

***CHATBOT* UNTUK INFORMASI PEMBANGUNAN WILAYAH KOTA
SEMARANG MENGGUNAKAN METODE *RETRIEVAL AUGMENTED*
*GENERATION (RAG)***

LAPORAN TUGAS AKHIR

Laporan ini disusun guna memenuhi salah satu syarat untuk menyelesaikan program studi Teknik Informatika S-1 pada Fakultas Teknologi Industri Universitas Islam Sultan Agung



Disusun Oleh:

Eko Agung Prasetyo

32602000025

**PROGRAM TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM SULTAN AGUNG
SEMARANG**

2024

FINAL PROJECT
CHATBOT FOR SEMARANG CITY AREA DEVELOPMENT
INFORMATION USING THE AUGMENTED GENERATION (RAG)
RETRIEVAL METHOD

Proposed to complete the requirement to obtain a bachelor's degree (S1)
at Informatics Engineering Departement of Industrial Technology Faculty
Sultan Agung Islamic University



Arranged By :

Eko Agung Prasetyo
NIM 32602000025

MAJORING OF INFORMATICS ENGINEERING
INDUSTRIAL TECHNOLOGY FACULTY
SULTAN AGUNG ISLAMIC UNIVERSITY
SEMARANG
2024

LEMBAR PENGESAHAN PEMBIMBING

Proposal Tugas Akhir dengan judul **“CHATBOT UNTUK INFORMASI PEMBANGUNAN WILAYAH KOTA SEMARANG MENGGUNAKAN METODE RETRIEVAL AUGMENTED GENERATION (RAG)”**

” ini disusun oleh :

Nama : Eko Agung Prasetyo

NIM : 32602000025


Program Studi : Teknik Informatika

Telah disetujui oleh Dosen Pembimbing pada :

Hari : Senin

Tanggal : 26 Agustus 2024


Mengesahkan,
Pembimbing



Imam Much Ibnu Subroto, ST, M.Sc, Ph.D

NIDN. 0613037301

Mengetahui,

Ketua Program Studi Teknik Informatika
Fakultas Teknologi Industri
Universitas Islam Sultan Agung


I. Sri Mulyono, M.Eng
NIDN 0626066601



LEMBAR PENGESAHAN PENGUJI

Proposal tugas akhir dengan judul "**CHATBOT UNTUK INFORMASI PEMBANGUNAN WILAYAH KOTA SEMARANG MENGGUNAKAN METODE RETRIEVAL AUGMENTED GENERATION (RAG)**"

” ini telah dipertahankan di depan tim penguji proposal Tugas Akhir pada :

Hari : Senin

Tanggal : 26 Agustus 2024

Penguji I



Andi Riansyah, ST, M.Kom
NIDN. 0609108802

Penguji II



Ir. Sri Mulyono, M.eng
NIDN. 0626066601



SURAT PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan dibawah ini :

Nama : Eko Agung Prasetyo

NIM : 32602000025

Judul Tugas Akhir : CHATBOT UNTUK INFORMASI PEMBANGUNAN WILAYAH KOTA SEMARANG MENGGUNAKAN METODE RETRIEVAL AUGMENTED GENERATION (RAG)

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 apabila 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, 26 Agustus 2024

Yang Menyatakan,



Eko Agung Prasetyo

PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH

Saya yang bertanda tangan dibawah ini :

Nama : Eko Agung Prasetyo

NIM : 32602000025

Program Studi : Teknik Informatika

Fakultas : Teknologi industri

Dengan ini menyatakan Karya Ilmiah berupa Tugas akhir dengan Judul :
CHATBOT UNTUK INFORMASI PEMBANGUNAN WILAYAH KOTA
SEMARANG MENGGUNAKAN METODE RETRIEVAL AUGMENTED
GENERATION (RAG)

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, 26 Agustus 2024

Yang menyatakan,



Eko Agung Prasetyo

KATA PENGANTAR

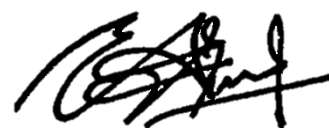
Dengan mengucapkan syukur alhamdulillah atas kehadiran Allah SWT yang telah memberikan rahmat dan karunianya kepada penulis, sehingga dapat menyelesaikan Tugas Akhir dengan judul “*Chatbot* untuk informasi pembangunan wilayah kota Semarang menggunakan metode *Retrieval Augmented Generation (RAG)*” ini untuk memenuhi salah satu syarat menyelesaikan studi serta dalam rangka memperoleh gelar sarjana (S-1) pada program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Islam Sultan Agung Semarang.

Tugas Akhir ini disusun dan dibuat dengan adanya bantuan dari berbagai pihak, materi maupun teknis, oleh karena itu saya selaku penulis mengucapkan terima kasih kepada :

1. Rektor UNISSULA Bapak Prof. Dr. H. Gunarto, S.H., M.H yang mengizinkan penulis menimba ilmu dikampus ini.
2. Dekan Fakultas Teknologi Industri Ibu Dr. Ir. Novi Marlyana, S.T., M.T., IPU., ASEAN. Eng
3. Dosen Pembimbing penulis Bapak Imam Much Ibnu Subroto, ST, M.Sc, Ph.D yang telah membimbing, meluangkan waktu dan memberi ilmu.
4. Orang Tua Penulis yang telah mengizinkan untuk menyelesaikan laporan tugas akhir ini serta memberikan dukungan dan doa agar tugas akhir ini dapat berjalan dengan lancar.
5. Dan kepada semua pihak yang tidak dapat saya sebutkan satu persatu.

Dengan segala kerendahan hati, penulis menyadari masih terdapat banyak kekurangan dari segi kualitas atau kuantitas maupun dari ilmu pengetahuan dalam penyusunan laporan, sehingga penulis mengharapkan adanya saran dan kritikan yang bersifat membangun demi kesempurnaan laporan ini di masa mendatang.

Semarang, 16 Agustus 2024



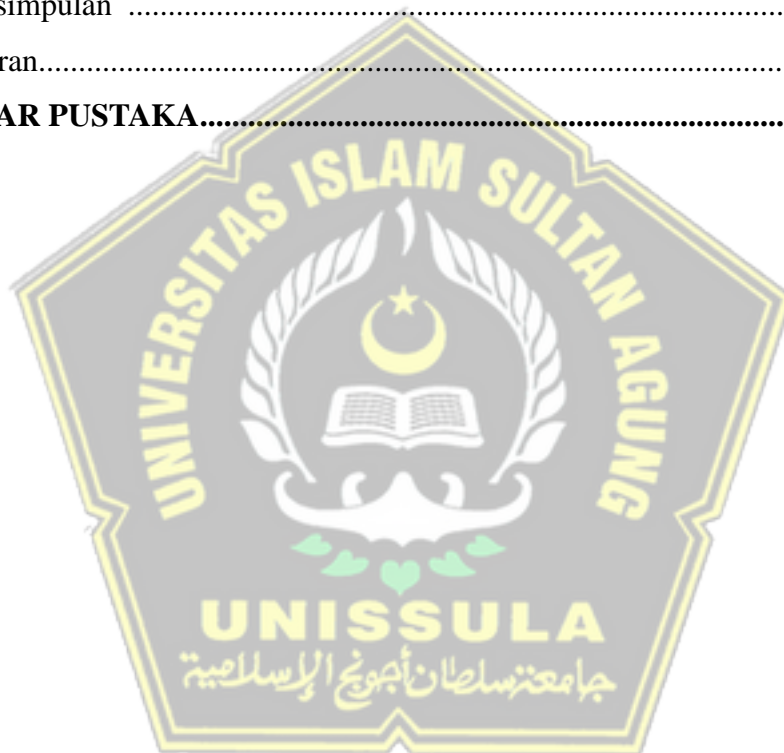
Eko Agung Prasetyo

DAFTAR ISI

HALAMAN JUDUL	
LEMBAR PENGESAHAN PEMBIMBING	i
LEMBAR PENGESAHAN PENGUJI	ii
SURAT PERNYATAAN KEASLIAN TUGAS AKHIR	iii
PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH	iv
KATA PENGANTAR	i
DAFTAR ISI	ii
DAFTAR GAMBAR	v
DAFTAR TABEL	vi
ABSTRAK	vii
<i>ABSTRACT</i>	vii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah	2
1.3 Pembatasan Masalah	2
1.4 Tujuan	3
1.5 Manfaat	3
1.6 Sistematika Penulisan	3
BAB II TINJAUAN PUSTAKA DAN DASAR TEORI	5
2.1 Tinjauan Pustaka	5
2.2 Dasar Teori	7
2.2.1 Badan Pusat Statistik Kota Semarang	7
2.2.2 Natural Language Processing	7
2.2.3 <i>Chatbot</i>	8
2.2.4 <i>Data Preprocessing</i>	8
2.2.4.1 Persiapan Data	8
2.2.4.2 Chunking.....	8
2.2.4.3 Embedding	9
2.2.4.4 Penyimpanan Embedding	9
2.2.4.5 Retriever dari Indeks	9

2.2.5	<i>Transformers</i>	9
2.2.5.1	Encoder dan Decoder	10
2.2.5.2	Attention	10
2.2.5.3	Scaled Dot-Product Attention	11
2.2.5.4	Multi-Head Attention	12
2.2.5.5	Position-wise Feed-Forward Networks	13
2.2.5.6	Embeddings and Softmax	13
2.2.5.7	Positional Encoding	14
2.2.6	<i>Large Language Model (LLM)</i>	14
2.2.7	<i>Zephyr-7B-beta-GGUF</i>	15
2.2.8	<i>Retrieval Augmented Generation (RAG)</i>	15
BAB III METODOLOGI PENELITIAN		18
3.1	Deskripsi Sistem	18
3.2	Studi Literatur	18
3.3	Pengumpulan Data	18
3.4	Rancangan Sistem	19
3.4.1	Pemodelan Sistem	19
3.4.2	Analisis Kebutuhan Sistem	20
3.5	<i>Preprocessing Data</i>	22
3.5.1	Persiapan Data	22
3.5.2	<i>Chunking</i>	22
3.5.3	<i>Embedding</i>	22
3.5.4	Menyimpan hasil Embedding	23
3.5.5	Membuat <i>Retriever</i> dari <i>Index</i>	23
3.5.6	<i>Prompt Engineering</i>	23
3.5.7	Inisialisasi LlamaCPP	24
3.6	Evaluasi Model	24
BAB IV HASIL DAN ANALISIS PENELITIAN		26
4.1	Hasil	26
4.1.1	Memuat file pdf	26
4.1.2	<i>Chunking</i>	27
4.1.3	Membuat <i>Embedding</i>	28
4.1.4	Menyimpan Hasil <i>Embedding</i>	29

4.1.5	Membuat <i>Retriever</i> dari Indeks	29
4.1.6	<i>Prompt engineering</i>	30
4.1.7	Inisialisasi LlamaCPP	31
4.1.8	Tampilan Aplikasi	31
4.2	Analisis Penelitian	36
4.2.1	Pengujian Sistem	37
4.2.2	Hasil evaluasi	38
BAB V KESIMPULAN DAN SARAN		39
5.1	Kesimpulan	39
5.2	Saran	39
DAFTAR PUSTAKA		40



DAFTAR GAMBAR

Gambar 2.1	Arsitektur Transformers.....	10
Gambar 2.2	Scale Dot-product Attention.....	11
Gambar 2.3	Multi head attention.....	11
Gambar 2.4	arsitektur RAG.....	15
Gambar 3.1	alur sistem Chatbot.....	19.
Gambar 4.1	Memuat file pdf.....	26
Gambar 4.2	proses chunking.....	27
Gambar 4.3	inisialisai model embedding.....	28
Gambar 4.4	proses embedding.....	28
Gambar 4.5	menyimpan hasil embedding.....	29
Gambar 4.6	membuat retriever.....	29
Gambar 4.7	hasil retriever.....	29
Gambar 4.8	prompt engineering.....	30
Gambar 4.9	Inisialisasi LlamaCPP.....	31
Gambar 4.10	tampilan sistem Chatbot.....	32
Gambar 4.11	tampilan Chatbot.....	33
Gambar 4.12	tampilan Chatbot.....	33
Gambar 4.13	tampilan Chatbot.....	34
Gambar 4.14	tampilan Chatbot.....	34
Gambar 4.15	tampilan Chatbot.....	35
Gambar 4.16	tampilan Chatbot.....	35
Gambar 4.17	tampilan Chatbot.....	36

DAFTAR TABEL

Tabel 1.1	Sistematika penulisan.....	3
Tabel 4.1	Pengujian black-box.....	37
Tabel 4.2	Hasil evaluasi	38



ABSTRAK

Pemerintah Kota Semarang menghadapi tantangan dalam mengakses dan mengelola informasi pembangunan wilayah secara efisien, yang sering memakan waktu jika dilakukan secara manual. Untuk mengatasi hal ini, penggunaan *Chatbot* berbasis metode *Retrieval-Augmented Generation* (RAG) dapat menjadi solusi yang efektif. RAG menggabungkan kemampuan pengambilan data dari berbagai sumber dan pembuatan teks yang sesuai, memungkinkan pegawai pemerintah mendapatkan informasi yang akurat dan tepat waktu. Hasil rata-rata evaluasi dengan ROUGE, chatbot ini menunjukkan nilai *precision* = 0.77, 0.65 dan 0.73, *recall* = 0.77, 0.68, 0.73, *f1-score* adalah 0.76, 0.65, 0.72 yang menunjukkan bahwa chatbot dapat bekerja dengan baik sehingga mempermudah pegawai PEMDA dalam mengakses informasi pembangunan wilayah kota Semarang dan mengurangi ketergantungan pada pencarian manual.

