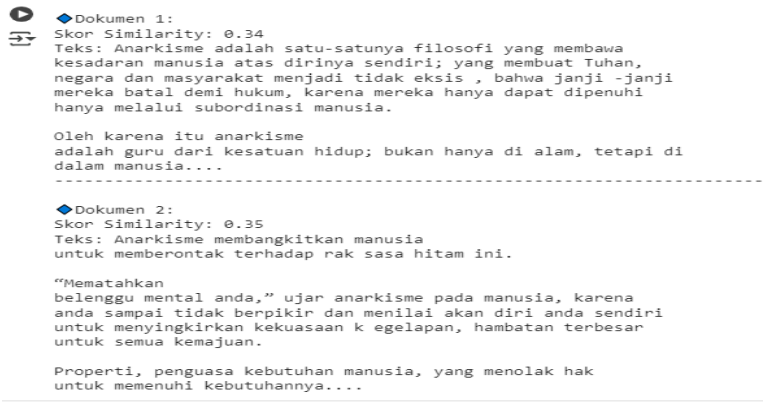
4.2 Analisa Penelitian

Berikut hasil dari serangkaian proses implementasi RAG :

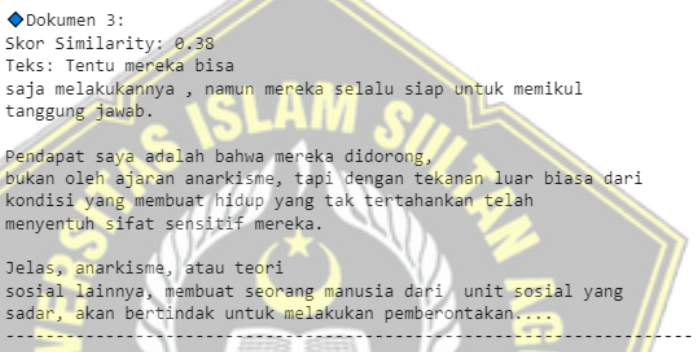


Gambar 4. 8 output response

Gambar di atas merupakan *output response* dari LLM berdasarkan tiga konteks dan pertanyaan yang diberikan.



