RANCANG BANGUN *PIPELINE* NOTULENSI OTOMATIS DENGAN INTEGRASI MODEL WHISPER *(SPEECH-TO-TEXT)* DAN MODEL PEGASUS *(ABSTRACTIVE SUMMARIZATION)*

Oleh

MUHAMMAD SYARIF AL HUSEIN

Skripsi

Sebagai Salah Satu Syarat untuk Mencapai Gelar
SARJANA TEKNIK

Pada

Program Studi Teknik Informatika

Fakultas Teknik Universitas Lampung

FAKULTAS TEKNIK

UNIVERSITAS LAMPUNG

BANDAR LAMPUNG

2025

RANCANG BANGUN *PIPELINE* NOTULENSI OTOMATIS DENGAN INTEGRASI MODEL WHISPER *(SPEECH-TO-TEXT)* DAN MODEL PEGASUS *(ABSTRACTIVE SUMMARIZATION)*

Oleh

MUHAMMAD SYARIF AL HUSEIN

Skripsi

Sebagai Salah Satu Syarat untuk Mencapai Gelar
SARJANA TEKNIK

Pada

Program Studi Teknik Informatika

Fakultas Teknik Universitas Lampung

FAKULTAS TEKNIK

UNIVERSITAS LAMPUNG

BANDAR LAMPUNG

2025

ABSTRAK

RANCANG BANGUN *PIPELINE* NOTULENSI OTOMATIS DENGAN INTEGRASI MODEL WHISPER (*SPEECH-TO-TEXT*) DAN MODEL PEGASUS (A*BSTRACTIVE SUMMARIZATION*)

Oleh:

Muhammad Syarif Al Husein

Seiring meningkatnya pertemuan daring, kebutuhan akan sistem pencatatan otomatis yang efisien menjadi krusial untuk mengatasi keterbatasan notulensi manual, seperti risiko kehilangan informasi penting dan ketergantungan pada notulis. Penelitian ini bertujuan untuk merancang, membangun, dan mengevaluasi sebuah *pipeline* notulensi otomatis yang mengintegrasikan model Whisper untuk transkripsi *speech-to-text* dan model Pegasus untuk *abstractive summarization*, dengan penyermpurnaan hasil akhir oleh model GPT. Metode penelitian ini meliputi perancangan arsitektur *pipeline* sekuensial yang diawalli dengan *preproceesing audio*, transkripsi, segmentasi teks, peringkasan, hingga parafrase. Kinerja sistem dievaluasi secara kuantitatif menggunakan metrik ROUGE dan secara kualitatif melalui kuesioner *Likert Scale* kepada 51 responden, dengan menggunakan *dataset* berisi 100 rekaman *audio* berbahasa Indonesia dari kategori rapat, perkuliahan, seminar, dan *podcast*. Hasil pengujian kuantitatif menunjukkan kinerja yang solid dengan skor F1-*score* rata-rata ROUGE-1 sebesar 0,5103, ROUGE-2 sebesar 0,4575, dan ROUGE-L sebesar 0,4743. Secara kualitatif, evaluasi pengguna menghasilkan skor kepuasan rata-rata 4,37 dari 5, yang mengindikasikan penerimaan yang sangat baik terhadap akurasi, keterbacaan, dan struktur notulensi. Disimpulkan bahwa *pipeline* yang dirancang berhasil mengotomatiskan proses notulensi secara efektif, meningkatkan efisiensi, serta menghasilkan dokumen yang akurat dan terstruktur yang siap untuk implementasi praktis.

Kata kunci: Notulensi Otomatis, *Speech-to-Text*, *Abstractive Summarization*, Model Whisper, Model Pegasus.