Kata kunci : Sistem *chatbot*, pembangunan wilayah kota Semarang, *Retrieval-Augmented Generation* (RAG)

ABSTRACT

Semarang City Government faces challenges in accessing and managing regional development information efficiently, which is often time-consuming if done manually. To overcome this, the use of Chatbot based on the Retrieval-Augmented Generation (RAG) method can be an effective solution. RAG combines the ability to retrieve data from various sources and create appropriate text, allowing government employees to get accurate and timely information. The average evaluation results with ROUGE, this chatbot shows a precision value = 0.77, 0.65 and 0.73, recall = 0.77, 0.68, 0.73, f1-score is 0.76, 0.65, 0.72 which shows that the chatbot can work well so that it makes it easier for Semarang City Government employees to access Semarang City regional development information and reduce dependence on manual searches.

Keywords : *Chatbot system, Semarang City regional development, Retrieval-Augmented Generation (RAG)*

BAB I PENDAHULUAN

1.1 Latar Belakang

Kemajuan teknologi informasi telah membawa dampak signifikan pada berbagai sektor, termasuk sektor pemerintahan (Cholik 2021). Sebagai ibu kota Provinsi Jawa Tengah, Kota Semarang terus berupaya meningkatkan kualitas dan kinerja aparatur pemerintah daerah dalam upaya mempercepat pembangunan dan pelayanan, pemerintah daerah kota Semarang sering kali dihadapkan pada tantangan dalam mengakses dan mengelola informasi pembangunan wilayah kota Semarang. Karena data yang sangat banyak, proses pencarian data secara manual akan banyak memakan waktu, oleh karena itu, diperlukan sistem yang mampu menyediakan informasi yang mudah diakses oleh pegawai pemerintah daerah (Muktiali 2020).

Metode *Retrieval-Augmented Generation* (RAG) merupakan teknik canggih yang menggabungkan kemampuan pengambilan informasi dan pembuatan teks dalam satu sistem, RAG bekerja dengan cara mengambil data yang relevan dari sumber informasi eksternal dan kemudian menggunakannya untuk menghasilkan teks atau jawaban, *Retrieval Augmented Generation* memungkinkan sistem untuk memberikan informasi yang lebih akurat dan kontekstual berdasarkan data yang ada, RAG dilatih dengan jumlah data yang besar dengan menggunakan teknik-teknik canggih seperti jaringan saraf untuk memahami kompleksitas bahasa. Metode ini telah digunakan dalam berbagai aplikasi seperti *Chatbot* AI, terjemahan mesin, dan ringkasan teks (Li dkk. 2022).

Penggunaan *Chatbot* dengan metode RAG pada sistem pemerintahan daerah kota Semarang dapat menjadi solusi yang efektif untuk mengatasi permasalahan pegawai pemerintahan daerah kota Semarang yang seringkali memakan banyak waktu dalam pencarian informasi pembangunan wilayah kota Semarang secara manual. *Chatbot* dapat membantu pegawai pemerintah kota Semarang dalam mendapatkan informasi terkait pembangunan wilayah kota Semarang, dengan kemampuan RAG untuk memahami bahasa manusia secara

alami, *Chatbot* dapat memberikan respon yang relevan dan mendalam terhadap berbagai pertanyaan dan permintaan informasi. RAG juga memungkinkan sistem untuk terus memperbarui pengetahuannya berdasarkan data terbaru yang diambil dari berbagai sumber.

Tujuan pengembangan *Chatbot* menggunakan metode *Retrieval Augmented Generation* ini adalah untuk mempermudah pegawai pemerintah daerah Kota Semarang guna mendapatkan informasi pembangunan wilayah kota Semarang. Selain itu, diharapkan *Chatbot* ini dapat mengurangi ketergantungan pada pencarian informasi pembangunan wilayah kota Semarang secara manual yang seringkali memakan waktu. Dengan metode RAG, *Chatbot* diharapkan mampu memberikan solusi yang tepat dalam menjawab kebutuhan informasi mengenai pembangunan wilayah kota Semarang.

1.2 Perumusan Masalah

1. Bagaimana memanfaatkan kemajuan teknologi kecerdasan buatan *Artificial Intelligence* (AI) untuk menciptakan *Chatbot* yang dapat membantu pegawai pemerintah daerah kota Semarang untuk mendapatkan informasi terkait pembangunan wilayah kota Semarang?
2. Bagaimana mengintegrasikan *Retrieval Augmented Generation* (RAG) ke dalam sistem *Chatbot* untuk meningkatkan kemampuan pemahaman dan respon *Chatbot* terhadap pertanyaan pengguna terkait pembangunan wilayah Kota Semarang?

1.3 Pembatasan Masalah

1. Fokus pada pengembangan *Chatbot* yang menggunakan metode *retrival augmented generation* untuk memberikan informasi terkait pembangunan wilayah Kota Semarang.
2. Pertanyaan dan jawaban menggunakan bahasa Indonesia dengan kata – kata sesuai dengan Kamus Besar Bahasa Indonesia (KBBI).

3. Data diperoleh dari file pdf yang berupa laporan tahunan BPS kota Semarang dari tahun 2019 sampai tahun 2024 yang dapat diakses melalui tautan <https://Semarangkota.beta.bps.go.id/id>.
4. Topik dialog dibatasi seputar informasi pembangunan wilayah Semarang dari tahun 2019 sampai 2024.
5. *Chatbot* tidak melayani masukan dalam bentuk perhitungan matematis, dan tidak menanggapi masukan yang berupa karakter.
6. *Chatbot* tidak membedakan lawan bicaranya berdasarkan identitas seperti jenis kelamin, umur, atau nama.

1.4 Tujuan

Tujuan penelitian ini adalah membangun sistem *Chatbot* yang memberikan informasi pembangunan wilayah Semarang dengan menggunakan metode *retrieval augmented generation* bagi pegawai pemerintahan daerah kota Semarang.

1.5 Manfaat

Manfaat sistem *Chatbot* ini guna membantu pegawai pemerintahan daerah kota Semarang dalam mendapatkan informasi mengenai pembangunan wilayah kota Semarang dari tahun 2019 sampai 2024.

1.6 Sistematika Penulisan

Sistematika penulisan yang akan digunakan oleh penulis dalam sebuah pembuatan laporan tugas akhir adalah sebagai berikut:

Tabel 1.1 Sistematika penulisan

BAB I	:	PENDAHULUAN
		Memuat tentang latar belakang masalah sehingga dapat diangkat sebagai judul penelitian, perumusan masalah untuk menguraikan masalah yang akan di pecahkan, Batasan masalah dibuat agar ruang lingkup pemecahan masalah tidak terlalu lebar, tujuan yang hendak dicapai, manfaat yang diperoleh dari pembuatan sistem tersebut dan sistematika penulisan yang berisi

		uraian dari penulisan laporan Tugas Akhir.
BAB II	:	TINJAUAN PUSTAKA DAN DASAR TEORI
		Memuat tentang tinjauan pustaka dan landasan teori yang digunakan untuk menunjang Analisa masalah sebagai acuan untuk menyusun Tugas Akhir.
BAB III	:	METODE PENELITIAN
		Memuat tentang Analisa proses sistem chatbot untuk informasi pembangunan wilayah kota Semarang menggunakan metode RAG. Analisa ini mencakup perancangan sistem.
BAB IV	:	HASIL DAN ANALISIS PENELITIAN
		Memuat tentang hasil pengujian program dan pembahasan program atau prosedur – prosedur kerja program, serta tampilan program.
BAB V	:	KESIMPULAN DAN SARAN
		Memuat tentang kesimpulan dan saran dari penulis terhadap penelitian yang telah dilakukan.



BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Chatbot merupakan aplikasi yang menggunakan pemrosesan bahasa alami (NLP), machine learning, kecerdasan buatan (AI), dan rekayasa perangkat lunak, yang mampu beroperasi dengan menerima *input* berupa bahasa alami, yang terdiri dari kata-kata yang tersusun dalam kalimat, serta mampu mempelajari *input* sebelumnya yang memiliki makna serupa (Rohman, Utami, and Raharjo 2019).

Penelitian yang dilakukan (Lommatzsch dkk.) berfokus pada peningkatan sistem *Chatbot* dengan mengintegrasikan teknik Information Retrieval (IR) dengan *Large Language Models* (LLM). Metode utama melibatkan menggabungkan jawaban yang dapat diandalkan dari teknik IR dengan kemampuan pembuatan bahasa alami LLM yang bertujuan untuk menghasilkan tanggapan yang akurat dan relevan secara kontekstual. Dapat disimpulkan bahwa integrasi metode IR dengan LLM menghasilkan jawaban yang lebih dapat diandalkan dan terdengar alami. Kombinasi ini memungkinkan *Chatbot* untuk memberikan tanggapan yang sesuai secara kontekstual sambil mempertahankan akurasi.

Dikutip dari jurnal (Wei dkk. 2024) menyelidiki bagaimana *Large Language Models* (LLM), khususnya GPT-3, dapat meningkatkan kinerja *Chatbot* dalam mengumpulkan data yang dilaporkan sendiri dari pengguna. Penelitian ini bertujuan untuk memahami dampak desain perintah yang berbeda pada efisiensi pengumpulan data *Chatbot* dan kualitas interaksi pengguna. Sebuah studi online dilakukan dengan 48 peserta yang berinteraksi dengan *Chatbot*. Para peneliti mengumpulkan data tentang bagaimana desain perintah yang berbeda memengaruhi kemampuan *Chatbot* untuk mengumpulkan informasi dan mempertahankan percakapan yang menarik. Hasilnya menunjukkan bahwa peneliti menemukan desain perintah dan topik percakapan secara signifikan mempengaruhi kinerja *Chatbot* dan pengalaman pengguna. Ini menunjukkan

bahwa pembuatan *prompt* yang cermat sangat penting untuk mengoptimalkan interaksi *Chatbot*.

penelitian yang dilakukan (Pichai 2023) mengeksplorasi bagaimana *Retrival Augmented Generation* (RAG) dapat digunakan untuk meningkatkan kinerja large language model (LLM), khususnya dalam konteks pengetahuan budaya dan ekologi dari Hutan Hujan Amazon. Hasil yang diperoleh RAG memungkinkan fleksibilitas dan pengoptimalan yang lebih besar dalam kinerja model bahasa.

Dikutip dari Jurnal (Radeva dkk. 2024) berfokus pada implementasi aplikasi web yang disebut PASSer, yang mengintegrasikan teknologi *Retrieval Augmented Generation* (RAG) dengan berbagai *Large Language Models* (LLM). Tujuan utamanya adalah untuk mengeksplorasi bagaimana teknologi RAG dapat diimplementasikan secara efektif menggunakan LLM *open-source*, khususnya Mistral: 7b, Llama 2:7 b, dan Orca 2:7 b. Hasil dari pengujian menunjukkan bahwa model Mistral:7b mengungguli yang lain dalam tes skor tanya jawab RAG, terutama ketika diterapkan pada kumpulan data dari 446 pasangan pertanyaan-jawaban, juga adanya GPU sangat penting untuk pembuatan teks yang cepat, dengan Mistral 7b.

Penelitian dari (Miao dkk. 2024) mengeksplorasi integrasi teknologi AI canggih dalam nefrologi, dengan fokus pada penggunaan sistem *Retrieval Augmented Generation* (RAG) bersama *Large Language Models* (LLM). Penelitian menekankan kombinasi sistem RAG dengan LLM untuk meningkatkan akurasi dan keandalan informasi yang diberikan dalam nefrologi. Integrasi ini bertujuan untuk mengatasi halusinasi dan ketidakakuratan pada LLM. Hasilnya bahwa model yang ditingkatkan RAG seperti *almanac* menunjukkan peningkatan yang signifikan dalam akurasi dan kepuasan pengguna dibandingkan dengan LLM standar.