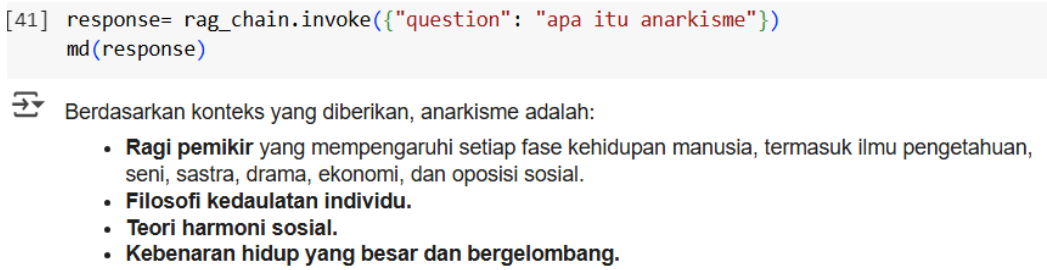
Gambar 4. 9 dokumen 1 dan 2



Gambar 4. 10 dokumen 3

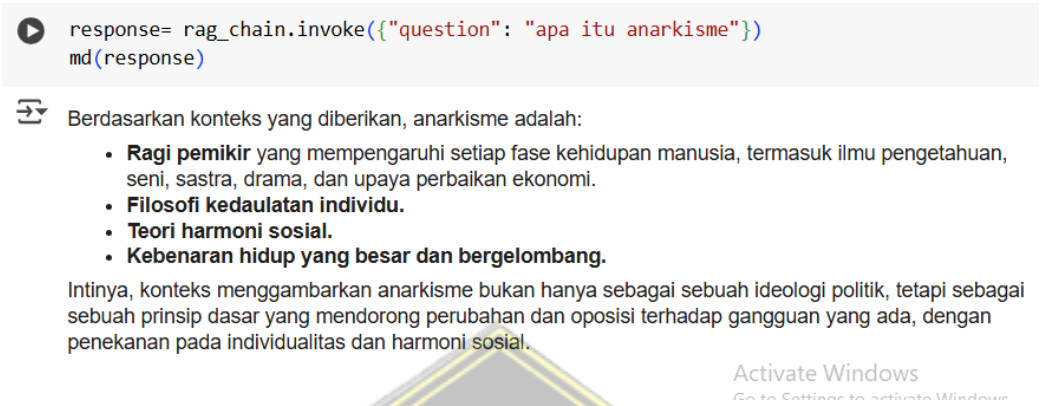
Gambar di atas merupakan gambar skor kemiripan pertanyaan dengan tiga dokumen, skor paling tinggi terdapat pada dokumen ketiga dengan nilai 0,38.

Hasil di sini tidak diambil dari dokumen dengan nilai kemiripan yang tertinggi melainkan hasil ringkasan dari konteks atau dokumen yang diberikan, sehingga semakin banyak dokumen yang dijadikan konteks maka jawaban LLM akan berubah.



Gambar 4. 11 response 1 dokumen

Gambar di atas merupakan gambar *response* dengan satu dokumen yang dijadikan sebagai konteks.

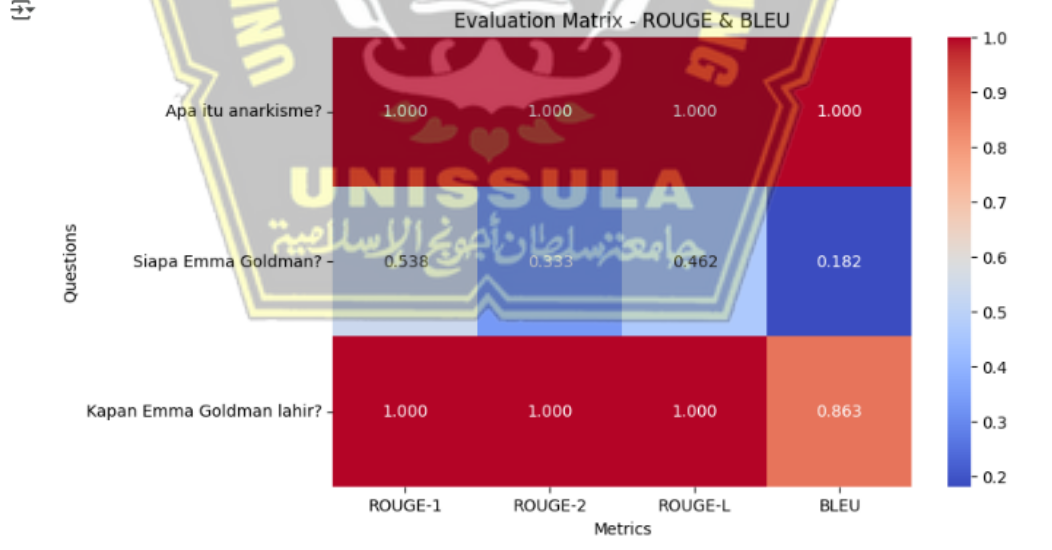


Gambar 4. 12 response 10 dokumen

Gambar di atas merupakan gambar *response* dengan sepuluh dokumen yang dijadikan sebagai konteks.

4.3 Evaluasi

Matrix ROUGE and BLEU



Gambar 4. 13 hasil evaluasi

Gambar di atas merupakan hasil evaluasi menggunakan matriks ROUGE & BLEU dengan parameter sumbu (Y) merupakan pertanyaan yang diuji, sumbu (X) metriks evaluasi yang digunakan, berikut penjelasannya :

- ROUGE 1 (kesamaan unigram/kata tunggal)
- ROUGE 2 (kesamaan bigram(dua kata)
- ROUGE L (kesamaan urutan kata terpanjang)
- BLEU (kesamaan berbaris n-gram)

generasi jawaban dari hasil implementasi RAG, skor dengan akurasi tinggi akan berwarna merah tua dan skor paling rendah berwarna biru tua.

a. Pertanyaan pertama “Apa itu Anarkisme?”