ABSTRACT

DESIGN AND IMPLEMENTATION AUTOMATED MINUTES PIPELINE BY INTEGRATING THE WHISPER (SPEECH-TO-TEXT) AND PEGASUS (ABSTRACTIVE SUMMARIZATION) MODELS

By:

Muhammad Syarif Al Husein

With the increasing number of online meetings, the need for an efficient automated recording system has become crucial to overcome the limitations of manual note-taking, such as the risk of losing important information and reliance on a human notetaker. This research aims to design, build, and evaluate an automated minutes pipeline that integrates the Whisper model for speech-to-text transcription and the Pegasus model for abstractive summarization, with final refinement by the GPT model. The research method includes designing a sequential pipeline architecture starting from audio preprocessing, transcription, text segmentation, summarization, to paraphrasing. The system's performance was evaluated quantitatively using ROUGE metrics and qualitatively through a Likert scale questionnaire distributed to 51 respondents, using a dataset of 100 Indonesian audio recordings from the categories of meetings, lectures, seminars, and podcasts. The quantitative test results show a solid performance with average F1-scores of 0.5103 for ROUGE-1, 0.4575 for ROUGE-2, and 0.4743 for ROUGE-L. Qualitatively, the user evaluation yielded an average satisfaction score of 4.37 out of 5, indicating excellent acceptance of the accuracy, readability, and structure of the minutes. It is concluded that the designed pipeline successfully automates the minute-taking process effectively, enhances efficiency, and produces accurate and structured documents ready for practical implementation.

Keywords: Automated Minutes, Speech-to-Text, Abstractive Summarization, Whisper Model, Pegasus Model.

Judul Skripsi	**: RANCANG BANGUN *PIPELINE* NOTULENSI OTOMATIS DENGAN INTEGRASI MODEL WHISPER (*SPEECH-TO-TEXT*) DAN MODEL PEGASUS (*ABSTRACTIVE SUMMARIZATION*)**
Nama Mahasiswa	**: Muhammad Syarif Al Husein**
Nomor Pokok Mahasiswa	**: 2055061013**
Program Studi	**: Teknik Informatika**
Fakultas	**: Teknik**

MENYETUJUI

1. Komisi Pembimbing

Yessi Mulyani S.T., M.T.	**Deny Budiyanto, S.kom., M.T.**
NIP. 197312262000122001	NIP. 199112082019031011

1. Mengetahui

Ketua Jurusan	**Ketua Program Studi**
Teknik Elektro	**Teknik Informatika**
Herlinawati, S.T., M.T.	**Yessi Mulyani S.T., M.T.**
NIP. 197103141999032001	NIP. 197312262000122001

MENGESAHKAN

1. Tim Penguji

Ketua : Yessi Mulyani S.T., M.T. .....

Sekretaris : Deny Budiyanto, S.kom., M.T. .....

Penguji : Ir.M. Komarudin, S.T., M.T. .....

2. Dekan Fakultas Teknik

Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc.

NIP. 197509282001121002

Tanggal Lulus Ujian Skripsi : 01 September 2025

SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa dalam skripsi ini tidak terdapat karya yang pernah dilakukan orang lain dan sepanjang sepengetahuan saya tidak terdapat atas diterbitkannya oleh orang lain, kecuali secara tertulis diacu dalam naskah ini sebagaimana yang disebutkan dalam Daftar Pustaka. Selain itu, saya menyatakan pula bahwa skripsi ini dibuat oleh saya sendiri.

Apabila pernyataan saya tidak benar, maka saya bersedia dikenai sanksi akademik sesuai dengan hukum yang berlaku.