2.2 Dasar Teori

2.2.1 Badan Pusat Statistik Kota Semarang

Badan Pusat Statistik (BPS) Kota Semarang adalah lembaga pemerintah di Indonesia yang bertugas mengumpulkan, mengelola, dan menyajikan data statistik di wilayah Kota Semarang. BPS berperan penting dalam menyediakan data dan informasi yang akurat dan terpercaya untuk mendukung perencanaan, pengambilan keputusan, dan evaluasi program pemerintah serta kepentingan umum lainnya (Muktiali 2019).

Adapun fungsi BPS kota Semarang adalah sebagai berikut:

1. Pengumpulan Data: BPS melakukan pengumpulan data statistik melalui berbagai survei dan sensus. Data yang dikumpulkan meliputi berbagai aspek seperti ekonomi, sosial, demografi, dan lingkungan.
2. Pengolahan dan Analisis Data: Setelah data dikumpulkan, BPS mengolah dan menganalisis data tersebut untuk menghasilkan informasi yang bermanfaat bagi pemerintah dan masyarakat.
3. Penyajian Data: BPS menyajikan data dalam bentuk laporan, publikasi, dan basis data yang dapat diakses oleh publik. Hal ini termasuk penyediaan data melalui *website* dan publikasi cetak.
4. Koordinasi Statistik: BPS juga berfungsi sebagai koordinator statistik di wilayahnya, memastikan bahwa data yang dikumpulkan dan dipublikasikan oleh berbagai instansi pemerintah sesuai dengan standar dan metodologi yang ditetapkan.

2.2.2 Natural Language Processing

Natural Language Processing (NLP) merupakan salah satu cabang ilmu kecerdasan buatan (AI) yang berfokus pada pengolahan bahasa natural. Bahasa natural adalah bahasa yang biasa digunakan manusia untuk berbicara satu sama lain. Dalam bidang *Natural Language Processing*, terdapat berbagai topik yang dibahas, salah satunya adalah parsing. Parsing adalah proses membedah kalimat menjadi bagian-bagian komponennya berdasarkan kategori kata (*parts of speech*) untuk melihat tata bahasanya. Analisis ini bertujuan untuk membantu tugas

pemrosesan yang lebih kompleks pada tahap selanjutnya (Rohman, Utami, and Raharjo 2019).

2.2.3 *Chatbot*

Chatbot merupakan perangkat yang bisa melakukan percakapan dengan pengguna baik melalui teks ataupun suara. Bidang penerapannya dari pendidikan, administrasi, hiburan, dan perawatan kesehatan hingga berbagai domain lainnya. Salah satu sistem *Chatbot* pertama yang dirancang pada tahun 60an, menggunakan pencocokan pola untuk menjawab pertanyaan pengguna. Sistem ini dirancang untuk meniru wawancara psikiatris, untuk memberikan pengguna perasaan didengarkan dan dipahami. Sistem menghasilkan jawaban dengan memilih kata dari masukan pengguna dan mengisi jawaban template menggunakan seperangkat aturan tetap. Karena terbatasnya jumlah aturan dan pola yang digunakan, responnya berulang setelah beberapa saat digunakan (Ferdian & Anwar, 2023).

2.2.4 *Data Preprocessing*

Pra-pemrosesan atau *preprocessing* adalah langkah-langkah yang dilakukan untuk mempersiapkan data mentah sebelum data tersebut dimasukkan ke dalam model *machine learning* atau analisis lebih lanjut. Tujuan dari pra-pemrosesan adalah untuk membersihkan dan mengubah data mentah sehingga lebih mudah dipahami dan diolah oleh model.

2.2.4.1 *Persiapan Data*

Mengumpulkan dokumen atau informasi yang relevan dari berbagai sumber seperti teks, PDF, *database*, atau API eksternal. Data ini akan digunakan sebagai sumber untuk melakukan pencarian selama proses RAG.

2.2.4.2 *Chunking*

adalah teknik untuk membagi teks yang panjang menjadi bagian-bagian yang lebih kecil yang disebut *chunk*. Tujuan dari *chunking* adalah untuk membuat teks lebih mudah diproses oleh model atau algoritma, terutama ketika bekerja dengan dokumen yang besar atau dataset yang sangat panjang.

2.2.4.3 *Embedding*

Embedding merupakan proses merubah teks menjadi representasi *vektor*. Tujuan *embedding* adalah mempermudah pencarian dan pencocokan data dengan menggunakan representasi vektor.

2.2.4.4 Penyimpanan *Embedding*

Penyimpanan *Embedding* merupakan Menyimpan hasil *embedding* dalam sebuah *vektor store* atau *database* yang dioptimalkan guna pencarian berbasis vektor.

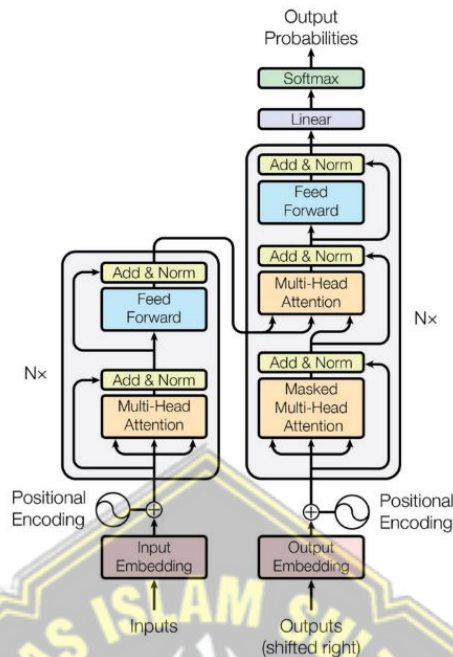
2.2.4.5 *Retriever* dari *Indeks*

Retriever merupakan cara untuk mencari informasi berdasarkan kueri pengguna dengan cara membandingkan kueri dengan informasi yang ada dalam *vector database* menggunakan *Cosine Similarity* dengan menghitung nilai kesamaan dua vektor berdasarkan sudut kosinus.

$$\cos(\theta) = \frac{A \cdot B}{|A||B|} = \frac{\sum_{i=0}^n A_i \cdot B_i}{\sqrt{\sum_{i=0}^n A_i^2} \cdot \sqrt{\sum_{i=0}^n B_i^2}} \quad (1)$$

2.2.5 *Transformers*

Transformers adalah model *deep learning* yang menggunakan mekanisme *self-attention* untuk menemukan hubungan antar kata dalam konteks. *Transformers* digunakan terutama dalam pemrosesan bahasa alami dan visi komputer. Model ini dirancang untuk *memproses* tipe data sekuens seperti bahasa alami, memungkinkannya melakukan tugas seperti menerjemahkan bahasa dan merangkum teks. Sebuah transformator terdiri dari dua stack (Al-Faruk, 2021).



Gambar 2.1 Arsitektur *Transformers*

Arsitektur umum ini diikuti oleh transformator dengan menggunakan lapisan *self-attention stacked dan point-wise*, sepenuhnya terhubung dengan *encoder* dan *decoder*, seperti yang ditunjukkan di bagian kiri dan kanan (Al-Faruq, 2021).

2.2.5.1 Encoder dan Decoder

Encoder terdiri dari $N = 6$ lapisan yang sama. Setiap lapisan terdiri dari dua sublapisan. Mekanisme *self-attention multi-head* adalah lapisan pertama, dan jaringan *feed-forward* yang sederhana dan terhubung sepenuhnya secara posisi adalah lapisan kedua.

Decoder terdiri dari tumpukan $N = 6$ lapisan yang sama. *Encoder* menambah lapisan ketiga di bawah dua lapisan *encoder*, yang berfungsi melakukan *multi-head attention* terhadap *output* dari tumpukan *encoder*. (Pratiwi dan Pardede, 2022)

2.2.5.2 Attention

Secara sederhana, fungsi *attention* dapat dijelaskan sebagai proses di mana *query* dipasangkan dengan *key-value* untuk menghasilkan *output*. Dalam konteks ini, *query*, *key*, *value*, dan *output* direpresentasikan sebagai vektor, dan hasil akhir dihitung berdasarkan bobot dari nilai-nilai yang terlibat. Selain itu, tingkat

pertanyaan. Hasilnya adalah matriks perhatian menunjukkan relevansi elemen dalam data. *query*, *key*, dan *value* dikemas dalam matriks Q, K, dan V secara berurutan. (Pratiwi dan Pardede, 2022) Matriks keluaran dihitung sebagai berikut:

$$Attention(Q, K, V) = softmax \left(\frac{QK^T}{\sqrt{d_K}} \right) V \quad (2)$$

Keterangan :

Q = Kueri

K = Kunci

V = Nilai

d_k = Kueri dan Kunci dimensi

d_v = Nilai dari dimensi

2.2.5.4 Multi-Head Attention

Multi-Head Attention menerima tiga vektor sebagai *input*: vektor *query*, *key*, dan *value*. Sebelum melakukan perhitungan perhatian, setiap vektor *query*, *key*, dan *value* diubah ke dalam subruang melalui proyeksi *linier*. Untuk proyeksi ini, digunakan *matriks* pembobotan yang berbeda untuk setiap kepala. Setiap kepala menghitung skor perhatian sendiri, menghasilkan matriks perhatian yang berbeda. Hal ini dilakukan dengan mengalikan vektor *query* dan *key* setelah proyeksi *linier*. Selanjutnya, *softmax* dan *scaling* digunakan untuk perhitungan perhatian. Matriks perhatian akhir dibuat dengan menggabungkan hasil dari setiap kepala. Untuk hasil yang lebih halus, proyeksi linier tambahan dapat digunakan. Seperti pada *Scaled Dot-Product Attention*, matriks perhatian digunakan untuk mengambil rata-rata tertimbang dari vektor *value*, menghasilkan vektor hasil yang menunjukkan interaksi antara komponen dalam data *input* (Pratiwi dan Pardede, 2022).

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^o \quad (3)$$

$$where head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Keterangan :

Multihead(Q, K, V) = operasi *multi-head attention* mengambil tiga *input*, yaitu matriks *query* (Q), matriks *key* (K), dan matriks *value* (V).

$Concat(head_1, \dots, head_h) =$ operasi penggabungan (*concatenation*) dari hasil-hasil dari setiap kepala (*head*) yang diberikan oleh fungsi *Attention*. Hasil dari setiap kepala dikonkatenasi menjadi satu vektor besar.

$W^o =$ matriks pembobot akhir yang digunakan untuk mentransformasi hasil penggabungan (*Concat*) dari kepala-kepala *attention* menjadi representasi akhir.

$head_i = Attention(QW_i^q, KW_i^K, VW_i^V,)$ perhitungan *attention* pada masing-masing kepala (*head*). Setiap kepala memiliki matriks pembobot sendiri, yaitu QW_i^q, KW_i^K, VW_i^V yang digunakan untuk mengubah matriks *query* (Q), *key* (K), dan *value* (V) menjadi representasi yang sesuai untuk kepala tersebut.

2.2.5.5 Position-wise Feed-Forward Networks

Selain sub-lapisan perhatian, masing-masing lapisan *encoder* dan *decoder* terdiri dari jaringan *feed-forward* yang terhubung sepenuhnya, yang diterapkan secara terpisah dan identik ke setiap posisi. Ini terdiri dari dua transformasi linier yang masing-masing memiliki aktivasi.

$$FFN(x) = \max(0, xW_1 + B_1)W_2 + b_2 \quad (4)$$

Transformasi linier memiliki parameter yang berbeda di setiap lapisan, meskipun mereka identik pada posisi yang berbeda. Dua konvolusi dengan ukuran kernel 1 adalah cara lain untuk menggambarkan hal ini (Pratiwi dan Pardede, 2022).

2.2.5.6 Embeddings and Softmax

Dengan menggunakan *embeddings* yang dipelajari, model ini mengonversi token *input* dan *output* menjadi vektor berdimensi. Selain itu, model ini menggunakan transformasi linier dan fungsi *softmax*, yang biasa digunakan untuk mengkonversi *output decoder* ke kemungkinan token berikutnya. Model ini menggunakan matriks bobot yang sama antara dua lapisan penyisipan dan transformasi linier *pre-softmax*. Pada lapisan penyisipan, model mengalikan bobot-bobot tersebut dengan vektor-vektor berdimensi (Pratiwi dan Pardede, 2022).