Referensi jawaban dari dokumen dengan skor kemiripan ROUGE 1 (1,000) ROUGE 2 (1,000) ROUGE L (1,000) BLEU (1.000)

```
Anarkisme bukan hanya sebagai sebuah ideologi politik, tetapi
sebagai sebuah prinsip dasar yang mendorong perubahan dan oposisi
terhadap gangguan yang ada, dengan penekanan pada individualitas
dan harmoni sosial
```

Jawaban dari model :

```
Anarkisme bukan hanya sebagai sebuah ideologi politik, tetapi
sebagai sebuah prinsip dasar yang mendorong perubahan dan oposisi
terhadap gangguan yang ada, dengan penekanan pada individualitas
dan harmoni sosial.
```

b. Pertanyaan kedua “Siapa Emma Goldman?”

Referensi jawaban dari dokumen dengan skor kemiripan ROUGE 1 (0,538) ROUGE 2 (0,333) ROUGE L (0,462) BLEU (0,182)

```
Emma Goldman adalah seorang anarkis dan feminis.
```

Jawaban dari model :

```
Emma Goldman adalah seorang anarkis, feminis, dan aktivis politik
yang dikenal karena kritiknya terhadap konsep emansipasi perempuan
pada masanya.
```

c. Pertanyaan ketika “Kapan Emma Goldman lahir?”

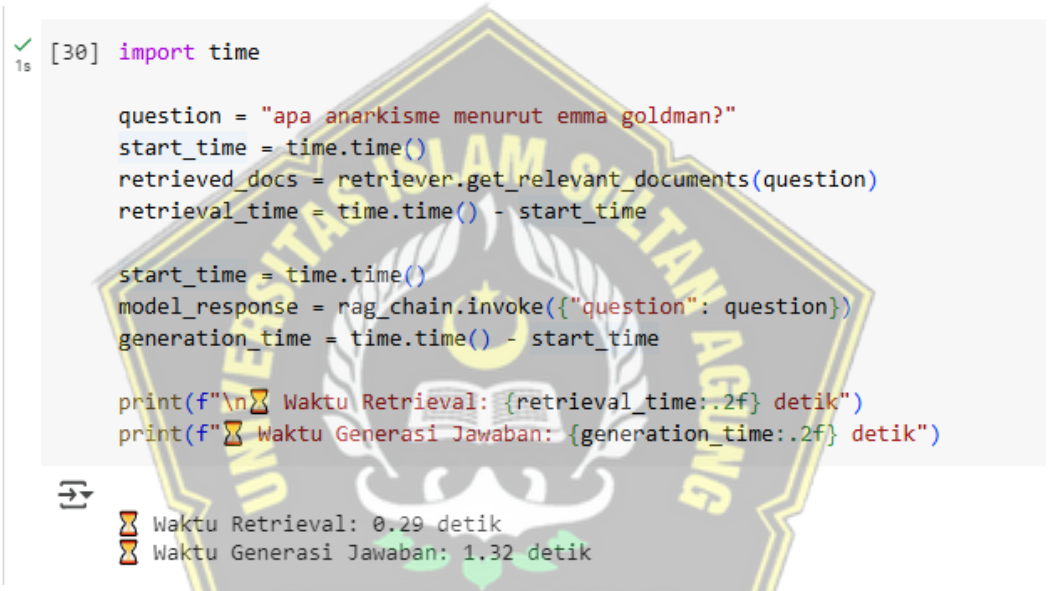
Referensi jawaban dari dokumen dengan skor kemiripan ROUGE 1 (1,000) ROUGE 2 (1,000) ROUGE L (1,000) BLEU (0,863)

```
Emma Goldman lahir pada 27 Juni 1869
```