Bandar Lampung, 15 Oktober 2025

Muhammad Syarif Al Husein
NPM. 2055061013

RIWAYAT HIDUP

Sayar lahir di Liwa, Lampung Barat, pada 2 September 2001. Saya adalah anak sulung dari tiga bersaudara dari pasangan Bapak Gempar Ependi dan Ibu Minarti. Fondasi pendidikan formal saya terbangun secara bertahap di kota kelahiran, mulai dari SDN 1 Liwa, berlanjut ke SMPN 1 Liwa, hingga lulus dari SMAN 1 Liwa. Perjalanan ini mencapai puncaknya pada tahun 2020 ketika saya diterima di Program Studi Teknik Informatika, Universitas Lampung. Di lingkungan kampus inilah fase pengembangan diri saya dimulai secara intensif. Awalnya saya aktif di Himpunan Mahasiswa Teknik Elektro (Himatro), namun sebuah titik balik penting terjadi ketika saya dianugerahi beasiswa Karya Salemba Empat (KSE). Kepercayaan yang diberikan kepada saya di dalam Paguyuban KSE UNILA terus bertumbuh secara signifikan dari seorang anggota, saya diamanahi menjadi Kepala Bidang *Community Development*, hingga puncaknya mengemban tanggung jawab sebagai Ketua Umum Paguyuban KSE UNILA.

Pengalaman memimpin tersebut mendorong saya untuk proaktif menerjemahkan teori ke dalam praktik nyata. Melalui program Merdeka Belajar Kampus Merdeka (MBKM), saya memperluas wawasan dengan mengikuti Studi Independen di Dicoding Indonesia untuk mengasah keahlian digital dan merasakan langsung dunia industri melalui magang di Widya Robotic. Kesempatan ini diperkaya dengan keterlibatan saya dalam penelitian "*Technology for Indonesia*" serta program pelatihan kepemimpinan intensif dari Yayasan KSE. Seluruh rangkaian perjalanan ini, dari ruang kelas hingga keterlibatan di masyarakat dan industri, telah menempa saya menjadi pribadi yang berprinsip untuk terus belajar, bertumbuh, dan berkontribusi secara nyata.

MOTTO

“Apa yang melewatkanku tidak akan pernah menjadi takdirku, dan apa yang ditakdirkan untukku tidak akan pernah melewatkanku.”

- Umar bin Khattab -

“Lidah orang yang berakal berada di belakang hatinya, sedangkan hati orang bodoh berada di belakang lidahnya.”

- Ali bin Abi Thalib -

“Patah, Pulih, Mengudara.”

- Muhammad Syarif Al Husein -

PERSEMBAHAN

Bismillahirrahmanirrahim

Dengan Ridho Allah SWT

Teriring shalawat kepada Nabi Muhammad SAW

Karya Tulis ini aku persembahkan untuk:

Ayah dan Bunda Tercinta

Bapak Gempar Ependi dan Ibu Minarti

Serta Adik-Adikku

Fitria Mayla Rosyada dan (Alm.) Abdul Karim Gumilang

Terima kasih untuk semua doa dan dukungan selama ini.

Sehingga aku dapat menyelesaikan hasil karyaku ini.

SANWACANA

Alhamdulillah, puji Syukur saya panjatkan ke hadirat Allah SWT atas segala rahmat, hidayah, serta inayah-Nya. Dengan segala nikmat dan karunia-Nya, laporan skripsi yang berjudul "Rancang bangun *pipeline* notulensi otomatis dengan integrasi model Whisper (*Speech-to-Text*) dan model Pegasus (A*bstractive Summarization*)" ini dapat diselesaikan. Laporan ini disusun sebagai salah satu syarat untuk memperolah gelar Sarjana Teknik di Program Studi Teknik Informatika, Jurusan Teknik Elektro, Fakultas Tenik, Univeristas Lampung.

Shalawat beriring salam, semoga selalu tercurah kepada Nabi Muhammad SAW, keluarga, sahabat, serta para pengikutnya yang teguh dan istiqomah di jalan Islam hingga akhir zaman.