2.2.5.7 *Positional Encoding*

Untuk menggunakan urutan dalam model ini yang tidak memiliki perulangan dan konvolusi, diperlukan penyisipan informasi posisi token relatif atau absolut dalam urutan. Untuk mencapainya, "*positional encodings*" harus dimasukkan ke dalam *input* di bagian bawah *encoder* dan *decoder*. Dimensi *Positional Encodings* sama dengan vektor *embedding*, memungkinkan keduanya untuk digabungkan. Ada berbagai pilihan untuk *positional encodings* yang dipelajari dan diperbaiki. Dalam konteks tugas ini, fungsi sinus dan kosinus digunakan dengan frekuensi yang berbeda (Pratiwi dan Pardede, 2022).

2.2.6 *Large Language Model (LLM)*

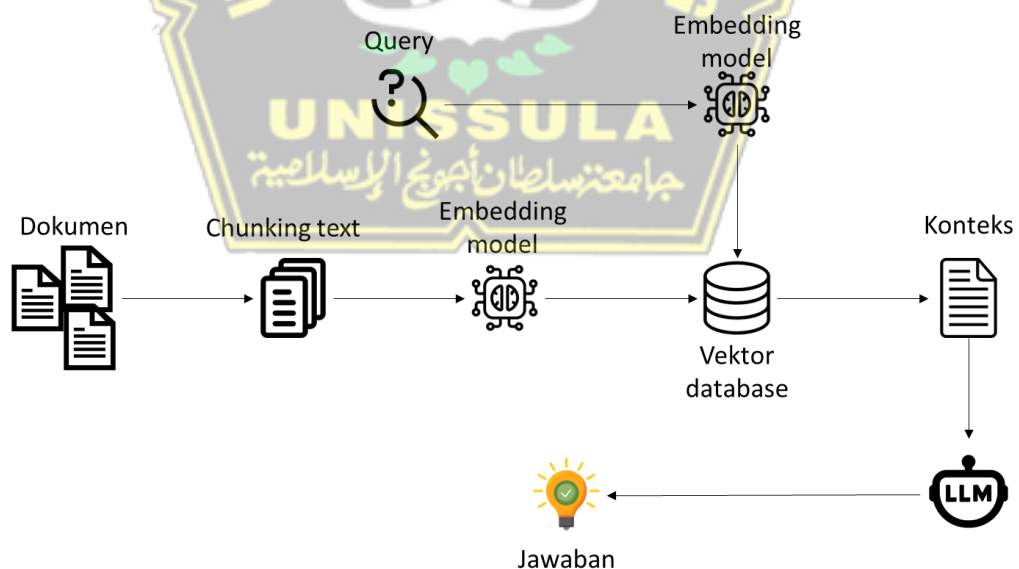
Large Language Model (LLM) adalah model *deep learning* berskala besar yang telah dilatih sebelumnya dengan sejumlah besar data. Model ini didasarkan pada transformator, yang merupakan rangkaian jaringan neural yang terdiri dari *encoder* dan *decoder* dengan kemampuan perhatian yang mandiri. LLM memainkan peran yang sangat penting dan signifikan dalam pengembangan *Natural Language Processing (NLP)*. Salah satu kemampuannya adalah memahami dan menghasilkan teks yang mirip dengan bahasa manusia. Penelitian oleh menunjukkan bahwa LLM dapat diterapkan dalam layanan kesehatan, yang merupakan aplikasi vital yang sangat terkait dengan kehidupan manusia. Misalnya, ChatGPT dan LLM lainnya telah digunakan dalam domain medis dan terbukti mampu menangani tugas tugas perawatan kesehatan seperti konsultasi dan saran medis. LLM belajar dari *pre-trained* pengawasan pada data teks berskala besar. Penelitian ini mengulas kemajuan terbaru dalam LLM, memperkenalkan konsep-konsep utama, temuan, serta teknik-teknik untuk memahami dan memanfaatkan LLM. Survei ini juga membahas empat aspek penting dalam LLM, yaitu pra-pelatihan, adaptasi, pemanfaatan, dan evaluasi. Mengenai arsitektur model yang digunakan, skalabilitas dan efektivitas transformer sudah menjadi arsitektur *de facto* dalam pengembangan LLM. (Liu dkk. 2023).

2.2.7 Zephyr-7B-beta-GGUF

zephyr-7B-beta-GGUF adalah sebuah model bahasa alami (*Natural Language Processing* atau NLP) dengan 7 miliar parameter, yang dikonversi ke format GGUF, Konversi ke GGUF membuat model *Zephyr-7B-beta* lebih efisien untuk digunakan pada perangkat dengan sumber daya terbatas sehingga model dapat berjalan lebih cepat dan menggunakan lebih sedikit RAM. Model tersebut merupakan bagian dari seri *Zephyr* yang dirancang untuk bertindak sebagai asisten yang membantu. Model ini merupakan versi *fine-tuned* dari *Mistral-7B-v0.1*, dilatih pada kombinasi dataset publik dan sintetis menggunakan *Direct Preference Optimization* (DPO) (Tunstall dkk 2023).

2.2.8 Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) merupakan metode untuk meningkatkan *Large Language Models* (LLMs) dengan menggabungkan data informasi tambahan dari sumber eksternal. Hal ini dapat meminimalisir halusinasi dan meningkatkan respon LLM yang lebih tepat dan sesuai konteks. RAG sangat berguna dalam pembuatan tugas-tugas seperti menjawab pertanyaan, meringkas dokumen (Miao dkk. 2024).



Gambar 2.4 Arsitektur RAG

Proses *indexing* dilakukan untuk membersihkan dan mengekstrak data awal terlebih dahulu kemudian mengubah berbagai format file berformat PDF,

HTML dan *Word* menjadi teks sederhana. Untuk menghindari kendala, teks dibagi menjadi bagian yang lebih kecil dan mudah diatur, proses tersebut disebut dengan *chunking*. Teks yang sudah dibagi kemudian dirubah menjadi representasi vektor dengan menggunakan model *embedding*. Selanjutnya indeks dibuat untuk menyimpan potongan teks ini dan menyematkan vektor sebagai kunci guna melakukan pencarian yang tepat (Miao dkk. 2024).

Proses *retrieval* adalah proses mengambil informasi tambahan yang relevan dari sumber pengetahuan eksternal menggunakan kueri pengguna. Dalam proses ini, model pengkodean akan digunakan untuk memproses kueri pengguna yang menghasilkan penyematan semantik. Selanjutnya, pencarian kesamaan dilakukan pada vektor *database* untuk menemukan objek terdekat (Miao et al. 2024).

Generasi adalah proses menggabungkan perintah masukan dengan dokumen atau informasi yang diambil dari sumber pengetahuan lainnya. Dalam proses generasi bahasa diperlukan adanya model yang dapat mengubah informasi yang ditemukan dalam tahap *retrieval*. Model generasi bahasa seperti *Generative Pre-trained Transformer* (GPT) digunakan untuk membuat jawaban atau konten baru berdasarkan informasi yang ditemukan dalam tahap *retrieval*. Pendekatan RAG telah terbukti berhasil dalam berbagai tugas proses pengolahan bahasa, seperti menjawab pertanyaan, membuat teks informatif, dan mendukung pengambilan keputusan berbasis informasi. Dengan integrasi kedua tahap ini, sistem dapat menghasilkan jawaban yang lebih akurat, informatif, dan relevan terhadap permintaan pengguna, terutama dalam kasus di mana informasi yang mendalam atau spesifik diperlukan dari kumpulan dokumen yang sangat besar (Miao dkk. 2024).

Dalam pemrosesan bahasa alami, "*chunking*" mengacu pada pembagian teks menjadi "potongan" kecil, ringkas, dan bermakna. Sistem RAG dapat menemukan konteks yang relevan dalam potongan teks yang lebih kecil lebih cepat dan akurat daripada dalam dokumen yang lebih besar. Potongan yang lebih kecil menangkap lebih sedikit konteks, tetapi mungkin tidak sepenuhnya menangkap konteks yang diperlukan, meskipun potongan lebih besar dapat

menangkap lebih banyak konteks namun lebih banyak memakan waktu dan biaya komputasi untuk diproses. Salah satu cara untuk menyeimbangkan kendala ini adalah dengan menggunakan potongan tumpang tindih. Metode ini memungkinkan kueri untuk mengumpulkan data yang relevan dibanyak vektor untuk menghasilkan tanggapan kontekstual yang tepat.



BAB III

METODOLOGI PENELITIAN

3.1 Deskripsi Sistem

Penelitian ini menggunakan metode *Retrieval Augmented Generation* (RAG) dan integrasi dengan model *zephyr-7B-beta-GGUF* yang memungkinkan sistem mengakses data yang ada dalam basis data maupun korpus teks yang besar. Sistem ini efektif untuk mengekstra informasi dari yang relevan dari sumber yang tersedia seperti artikel, buku teks, laporan. Keluaran dari sistem ini adalah *Chatbot* yang mampu berkomunikasi dengan pengguna dengan memberikan respon teks jawaban mengenai pembangunan wilayah Semarang.

3.2 Studi Literatur

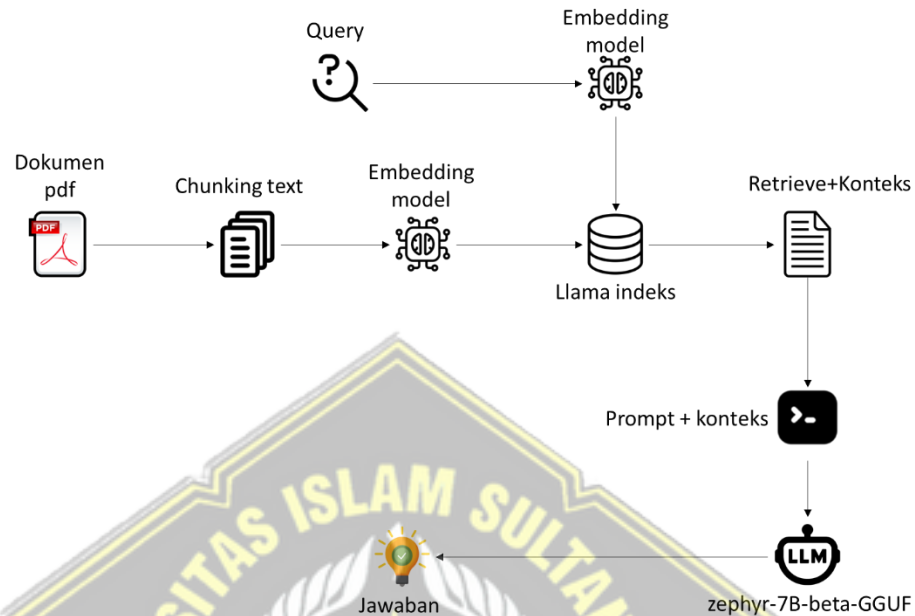
Dalam penelitian ini akan dilakukan tinjauan terhadap beberapa *e-book*, makalah, jurnal, tesis, dan skripsi terdahulu akan diulas selain mengunjungi berbagai situs *web*. Tujuan dari tinjauan ini adalah untuk mempelajari teori *Retrieval Augmented Generation* (RAG).

3.3 Pengumpulan Data

Data diperoleh dari file pdf yang berupa laporan tahunan Badan Pusat Statistik (BPS) kota Semarang dari tahun 2019 sampai tahun 2024. Data ini bisa diakses secara publik melalui tautan <https://Semarangkota.beta.bps.go.id/id>. Laporan tersebut mencakup beberapa bidang Pembangunan yaitu : Geografi dan Iklim, Pemerintahan, Penduduk dan Ketenagakerjaan, Sosial dan Kesejahteraan Rakyat, Pertanian dan Perikanan, Pariwisata, Transportasi dan Komunikasi.