Jawaban dari model :

```
Emma Goldman lahir pada 27 Juni 1869.
```

Performa Durasi Retrieval dan Generasi



Gambar 4. 14 Gambar durasi retrieval dan generasi

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini berhasil mengimplementasikan *Retrieval Augmented Generation* pada *Information Retrieval and Response System* menggunakan LangChain. Model mampu menampilkan hasil respon yang berbeda tergantung berapa banyak konteks yang diberikan, dokumen yang merupakan hasil terjemahan dari *e-book* berbahasa inggris memiliki pembendaharaan kata yang cukup kompleks namun model dapat mengangkap makna dari dari dokumen tersebut sehingga mengasilkan respon yang relevan dengan *input question*. Hasil evaluasi dua dari tiga jawaban menunjukan hasil yang akurat dengan skor 1,000 dan satu jawaban menunjukan hasil skor 0,182. Jawaban dengan skor rendah merupakan hasil dari sistem dengan jumlah sepuluh konteks sehingga jawaban merupakan hasil ringkasan dari konteks yang diberikan.

5.2 Saran

Berikut saran untuk memksimalkan penelitian selanjutnya :

1. Penambahan dokumen dengan beragam Bahasa, pada penelitian kali ini penulis hanya menggunakan dokumen dengan Bahasa Indonesia agar sistem dapat digunakan dalam skala global.
2. Teknik dapat diaplikasikan dalam salah satu jenis Information Retrieval and Response System seperti QA system, Asistant Virtual, FAQ, dll.
3. Penambahan antarmuka sederhana seperti streamlit agar memudahkan pengguna untuk memahami literatur yang yang dijadikan sebagai dokumen berbasis ringkasan dari model.

DAFTAR PUSTAKA

Bima Satria Putra (no date) *I*.

Es, S. *et al.* (2023) 'RAGAS: Automated Evaluation of Retrieval Augmented Generation'. Available at: http://arxiv.org/abs/2309.15217.

Lakshmi, B. (no date) *Comparative Study and Framework for Automated Summariser Evaluation: LangChain and Hybrid Algorithms*.

Mavroudis, V. (no date) *LangChain*.

Neupane, S. *et al.* (2024) 'From Questions to Insightful Answers: Building an Informed Chatbot for University Resources'. Available at: http://arxiv.org/abs/2405.08120.

Park, K., Hong, J.S. and Kim, W. (2020) 'A Methodology Combining Cosine Similarity with Classifier for Text Classification', *Applied Artificial Intelligence*, 34(5), pp. 396–411. Available at: https://doi.org/10.1080/08839514.2020.1723868.

Roy, K. *et al.* (2024) 'QA-RAG: Leveraging Question and Answer-based Retrieved Chunk Re-Formatting for Improving Response Quality During Retrieval-augmented Generation'. Available at: https://doi.org/10.20944/preprints202407.0376.v1.

Singh, A. *et al.* (no date a) *Revolutionizing Mental Health Care through LangChain: A Journey with a Large Language Model*.

Singh, A. *et al.* (no date b) *Revolutionizing Mental Health Care through LangChain: A Journey with a Large Language Model*.

Takale, D.G. (2024) 'A Study of Natural Language Processing in Healthcare Industries'. Available at: https://doi.org/10.48001/JoWACS.2024.221-6.

Topsakal, O. and Akinci, T.C. (2023) 'Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast', *International Conference on Applied Engineering and Natural Sciences*, 1(1), pp. 1050–1056. Available at: https://doi.org/10.59287/icaens.1127.

Wang, X. *et al.* (2024) 'Searching for Best Practices in Retrieval-Augmented Generation'. Available at: http://arxiv.org/abs/2407.01219.