Selama proses penyusunan skripsi ini, banyak sekali pihak yang telah memberikan bantuan, baik berupa pemikiran maupun dorongan moril. Oleh karena itu, dengan penuh rasa hormat, saya ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Bapak Gempar Ependi dan Ibu minarti selaku orang tua yang senantiasa memberikan dukungan dan do'a kepada penulis.
2. Fitria Mayla Rosyada dan (Alm.) Abdul Karim Gumilang selaku adik-adik yang senantiasa mendukung dan bangga kepada penulis.
3. Ibu Prof. Dr.Ir. Lusmeilia Afriani, D.E.A., I.P.M., selaku Rektor Universitas Lampung.
4. Bapak Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc., selaku Dekan Fakultas Teknik Universitas Lampung.
5. Ibu Yessi Mulyani S.T., M.T. selaku Ketua Program Studi Teknik Informatika Fakultas Teknik Universitas Lampung dan selaku Dosen Pembimbing Utama bagi penulis.

6. Bapak Deny Budiyanto, S.Kom., M.T. selaku Pembimbing Pendamping tugas akhir, yang telah membantu, membimbing, dan memberi dukungan kepada penulis.
7. Bapak Ir. M. Komarudin, S.T., M.T. selaku Dosen Penguji bagi penulis.
8. Bapak Mona Arif Muda, S.T., M.T. sebagai Dosen Pembimbing Akademik, yang telah membantu dalam perjalanan akademik penulis.
9. Seluruh Dosen dan Karyawan Jurusan Teknik Elektro Universitas Lampung, berkat bantuan dan ilmu yang telah diajarkan kepada penulis selama penulis menjalani masa studi di perkuliahan.
10. Yayasan Karya Salemba Empat, yang telah memberikan beasiswa, kepercayaan, dan dukungan kepada penulis.
11. Keluarga Paguyuban Karya Salemba Empat Universitas Lampung periode 2021, 2022, dan 2023 yang telah memberikan pengalaman berharga dan warna dalam kanvas perjalanan hidup penulis.
12. Teman-teman perjuangan (Ivan, Faniel, dan Farhan) yang telah mendukung dan membantu penulis selama masa perkuliahan.
13. Putri Nabilah yang telah mendukung dan menemani penulis selama proses pengerjaan skripsi

Semoga Allah SWT membalas semua perbuatan dan kebaikan yang telah diberikan kepada Penulis sampai dengan terselesaikannya Skripsi ini. Penulis menyadari bahwa laporan skripsi ini masih memiliki banyak kekurangan, baik dari segi penyusunan maupun pemilihan kata. Maka dari itu penulis terbuka untuk menerima masukan, kritik, dan saran yang dapat membangun Penulis kedepannya. Semoga penulisan skripsi ini dapat bermanfaat bagi pembaca.

Bandar Lampung, 15 Oktober 2025

Penulis,

Muhammad Syarif Al Husein

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Gambar Halaman

DAFTAR TABEL

Tabel Halaman

I. PENDAHULUAN

1.1 Latar Belakang

Di era digital yang selalu berkembang, teknologi telah memainkan peran yang sangat signifikan dalam memfasilitasi berbagai aspek kehidupan manusia, termasuk cara berkomunikasi dan bekerja. *Online platform* memungkinkan manusia untuk dapat mengakses kegiatan rapat dan perkuliahan dari mana saja, hal tersebut dapat mengurangi waktu dan biaya perjalanan, dan membuat kegiatan lebih inklusif untuk orang-orang yang tinggal jauh dari lokasi fisik. Penggunaan metode ini juga memungkinkan peserta untuk mengikuti kegiatan secara *synchronous* dan *asynchronous* yang memungkinkan peserta untuk menyesuaikan dengan waktu mereka sendiri. Metode ini meningkatkan keterlibatan peserta karena fleksibilitas waktu dan lokasi yang ditawarkan oleh *online platform* [1].