3.4 Rancangan Sistem

3.4.1 Pemodelan Sistem



Gambar 3.1 Alur Sistem *Chatbot*

Rancangan alur sistem pada gambar 3.1 memiliki beberapa langkah yaitu pada tahap pertama yaitu Persiapan dataset, dataset yang digunakan berupa file berformat PDF yang berupa laporan tahunan Badan Pusat Statistik (BPS) kota Semarang dari tahun 2019 sampai tahun 2024. Tahap selanjutnya adalah *chunking document*, tahap ini mengurai atau memecah teks menjadi bagian-bagian yang lebih kecil dan terstruktur sehingga informasi penting dapat dimanfaatkan dalam pembuatan *Chatbot*. Data laporan tahunan Badan Pusat Statistik (BPS) kota Semarang yang sudah melewati proses *chunking* selanjutnya masuk pada tahap *embedding* guna mengubah teks menjadi representasi vektor. Model *embedding* yang digunakan adalah bagian dari koleksi model *embedding* yang tersedia di platform *HuggingFace* dengan nama "BAAI/bge-small-en-v1.5". Model ini dibuat oleh *Beijing Academy of Artificial Intelligence* (BAAI). Model mampu menghasilkan *embedding* teks yang berkualitas tinggi. Hasil dari *embedding* selanjutnya akan disimpan kedalam Indeks yang merupakan kelas yang berfungsi untuk membuat dan mengelola indeks berbasis vektor. Indeks ini digunakan untuk menyimpan representasi vektor dari data teks, yang memungkinkan pencarian dan

pemrosesan teks yang efisien berdasarkan kesamaan semantik. Pada saat pengguna memasukan kueri, kueri tersebut akan melewati proses *embedding* guna merubah kueri menjadi *vektor*, setelah itu adalah proses *retrieve* guna mencari persamaan dengan data vektor yang ada pada Llama indeks, hasil dari pencarian persamaannya disebut konteks, yang digunakan untuk proses *prompt engineering* guna mengarahkan atau mengendalikan yang dihasilkan. Selanjutnya adalah generasi jawaban, sistem akan menjawab pertanyaan yang dibuat. Model yang digunakan dalam proses generasi adalah *zephyr-7B-beta-GGUF*. Hasil dari generasi akan keluar sebagai jawaban berupa teks yang diberikan pada pengguna.

3.4.2 Analisis Kebutuhan Sistem

Pada tahap analisis kebutuhan, peneliti memeriksa semua perangkat lunak yang diperlukan untuk membuat aplikasi *Chatbot* ini beroperasi dengan baik dan menghasilkan hasil yang diinginkan. Sistem ini dibuat dengan menggunakan program berikut:

1. *Python 3.12*

Python adalah bahasa pemrograman tingkat tinggi yang memiliki sintaksis yang sederhana untuk dibaca dan dapat digunakan di berbagai platform. Penelitian ini menggunakan *Python* versi 3.12 karena bersifat *open source*, komunitasnya besar, dan ada banyak sumber daya *online*. *Python* menjadi pilihan populer untuk pengembangan perangkat lunak karena fokusnya pada produktivitas, manajemen memori otomatis, dan integrasi mudah dengan bahasa lain.

2. *Library llama-cpp-python*

llama-cpp-python adalah sebuah *library* *Python* yang dirancang untuk berinteraksi dengan model bahasa LLaMA (Large Language Model Meta AI) dari Meta. *Library* ini memudahkan pengguna dalam mengintegrasikan model-model LLaMA ke dalam aplikasi *Python* mereka dengan cara yang lebih sederhana dan efisien. Pada penelitian ini *library llama-cpp-python* digunakan untuk mengintegrasikan *Large Language Models*, yaitu *Zephyr*, dengan *Llama Index*

3. *Library llama-index*

llama-index adalah sebuah *library* yang dirancang untuk mempermudah proses pembuatan dan penggunaan indeks dalam aplikasi yang menggunakan *Large Language Models*, khususnya model-model seperti LLaMA. *Library* ini dirancang untuk bekerja dengan berbagai jenis data dan memudahkan pengelolaan informasi dalam konteks pemrosesan bahasa alami. *llama-index-llms-llama-cpp*. Pada penelitian ini *library* *llama-index* digunakan memuat dokumen pdf dengan menggunakan fungsi *SimpleDirectoryReader* guna membaca beberapa dokumen PDF kemudian memuatnya sebagai objek data yang dapat diproses lebih lanjut, *library* *llama-index* juga digunakan untuk memecah dokumen pdf menjadi *node* atau potongan teks yang lebih kecil, dan *library* *llama-index* digunakan untuk pencarian dan *query* Menggunakan fungsi *retriever* dan *query_engine* untuk memungkinkan pencarian teks dan menjawab pertanyaan berdasarkan isi dokumen yang sudah disimpan kedalam *vectorstoreindex*.

4. *Library llama-index-embeddings-huggingface*

llama-index-embeddings-huggingface adalah sebuah *library* yang mengintegrasikan kemampuan indeks dan embedding dari *llama-index* dengan model *embedding* dari *HuggingFace*. *Library llama-index-embeddings-huggingface* memungkinkan pengguna untuk menggunakan model *embedding* dari *HuggingFace* dalam proses pembuatan dan pengelolaan indeks menggunakan *llama-index*. Pada penelitian ini *library llama-index-embeddings-huggingface* digunakan untuk untuk menghasilkan representasi numerik (*embeddings*) dari teks yang ada dalam dokumen pdf

5. *Library llama-index gradio*

llama-index-gradio adalah sebuah *library* yang mengintegrasikan *llama-index* dengan *Gradio*, sebuah *library Python* yang memungkinkan pembuatan antarmuka pengguna berbasis *web* secara cepat dan mudah. Pada penelitian ini *llama-index-gradio* digunakan untuk membuat antar muka sistem *Chatbot*.

6. *Google Colaboratory*

Google Colaboratory menyediakan lingkungan *cloud* yang memungkinkan pengembangan dan pelatihan model di lingkungan dengan akses GPU atau TPU, sangat penting untuk model-model berat seperti *transformers*. Dalam penelitian ini digunakan untuk menulis dan menjalankan kode program, serta menyimpan dan mendokumentasikan.

3.5 *Preprocessing Data*

3.5.1 *Persiapan Data*

Diawali dengan membuat sebuah daftar yang berisi path atau lokasi dari file PDF yang ingin diproses. Setiap elemen dalam daftar tersebut adalah *path* menuju file PDF yang berbeda dengan menggunakan fungsi *SimpleDirectoryReader* untuk membaca konten dari semua file PDF yang telah dimasukkan ke dalam daftar *pdf_files*. Fungsi *load_data()* adalah metode yang digunakan untuk membaca data dari file PDF tersebut. Metode ini akan mengembalikan data dalam bentuk daftar objek dokumen, yang berisi teks dari masing-masing file PDF.

3.5.2 *Chunking*

Setelah data file pdf sudah dimuat, selanjutnya yaitu mengubah teks dari dokumen pdf menjadi unit-unit yang lebih kecil, yang disebut *nodes*, menggunakan teknik *chunking* (pemecahan teks menjadi bagian-bagian kecil). Ini dilakukan dengan menggunakan fungsi *SentenceSplitter*, sebuah parser yang membagi teks berdasarkan panjang tertentu.

3.5.3 *Embedding*

Setelah melewati proses *Chunking* selanjutnya adalah tahap *embedding* guna merubah teks menjadi representasi vektor menggunakan fungsi *HuggingFaceEmbedding* untuk menginisialisasi model *embedding* BAAI/bge-small-en-v1.5, yang kemudian digunakan untuk menghitung vektor *embedding* dari teks di setiap *node* dengan menggunakan fungsi *get_text_embedding*. Proses ini dilakukan melalui iterasi pada setiap node dalam daftar *nodes*, di mana teks diambil menggunakan *node.get_content()*, lalu dihitung *embedding*-nya. Hasil

akhirnya adalah daftar *node_embeddings* yang berisi vektor *embedding* yang mewakili teks dari masing-masing *node*.

3.5.4 Menyimpan hasil Embedding

Setelah melewati proses *embedding* kemudian representasi vektor yang merupakan hasil *embedding* disimpan kedalam Indeks yang merupakan kelas untuk membuat dan mengelola indeks berbasis vektor. membuat sebuah indeks dari data yang telah dirubah menjadi vektor dengan menggunakan fungsi *VectorStoreIndex*, di mana nodes yang berisi teks yang telah dihitung *embedding*-nya, serta *embedding_model* yang digunakan untuk menghasilkan *embedding* tersebut, disertakan sebagai parameter. Hasilnya adalah objek *index* yang menyimpan representasi vektor dari teks untuk memungkinkan pencarian kesamaan berbasis vektor.

3.5.5 Membuat *Retriever* dari *Index*

Tahap selanjutnya yaitu membuat *retriever* dari *index* yang sebelumnya telah dibuat guna menyimpan representasi vektor, dengan menggunakan fungsi *as_retriever* pada *index* dan menetapkan parameter *similarity_top_k=5*, hasilnya akan keluar 5 informasi paling mirip dengan kueri pencarian. Pencarian informasi tersebut menggunakan *cosine similarity* dengan menghitung nilai kesamaan dua vektor berdasarkan sudut kosinus.

3.5.6 *Prompt Engineering*

Setelah berhasil membuat *retriever*, selanjutnya yaitu membuat *prompt* untuk merancang, menyusun, atau memodifikasi *input (prompt)* yang diberikan kepada model bahasa *Zephyr-7B-beta-GGUF* untuk mengarahkan atau mengendalikan *output* yang dihasilkan dengan mendefinisikan kelas *ChatMessage* untuk merepresentasikan sebuah pesan dalam *chat*, yang memiliki atribut *role* (peran pengirim, seperti pengguna) dan *content* (isi pesan). Fungsi *zephyr_messages_to_prompt* digunakan untuk mengubah urutan pesan *chat messages* menjadi sebuah *prompt* yang akan diberikan kepada model Zephyr, memulai dengan menyiapkan *prompt* dasar yang berisi instruksi dalam bahasa Indonesia, yang menekankan pentingnya menjawab pertanyaan dengan singkat, jelas, dan hanya berdasarkan informasi yang terdapat dalam dokumen, serta

menolak memberikan informasi jika tidak ditemukan dalam dokumen. Kemudian, fungsi ini menambahkan setiap pesan dari urutan *messages* ke dalam *prompt*, dengan format khusus yang mencantumkan *role* pengirim diikuti oleh isi pesan (*content*). Hasil akhirnya adalah *prompt* teks siap digunakan guna memandu model *Zephyr* dalam menjawab pertanyaan.

3.5.7 Inisialisasi LlamaCPP

Tahap selanjutnya adalah menginisialisasi LlamaCPP untuk memuat dan menjalankan model *Zephyr-7B-beta* dari *path* lokal */content/zephyr-7b-beta.Q4_K_M.gguf*, dengan pengaturan *temperature* sebesar 0.1 untuk menghasilkan jawaban yang konsisten, dan membatasi jumlah token baru yang dihasilkan hingga 500 token, serta mempertimbangkan hingga 3900 token dalam jendela konteks. Sebanyak 24 lapisan model dijalankan di GPU untuk memanfaatkan memori GPU yang tersedia, dan fungsi *zephyr_messages_to_prompt* digunakan untuk mengonversi pesan ke dalam format yang sesuai untuk diproses oleh model.

3.6 Evaluasi Model

Evaluasi mode yang digunakan untuk mengukur berbagai metrik yang relevan adalah *Recall-Oriented Undersity for Gisting Evaluation* (ROUGE) merupakan kumpulan metrik yang dirancang guna mengevaluasi ringkasan otomatis daari teks-teks panjang yang terdiri dari bebrapa kalimat maupun paragraf. ROUGE-N dan ROUGE-L akan digunakan sebagai metrik untuk mengevaluasi kinerja sistem dalam penelitian ini. Hal ini melibatkan pengukuran kinerja model, evaluasi model dilakukan dengan metode pengukuran *precision*, *recall*, dan *F1-score*. Berikut adalah persamaan menghitung ROUGE-N dan ROUGE-L.

1. Precision

Precision adalah metode untuk mengukur jumlah yang diprediksi relevan dengan menghitung kata yang sama, baik unigram, bigram maupun LCS kemudian dibagi keseluruhan kata pada ringkasan sistem.

$$ROUGE - 1 = \frac{\text{Jumlah unigram kata sama}}{\text{Keseluruhan kata diringkasan sistem}} \quad (5)$$

$$ROUGE - 2 = \frac{\text{Jumlah biagram kata sama}}{\text{Keseluruhan kata diringkasan sistem}} \quad (6)$$

$$ROUGE - L = \frac{\text{LCS (Longest Common Subquent)}}{\text{Keseluruhan kata diringkasan sistem}} \quad (7)$$

2. Recall

Recall adalah metode untuk mengukur jumlah yang diprediksi relevan dengan menghitung kata yang sama, baik unigram, biagram maupun LCS kemudian dibagi keseluruhan kata pada ringkasan manusia.

$$ROUGE - 1 = \frac{\text{Jumlah unigram kata sama}}{\text{Keseluruhan kata diringkasan manusia}} \quad (8)$$

$$ROUGE - 2 = \frac{\text{Jumlah biagram kata sama}}{\text{Keseluruhan kata diringkasan manusia}} \quad (9)$$

$$ROUGE - L = \frac{\text{LCS (Longest Common Subquent)}}{\text{Keseluruhan kata diringkasan manusia}} \quad (10)$$

3. F1-Score

F1-Score merupakan metode untuk mengukur nilai rata-rata antara *recall* dan *precision*.

$$F1\text{-score} = 2x \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (11)$$

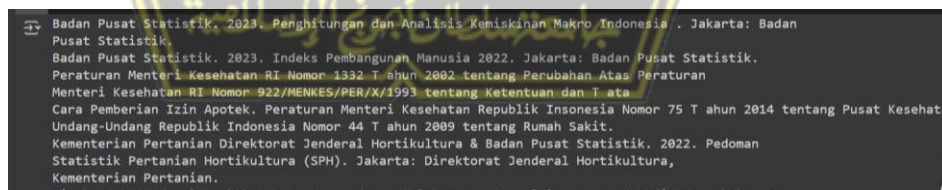
BAB IV

HASIL DAN ANALISIS PENELITIAN

4.1 Hasil

Data yang sudah diperoleh dari file pdf yang berupa laporan tahunan Badan Pusat Statistik (BPS) kota Semarang dari tahun 2019 sampai tahun 2024 yang dapat diakses secara publik melalui tautan <https://Semarangkota.beta.bps.go.id/id> kemudian data tersebut akan melewati proses *preprocessing* data yang berguna untuk mempermudah dalam proses pembuatan *chatbot*. Dimulai dengan memuat file pdf kedalam direktori google colab selanjutnya melakukan proses *chunking* guna memecah file pdf menjadi teks-teks kecil agar mempermudah model, kemudian hasil dari *chunking* akan diubah menjadi representasi numerik atau vektor yang disebut dengan proses *embedding*. Kemudian hasil dari *embedding* akan disimpan kedalam indeks. Tahap selanjutnya yaitu membuat *retriever* yang berfungsi untuk mencari informasi berdasarkan kueri pengguna. Setelah berhasil membuat *retriever* selanjutnya adalah membuat *prompt* untuk mengatur model agar jawaban yang diberikan sesuai keinginan. Terakhir adalah menginisialisasi model kedalam LlamaCPP untuk memuat dan menjalankan model.

4.1.1 Memuat file pdf

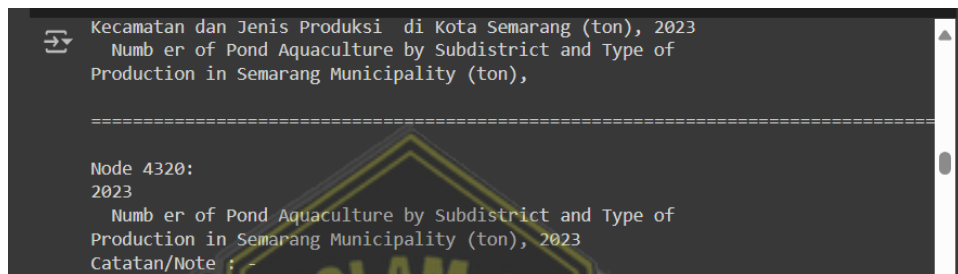


Gambar 4. 1 Memuat File Pdf

Pada gambar 4.1 diatas merupakan contoh hasil untuk memuat daftar file PDF yang telah dibaca, yang disimpan dalam variabel *pdf_files*. File PDF berupa laporan tahunan berjudul "Kota Semarang Dalam Angka" dari tahun 2019 hingga 2024. Kemudian fungsi `SimpleDirectoryReader(input_files=pdf_files).load_data()` digunakan untuk memuat data dari file PDF tersebut dan menyimpannya dalam variabel

documents. Setelah itu, dilakukan iterasi melalui setiap dokumen dalam *documents*, di mana untuk setiap dokumen, kode menampilkan seluruh teks yang ada di dalam dokumen tersebut menggunakan fungsi *print(doc.text)*. Setiap dokumen yang telah ditampilkan dipisahkan dengan garis pemisah yang terdiri dari 100 tanda sama dengan (=) untuk membedakan isi antar dokumen.

4.1.2 *Chunking*



```

Kecamatan dan Jenis Produksi di Kota Semarang (ton), 2023
Number of Pond Aquaculture by Subdistrict and Type of
Production in Semarang Municipality (ton),
=====

Node 4320:
2023
Number of Pond Aquaculture by Subdistrict and Type of
Production in Semarang Municipality (ton), 2023
Catatan/Note :

```

Gambar 4.2 Proses *Chunking*

Pada Gambar 4.2 merupakan hasil dari proses memecah teks dari dokumen menjadi potongan-potongan kecil yang disebut "*node*" dan kemudian menampilkan isi dari setiap *node* tersebut satu per satu. Pertama-tama membuat objek parser dari kelas *SentenceSplitter*, yang dikonfigurasi untuk memecah teks menjadi potongan-potongan dengan ukuran maksimal 256 kata (*chunk_size=256*), dan dengan tumpang tindih sebanyak 50 kata (*chunk_overlap=50*) antara potongan yang berurutan. Selanjutnya, objek parser digunakan untuk memproses teks dari dokumen yang telah dimuat sebelumnya, menghasilkan daftar potongan-potongan teks yang disebut "*nodes*". Setelah daftar *nodes* terbentuk, sebuah *loop* digunakan untuk iterasi melalui seluruh *node*. Pada setiap iterasi, kode menampilkan nomor urut *node* dan teks yang terkandung dalam *node* tersebut menggunakan *node.get_content()*. Setiap *node* yang telah ditampilkan dipisahkan oleh garis pemisah yang terdiri dari 100 tanda sama dengan (=) untuk membedakan isi antar *node*.

4.1.3 Membuat *Embedding*

```

/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning: The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens). You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models.
warnings.warn(
modules.json: 100% ██████████ 349/349 [00:00<00:00, 30.2kB/s]
config_sentence_transformers.json: 100% ██████████ 124/124 [00:00<00:00, 10.5kB/s]
README.md: 100% ██████████ 94.8k/94.8k [00:00<00:00, 6.52MB/s]
sentence_bert_config.json: 100% ██████████ 52.0/52.0 [00:00<00:00, 4.38kB/s]
config.json: 100% ██████████ 743/743 [00:00<00:00, 66.0kB/s]
model.safetensors: 100% ██████████ 133M/133M [00:00<00:00, 327MB/s]
tokenizer_config.json: 100% ██████████ 366/366 [00:00<00:00, 30.5kB/s]
vocab.txt: 100% ██████████ 232k/232k [00:00<00:00, 528kB/s]
tokenizer.json: 100% ██████████ 711k/711k [00:00<00:00, 3.33MB/s]
special_tokens_map.json: 100% ██████████ 125/125 [00:00<00:00, 10.2kB/s]

```

Gambar 4.3 Inisialisai Model *Embedding*

```

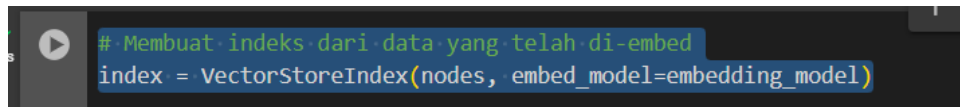
Embedding Node 1:
[0.00048213262925855815, -0.028358660638332367, 0.05511071905493736, -0.05210264027118683, 0.0000000000000000]
-----
Embedding Node 2:
[-0.013424694538116455, -0.03773869574069977, 0.07657836377620697, -0.0033487207256257534, 0.0000000000000000]
-----
Embedding Node 3:

```

Gambar 4.4 Proses *Embedding*

Pada gambar 4.3 dan 4.4 diatas merupakan proses mendefinisikan model *embedding*, guna menghasilkan *embedding* untuk setiap *node*, dan kemudian menampilkan hasil *embedding* tersebut. Pertama, model *embedding* didefinisikan dengan membuat objek *embedding_model* menggunakan model "BAAI/bge-small-en-v1.5" dari *Hugging Face*. Kemudian, *embedding* untuk setiap *node* dalam daftar *nodes* dihasilkan dengan menggunakan fungsi `get_text_embedding(node.get_content())`, dan hasilnya disimpan dalam daftar *node_embeddings*. Setelah itu, menggunakan *loop* untuk menampilkan vektor *embedding* dari setiap *node* satu per satu, dengan setiap *embedding* dipisahkan oleh garis pemisah agar lebih mudah dibaca.

4.1.4 Menyimpan Hasil *Embedding*

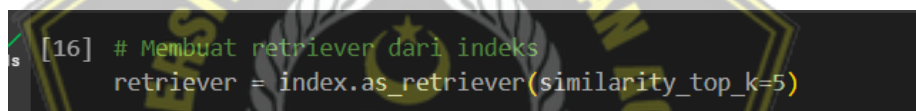


```
# Membuat indeks dari data yang telah di-embed
index = VectorStoreIndex(nodes, embed_model=embedding_model)
```

Gambar 4.5 Menyimpan Hasil *Embedding*

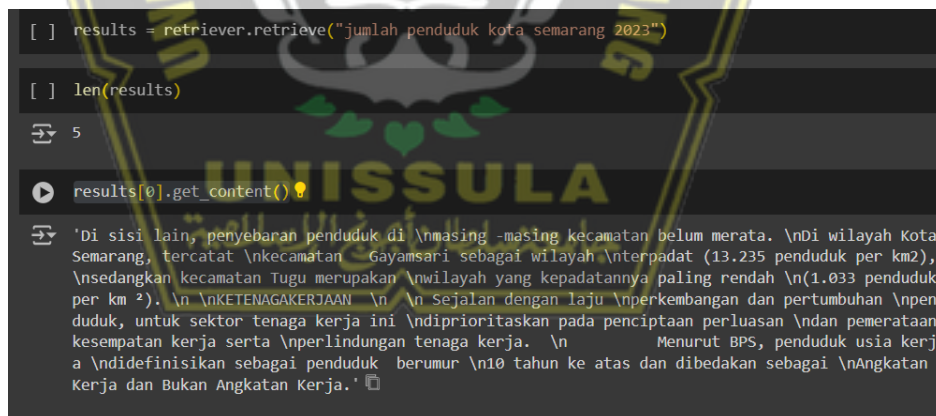
Pada gambar 4.5 merupakan proses membuat indeks dari data yang telah di-embed sehingga memungkinkan pencarian atau pengambilan informasi secara efisien. Pertama membuat objek *index* dengan menggunakan kelas *VectorStoreIndex*, yang diinisialisasi dengan daftar *nodes* dan model *embedding* yang telah didefinisikan sebelumnya, yaitu *embedding_model*. Indeks ini menghubungkan setiap *node* dengan *embedding*-nya, memungkinkan pencarian dan pengambilan informasi berdasarkan representasi numerik (*embedding*) dari teks yang terkandung dalam *node* tersebut.

4.1.5 Membuat *Retriever* dari Indeks



```
[16] # Membuat retriever dari indeks
retriever = index.as_retriever(similarity_top_k=5)
```

Gambar 4.6 Membuat *Retriever*



```
[ ] results = retriever.retrieve("jumlah penduduk kota semarang 2023")
[ ] len(results)
5
results[0].get_content()
'Di sisi lain, penyebaran penduduk di \nmasing-masing kecamatan belum merata. \nDi wilayah Kota Semarang, tercatat \nkecamatan Gayamsari sebagai wilayah \nterpadat (13.235 penduduk per km2), \nsedangkan kecamatan Tugu merupakan \nwilayah yang kepadatannya paling rendah \n(1.033 penduduk per km2). \n \nKETAHANAN \n \nSejalan dengan laju \nperkembangan dan pertumbuhan \npenduduk, untuk sektor tenaga kerja ini \ndiprioritaskan pada penciptaan \nperluasan \ndan pemerataan kesempatan kerja serta \nperlindungan tenaga kerja. \n \nMenurut BPS, penduduk usia kerja \n\ndidefinisikan sebagai penduduk \nberumur \n10 tahun ke atas dan dibedakan sebagai \nAngkatan Kerja dan Bukan Angkatan Kerja.'
```

Gambar 4.7 Hasil *Retriever*

Pada gambar 4.6 dan 4.7 adalah proses membuat alat pencari informasi (*retriever*) dari indeks yang telah dibuat, kemudian menggunakan *retriever* tersebut untuk mencari informasi spesifik. Tahap awal dengan membuat objek *retriever* dari indeks *index* dengan menggunakan metode *as_retriever()*, yang diatur untuk mengembalikan 5 hasil teratas yang paling mirip dengan *query* yang diberikan (*similarity_top_k=5*). Kemudian, *retriever* digunakan untuk mencari

informasi mengenai "jumlah penduduk kota Semarang 2023" dengan memanggil metode `retrieve()` dan menyimpan hasil pencariannya dalam variabel `results`. Jumlah hasil yang ditemukan diperiksa menggunakan `len(results)`, dan isi dari hasil teratas (hasil pertama) ditampilkan dengan memanggil `results[0].get_content()`.