Diskusi dan interaksi yang difasilitasi oleh *online platform* dapat memperdalam pemahaman peserta. Kelas daring yang dilakukan dengan interaktif dapat mendukung pemahaman yang mendalam melalui komunikasi dua arah yang membantu mengurangi pendekatan hafalan dan mengoptimalkan diskusi serta umpan balik secara *real-time* [2]. Sejalan dengan meningkatnya penggunaan teknologi digital di berbagai sektor, sistem daring dan hybrid semakin banyak digunakan dalam kegiatan rapat, perkuliahan dan seminar. Peningkatan jumlah pertemuan daring ini menuntut adanya sistem pencatatan otomatis yang efisien, mengingat keterbatasan sumber daya manusia dan risiko kehilangan informasi penting selama diskusi berlangsung.

Rapat merupakan kegiatan yang sering dilakukan dalam suatu organisasi. Pada kegiatan ini setiap peserta akan berkumpul, bertukar informasi, serta berdiskusi

untuk menyelesaikan suatu masalah tertentu. Di dalam sebuah rapat harus menghasilkan suatu keputusan yang nantinya akan dilaksanakan sesuai dengan kesepakatan yang telah dibuat. Kegiatan rapat dilakukan dengan berbagai alat bantu, salah satunya adalah notulen rapat. Notulen rapat adalah suatu dokumentasi yang dibuat dengan tujuan untuk mencatat pembahasan penting selama rapat tersebut berlangsung [3]. Orang yang bertanggung jawab untuk mencatat laporan disebut notulis, dan peran ini dapat diemban oleh seorang sekretaris atau seseorang yang ditunjuk sebagai asisten ketua dalam rapat. Namun, ketergantungan akan seseorang untuk melaksanakan tugas notulis dapat menyebabkan masalah bila orang tersebut tidak hadir dalam rapat, dan masalah lainnya adalah terkadang notulis kesulitan mengikuti ritme pembahasan dalam rapat, yang dapat mengakibatkan beberapa poin pembahasan terlewatkan [4]. Berdasarkan permasalahan tersebut, penulis tertarik untuk melakukan penelitian dengan judul "Rancang bangun *pipeline* notulensi otomatis dengan integrasi model Whisper (*Speech-to-Text*) dan model Pegasus (A*bstractive Summarization*)". Penelitian ini bertujuan untuk mengembangkan rancangan *pipeline* notulensi otomatis yang menggabungkan model-model khusus untuk setiap tahap notulensi. Mulai dari tahap transkripsi, peringkasan, dan penyempurnaan sebagai solusi yang lebih efisien dan efektif. Dengan mengintegrasikan model Whisper untuk transkripsi, model Pegasus untuk *abstractive summarization*, dan model GPT untuk penyempurnaan hasil akhir notulensi. Sistem ini dirancang untuk mengoptimalkan kinerja setiap tugas secara spesifik. Selain itu, keseluruhan *pipeline* diimplementasikan pada lingkungan Google Colab untuk menekan biaya infrastruktur, dan membuktikan bahwa sistem canggih dapat dibangun dengan sumber daya yang terjangkau. Diharapkan hasil dari penelitian ini dapat membantu menghasilkan dokumentasi yang lebih sistematis, mudah diakses dan mendukung pengambilan keputusan yang lebih baik.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, maka rumusan masalah dari penelitian ini adalah sebagai berikut:

1. Bagaimana membangun sebuah arsitektur *pipeline* yang mampu mengintegrasikan model Whisper dan model Pegasus untuk menghasilkan notulensi yang terstruktur dan akurat?
2. Bagaimana menentukan parameter model Whisper yang optimal untuk kualitas notulensi dalam aristektur *pipeline?*
3. Bagaimana menentukan strategi segmentasi teks dan parameter model Pegasus yang optimal untuk kualitas notulensi dalam arsitektur *pipeline?*
4. Bagaimana evaluasi arsitektur *pipeline* notulensi otomatis yang mengintegrasikan model Whisper dan model Pegasus sebagai satu kesatuan yang meningkatkan efisiensi dan menghasilkan notulensi yang berkualitas?