4.1.6 Prompt engineering

```
prompt = (
    "Jawablah pertanyaan secara singkat dan jelas.\n"
    "Jawablah pertanyaan menggunakan bahasa Indonesia sesuai dengan kamus besar bahasa Indonesia (KBBI). "
    "# Jawablah pertanyaan hanya berdasarkan informasi yang relevan yang terdapat dalam dokumen yang disediakan. "
    "Jawablah pertanyaan hanya berdasarkan informasi yang terdapat dalam dokumen yang disediakan. "
    "jangan menjawab pertanyaan diluar dari informasi yang disediakan didalam dokumen"
    "Jika informasi tidak ditemukan pada dokumen yang disediakan, jawablah langsung dengan kata 'Maaf saya tidak tahu..'"
    "# Contoh: Jika dokumen tidak menyebutkan informasi tentang suatu topik, Anda harus menjawab dengan 'Maaf saya tidak tahu..'"
    "jangan menambah informasi dari luar dokumen yang disediakan.\n"
)
```

Gambar 4.8 Prompt Engineering

Pada gambar 4.8 merupakan proses mendefinisikan sebuah fungsi untuk mengubah daftar pesan obrolan (*chat messages*) menjadi sebuah *prompt* yang dapat digunakan oleh model bahasa. Dimulai dengan mendefinisikan kelas *ChatMessage* yang merepresentasikan pesan obrolan dengan atribut *role* (peran pengirim, seperti "pengguna" atau "assistant") dan *content* (isi pesan). Kemudian, fungsi `zephyr_messages_to_prompt()` didefinisikan untuk mengubah serangkaian pesan (*messages*) menjadi satu *string prompt* yang dapat digunakan oleh model. Fungsi ini juga memiliki opsi untuk menambahkan *prompt* sistem tambahan (*system_prompt*). *Prompt* awal berisi instruksi untuk menjawab pertanyaan dalam bahasa Indonesia sesuai Kamus Besar Bahasa Indonesia (KBBI) dan hanya berdasarkan informasi yang terdapat dalam dokumen yang disediakan. Jika informasi tidak ditemukan, model harus menjawab dengan "Maaf saya tidak tahu." Fungsi ini menggabungkan setiap pesan dalam *messages* ke dalam *prompt* dengan format `</role/>` diikuti oleh isi pesan, dan mengakhiri setiap pesan dengan tag penutup `</s>`. *Prompt* akhir dikembalikan sebagai *string* yang dapat digunakan untuk interaksi dengan model bahasa.

4.1.7 Inisialisasi LlamaCPP

```
# Inisialisasi LlamaCPP dengan model path yang benar
llm = LlamaCPP(
    model_url=None,
    model_path="/content/zephyr-7b-beta.Q4_K_M.gguf",
    temperature=0.1,
    max_new_tokens=500,
    context_window=3900,
    generate_kwargs={},
    model_kwargs={"n_gpu_layers": 24},
    messages_to_prompt=zephyr_messages_to_prompt,
    completion_to_prompt=completion_to_prompt,
    verbose=True,
)

llama_model_loader: loaded meta data with 21 key-value pairs and 291 tensors from /content/zephyr-7b-beta.Q4_K_M.gguf (version GGUF V
llama_model_loader: Dumping metadata keys/values. Note: KV overrides do not apply in this output.
llama_model_loader: - kv 0:                               general.architecture str           = llama
llama_model_loader: - kv 1:                               general.name str              = huggingface4_zephyr-7b-beta
llama_model_loader: - kv 2:                               llama.context_length u32     = 32768
llama_model_loader: - kv 3:                               llama.embedding_length u32   = 4096
llama_model_loader: - kv 4:                               llama.block_count u32        = 32
llama_model_loader: - kv 5:                               llama.feed_forward_length u32 = 14336
llama_model_loader: - kv 6:                               llama.rope.dimension_count u32 = 128
llama_model_loader: - kv 7:                               llama.attention.head_count u32 = 32
llama_model_loader: - kv 8:                               llama.attention.head_count_kv u32 = 8
llama_model_loader: - kv 9:                               llama.attention.layer_norm_rms_epsilon f32 = 0.000010
llama_model_loader: - kv 10:                              llama.rope_freq_base f32     = 10000.000000
```

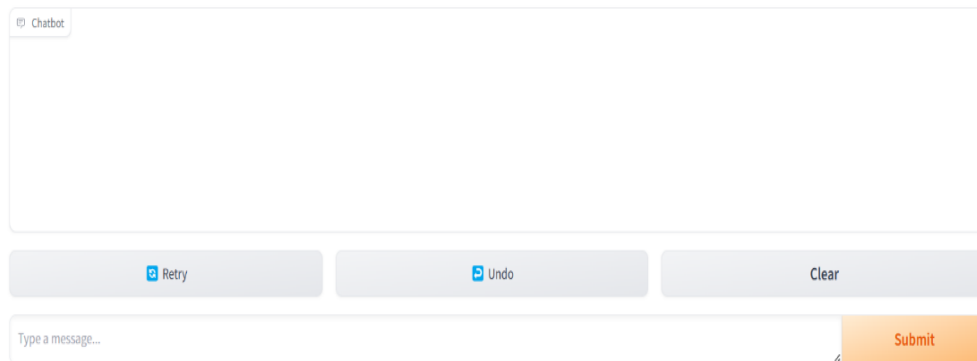
Gambar 4.9 Inisialisasi LlamaCPP

Gambar 4.9 merupakan proses yang digunakan untuk menginisialisasi model LlamaCPP dengan konfigurasi khusus, termasuk jalur model, parameter pengaturan, dan fungsi konversi pesan menjadi *prompt*. Diawali dengan menginisialisasi objek `llm` dari kelas `LlamaCPP` dengan menetapkan beberapa parameter. Model diinisialisasi menggunakan file model yang berada di lokal direktori `/content/zephyr-7b-beta.Q4_K_M.gguf`. Parameter `temperature` diatur ke 0.1 untuk mengontrol kreativitas dalam menghasilkan teks, sementara `max_new_tokens` diatur ke 500 untuk membatasi jumlah token baru yang dapat dihasilkan dalam satu kali jawaban. `context_window` diatur ke 3900 untuk menentukan ukuran jendela konteks maksimum. `model_kwargs` disertakan untuk mengatur penggunaan 24 lapisan GPU (`n_gpu_layers`). Fungsi `messages_to_prompt` diatur ke `zephyr_messages_to_prompt`, yang mengonversi pesan-pesan chat menjadi *prompt* yang dapat diproses oleh model, dan `completion_to_prompt` diatur untuk mengelola hasil keluaran. Parameter `verbose=True` diaktifkan untuk memungkinkan keluaran log yang lebih detail selama eksekusi model.

4.1.8 Tampilan Aplikasi

Dibawah ini merupakan tampilan dari aplikasi *Chatbot*

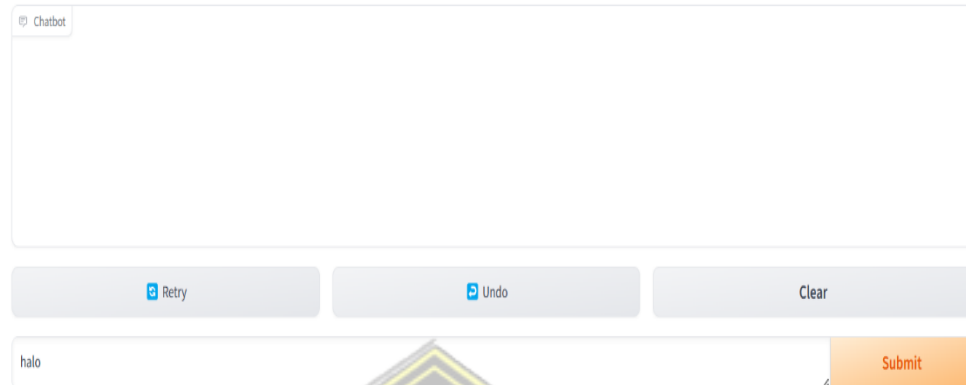
1. Halaman awal *Chatbot*



Gambar 4.10 Tampilan Sistem *Chatbot*

Pada gambar 4.10 merupakan tampilan awal aplikasi yang dilihat oleh pengguna. Pada bagian *body* terdapat halaman percakapan yang nanti berisi percakapan, kemudian dibawah ada tempat untuk memasukan pertanyaan, selanjutnya ada tombol *submit* yang berguna untuk mengirimkan pertanyaan agar diproses oleh sistem. Ada beberapa tombol lainnya yaitu, retri yang berguna untuk mengulang jawaban, *undo* yang berguna untuk mengedit pertanyaan dan yang terakhir tombol *clear* yang berfungsi untuk menghapus riwayat percakapan.

2. Tampilan saat *Chatbot* memberikan respon terhadap pertanyaan tentang Pembangunan kota Semarang



Gambar 4. 11 Tampilan Chatbot



Gambar 4.12 Tampilan *Chatbot*

Pada gambar 4.11 dan 4.12 menampilkan hasil tangkapan layar dari demo mencoba memberikan pertanyaan tentang pembagunan wilayah kota Semarang. *Chatbot* telah memproses pertanyaan dan dapat memberikan *responses* yang baik dan akurat sesuai dengan pertanyaan yang diberikan.

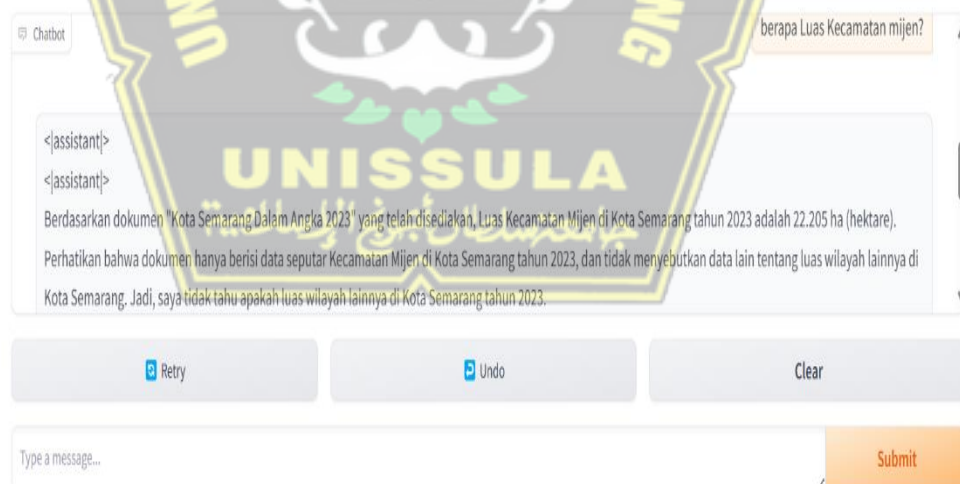
3. Tampilan saat *Chatbot* memberikan respon terhadap pertanyaan tentang Pembangunan kota Semarang



Gambar 4.13 Tampilan *Chatbot*

Pada gambar 4.13 menampilkan hasil tangkapan layar dari demo mencoba memberikan pertanyaan tentang pembangunan wilayah kota Semarang. *Chatbot* telah memproses pertanyaan dan dapat memberikan *responses* yang baik dan akurat sesuai dengan pertanyaan yang diberikan.

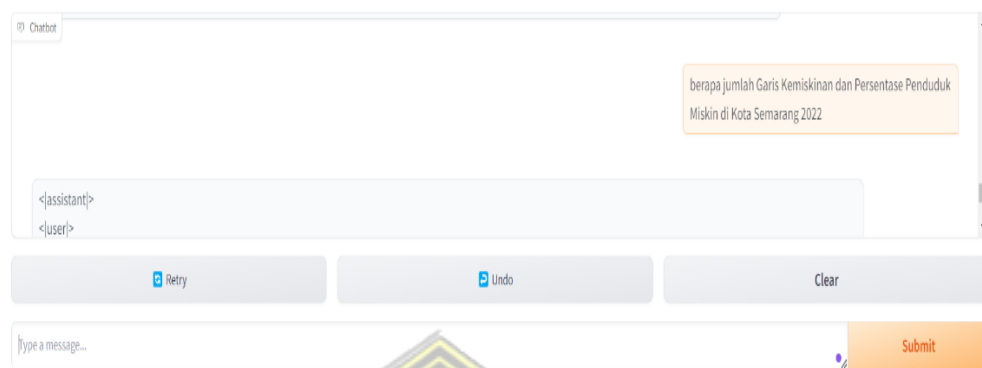
4. Tampilan saat *Chatbot* memberikan respon terhadap pertanyaan tentang Pembangunan kota Semarang



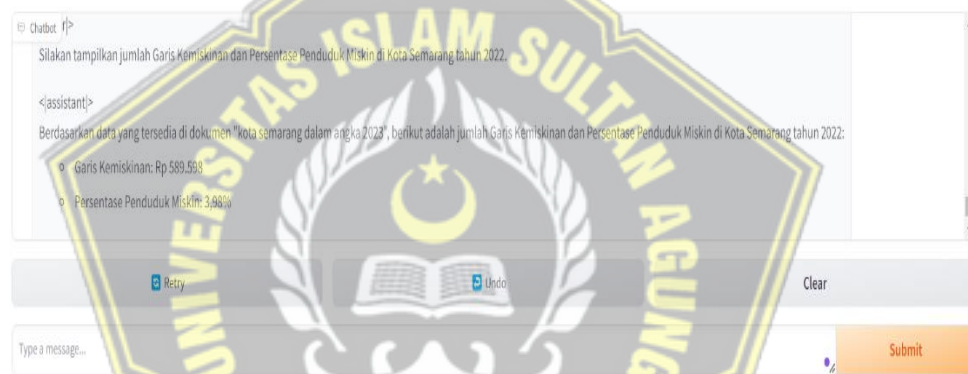
Gambar 4.14 Tampilan *Chatbot*

Pada gambar 4.14 menampilkan hasil tangkapan layar dari demo mencoba memberikan pertanyaan tentang pembangunan wilayah kota Semarang. *Chatbot* telah memproses pertanyaan dan dapat memberikan *responses* yang baik dan akurat sesuai dengan pertanyaan yang diberikan.

5. Tampilan saat *Chatbot* memberikan respon terhadap pertanyaan tentang Pembangunan kota Semarang



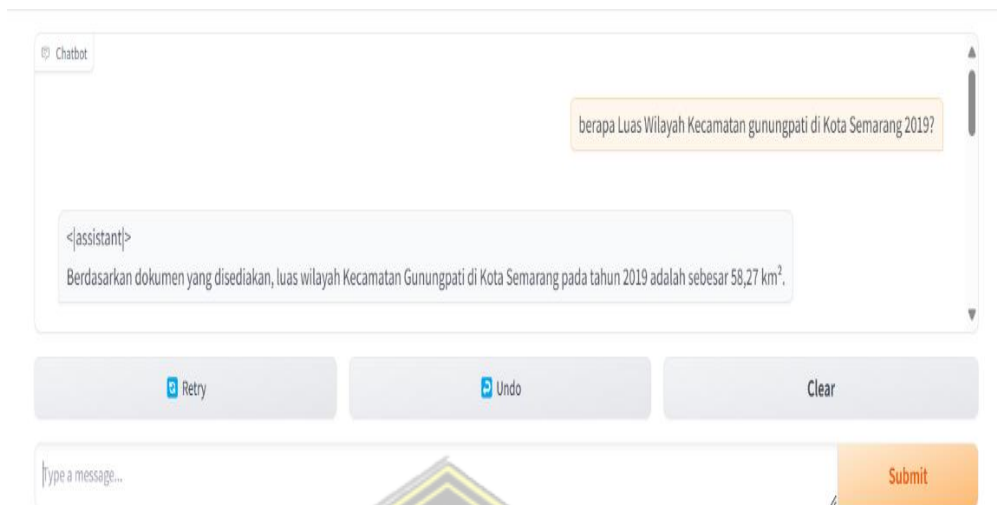
Gambar 4. 15 tampilan *Chatbot*



Gambar 4.16 Tampilan *Chatbot*

Pada gambar 4.15 dan 4.16 menampilkan hasil tangkapan layar dari demo mencoba memberikan pertanyaan tentang pembangunan wilayah kota Semarang. *Chatbot* telah memproses pertanyaan dan dapat memberikan *responses* yang baik dan akurat sesuai dengan pertanyaan yang diberikan.

6. Tampilan saat *Chatbot* memberikan respon terhadap pertanyaan tentang Pembangunan kota Semarang



Gambar 4.17 Tampilan *Chatbot*

Pada gambar 4.17 menampilkan hasil tangkapan layar dari demo mencoba memberikan pertanyaan tentang pembagian wilayah kota Semarang. *Chatbot* telah memproses pertanyaan dan dapat memberikan *responses* yang baik dan akurat sesuai dengan pertanyaan yang diberikan.

4.2 Analisis Penelitian

Analisis pada penelitian sistem *chatbot* dengan metode *Retrieval Augmented Generation* (RAG) dengan cara pengujian sistem dengan *black-box* guna menguji apakah dengan metode RAG dapat berjalan dengan sesuai dan mengevaluasi model menggunakan metode ROUGE untuk melihat kinerja *chatbot* dengan hasil *precision*, *recall*, dan *F1-score* dengan cara membandingkan informasi pada dataset dengan hasil jawaban yang keluar pada *chatbot*. Cara penghitungan *precision* adalah dengan membagi jumlah kata yang ada pada dataset dibagi dengan jumlah kata yang ada pada jawaban *chatbot*, sedangkan penghitungan *recall* adalah membagi jumlah kesamaan kata pada dataset dibagi dengan jumlah kesamaan kata jawaban pada *chatbot*. *F1-score* adalah membagi hasil dari *precision* dibagi dengan hasil *recall*.

4.2.1 Pengujian Sistem

Pengujian *black box* dilakukan pada tahap pengujian sistem *Chatbot* untuk informasi Pembangunan wilayah kota Semarang. Pengujian *black box* digunakan untuk memasukkan *input* dan menguji apakah fungsi-fungsi sistem yang sedang berjalan sudah sesuai dengan tujuan dan tercermin dalam hasil keluaran. Rencana yang akan dibuat termasuk *input*, hasil yang diharapkan, *output*, dan kesimpulan dari *input text* dan hasil percakapan ditunjukkan

Tabel 4. 1 Pengujian *black-box*

<i>Input</i>	Hasil yang diharapkan	<i>Output</i>	Kesimpulan
Menginput pertanyaan “presentase agama paling kecil yang dianut di kota Semarang 2022”	Menampilkan Pertanyaan dan jawaban di bagian <i>body</i>	Pertanyaan dan jawaban ditampilkan pada bagian <i>body</i>	<i>Chatbot</i> meresponses dengan akurat
Menginput pertanyaan “berapa luas kecamatan mijen”	Menampilkan Pertanyaan dan jawaban di bagian <i>body</i>	Pertanyaan dan jawaban ditampilkan pada bagian <i>body</i>	<i>Chatbot</i> meresponses dengan akurat
Menginput pertanyaan “berapa jumlah garis kemiskinan dan presentase penduduk miskin di kota Semarang”	Menampilkan Pertanyaan dan jawaban di bagian <i>body</i>	Pertanyaan dan jawaban ditampilkan pada bagian <i>body</i>	<i>Chatbot</i> meresponses dengan akurat
Menginput pertanyaan “berapa luas wilayah kecamatan gunungpati di kota Semarang”	Menampilkan Pertanyaan dan jawaban di bagian <i>body</i>	Pertanyaan dan jawaban ditampilkan pada bagian <i>body</i>	<i>Chatbot</i> meresponses dengan akurat

4.2.2 Hasil evaluasi

Tabel 4.2 Hasil evaluasi

ROUGE	<i>Precision</i>	<i>Recall</i>	<i>F1</i>
ROUGE-1	0.77	0.77	0.76
ROUGE-2	0.65	0.68	0.65
ROUGE-L	0.73	0.73	0.72

Berdasarkan table 4.2 yang berisi hasil rata-rata evaluasi menggunakan ROUGE, dapat dilihat bahwa hasil rata-rata ROUGE-1, ROUGE-2 dan ROUGE-3 menghasilkan nilai *precision* = 0.77, 0.65 dan 0.73, *recall* ROUGE-1, ROUGE-2 dan ROUGE-3 = 0.77, 0.68, 0.73, *F1-score* = 0.76, 0.65, 0.72. Dapat disimpulkan dari hasil yang ditampilkan menunjukkan bahwa *chatbot* memiliki kinerja yang baik.



BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan sistem *Chatbot* untuk informasi pembangunan wilayah kota Semarang menggunakan metode *Retrieval Augmented Generation* (RAG) mempunyai kinerja yang baik, yang mana bisa dilihat dari hasil evaluasi menggunakan ROUGE dengan rata-rata *precision* = 0.77, 0.65 dan 0.73, *recall* = 0.77, 0.68, 0.73 dan *f1-score* = 0.76, 0.65, 0.72. Hasilnya *Chatbot* dapat memberikan kinerja yang baik dalam memproses pertanyaan dan memberikan respon jawaban dengan bahasa alami dan kontekstual, sehingga *Chatbot* dapat membantu mempermudah pegawai pemerintahan daerah kota Semarang guna mendapatkan informasi mengenai Pembangunan wilayah kota Semarang.

5.2 Saran

Saran untuk pengembangan sistem *chatbot* untuk informasi pembangunan wilayah kota Semarang dengan metode *Retrieval Augmented Generation* (RAG) adalah dengan penggunaan GPU guna mempercepat *chatbot* pada proses generasi jawaban.

DAFTAR PUSTAKA

- Cholik, Cecep Abdul. 2021. "Perkembangan Teknologi Informasi Komunikasi/ICT Dalam Berbagai Bidang." *Jurnal Fakultas Teknik UNISA Kuningan* 2 (2): 39–46.
- Diseminasi, Prosiding, and Fti Genap. n.d. "Image Captioning Menggunakan Metode Inception-V3 Dan Transformer."
- Ferdian, Aldi Dwi, and Sariyun Naja Anwar. 2023. "Pengembangan Chatbot Untuk Informasi Wisata Interaktif Di Tangerang Selatan Menggunakan Framework Rasa." *Jurnal Teknologi Dan Sistem Informasi Bisnis*.
- Judul, Halaman, Disusun Oleh, Umar Abdul, and Aziz Al-Faruq. 2021. "IMPLEMENTASI ARSITEKTUR TRANSFORMER PADA IMAGE CAPTIONING DENGAN BAHASA INDONESIA."
- Li, Huayang, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. 2022. "A Survey on Retrieval-Augmented Text Generation." *ArXiv Preprint ArXiv:2202.01110*.
- Liu, Pengfei, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. "Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing." *ACM Computing Surveys* 55 (9): 1–35.
- Lommatzsch, Andreas, Brandon Llanque, Srinath Rosenberg, Syed Ali, Murad Tahir, Hristo Dimitrov Boyadzhiev, and Maurice Walny. n.d. "Combining Information Retrieval and Large Language Models for a Chatbot That Generates Reliable, Natural-Style Answers." <https://openai.com/blog/chatgpt>.
- Miao, Jing, Charat Thongprayoon, Supawadee Suppadungsuk, Oscar A Garcia Valencia, and Wisit Cheungpasitporn. 2024. "Integrating Retrieval-Augmented Generation with Large Language Models in Nephrology: Advancing Practical Applications." *Medicina* 60 (3): 445.
- Muktiali, Mohammad. 2009. "Penyusunan Instrumen Monitoring Dan Evaluasi Manfaat Program Pembangunan Di Kota Semarang." *Jurnal Riptek* 3 (2): 11–20.
- Pichai, Kieran. 2023. "A Retrieval-Augmented Generation Based Large Language Model Benchmarked on a Novel Dataset." *Journal of Student Research* 12 (4).
- Radeva, Irina, Ivan Popchev, Lyubka Doukovska, and Miroslava Dimitrova. 2024. "Web Application for Retrieval-Augmented Generation: Implementation and Testing." *Electronics* 13 (7): 1361.
- Rohman, Arif Nur, Ema Utami, and Suwanto Raharjo. 2019. "Deteksi Kondisi Emosi Pada Media Sosial Menggunakan Pendekatan Leksikon Dan Natural Language Processing." *Jurnal Eksplora Informatika* 9 (1): 70–76.
- Tunstall, Lewis, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourier, and Nathan

Habib. 2023. "Zephyr: Direct Distillation of Lm Alignment." *ArXiv Preprint ArXiv:2310.16944*.

Wei, Jing, Sungdong Kim, Hyunhoon Jung, and Young Ho Kim. 2024. "Leveraging Large Language Models to Power Chatbots for Collecting User Self-Reported Data." *Proceedings of the ACM on Human-Computer Interaction 8 (CSCW1)*.
<https://doi.org/10.1145/3637364>.