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Penelitian ini akan berfokus pada perancangan, implementasi, dan evaluasi *pipeline* terintegrasi, bukan pada modifikasi atau analisis terhadap model Whisper dan model Pegasus.
2. Penelitian ini akan terbatas pada data *audio* yang sudah direkam sebelumnya (*pre-recorded*) dan tidak dirancang untuk melakukan transkripsi langsung (*live transcription*) dari sumber suara secara *real time*.
3. Evaluasi sistem berfokus pada kualitas notulensi yang dihasilkan oleh keseluruhan *pipeline*, dengan menggunakan metrik ROUGE untuk mengukur tumpang tindih informasi dan metode *Likert Scale* untuk mengukur persepsi pengguna.

1.4 Tujuan Penelitian

1. Membangun sebuah arsitektur *pipeline* notulensi otomatis yang mengintegrasikan model Whisper untuk transkripsi dan model Pegasus untuk ringkasan dari rekaman *audio* rapat, perkuliahan, dan seminar.
2. Menentukan parameter model Whisper yang optimal untuk kualitas notulensi dalam aristektur *pipeline*.
3. Menentukan strategi segmentasi teks dan parameter model Pegasus yang paling optimal untuk meningkatkan kualitas hasil notulensi dalam alur kerja *pipeline*.
4. Mengevaluasi performa dan kualitas hasil notulensi dari *pipeline* yang dikembangankan untuk mengukur peningkatan efisiensi, akurasi, dan keterbacaan notulensi otomatis.

1.5 Manfaat Penelitian

Manfaat penelitian ini adalah sebagai berikut:

1. Meningkatkan efisiensi kegiatan rapat, perkuliahan, dan seminar dengan ketergantungan pada notulis manusia. Ini akan membantu peserta untuk tetap fokus pada pembahasan inti tanpa harus khawatir tentang pencatatan manual.
2. Menghasilkan dokumen notulensi yang berkualitas, terstruktur, dan mudah dipahami. Rancangan *pipeline* yang mengintegrasikan transkripsi dan peringkasan cerdas memastikan bahwa informasi penting disajikan secara koheren dan ringkas, sehingga dapat mendukung proses penyebaran informasi dan pengambilan keputusan yang lebih cepat.
3. Menjadi referensi teknis dan studi kasus bagi peneliti atau pengembang selanjutnya mengenai bagaimana merancang dan membangun arsitektur *pipeline* yang menggabungkan beberapa model *pre-trained* untuk menyelesaikan sebuah masalah yang kompleks.

1.6 Sistematika Penulisan

Sistematika penulisan yang digunakan pada skripsi ini adalah pembagian menjadi 5 bab sebagai berikut:

BAB I : PENDAHULUAN

Bab ini menguraikan secara umum mengenai latar belakang penelitian, rumusan masalah, batasan penelitian, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB II : TINJAUAN PUSTAKA

Pada bab ini membahas mengenai *Pipeline, Artificial Intelligence,* Cara Kerja *Artificial Intelligence, Machine learning*, *Deep Learning*, *Natural Language Processing*, *Speech-To-Text*, *Abstractive Summarization*, Model Whisper, Model Pegasus, Model GPT, *Audio*, ROUGE, *Likert scale,* Streamlit, Ngrok dan penelitian terkait yang berfungsi dalam memahami permasalahan terkait rancang bangun *pipeline* notulensi otomatis dengan integrasi model Whisper (*speech-to-text*) dan model Pegasus (*abstractive summarization*)".

BAB III : METODE PENELITIAN

Pada bab ini membahas mengenai Studi Literatur, Pemilihan Komponen dan Teknologi *Pipeline*, Pengumpulan Data, *Preprocessing* Data, Desain Arsitektur *Pipeline*, Implementasi *Pipeline*, Pengujian, Penyusunan Laporan yang digunakan dalam rancang bangun *pipeline* notulensi otomatis dengan integrasi model Whisper (*speech-to-text*) dan model Pegasus (*abstractive summarization*)".

BAB IV : HASIL DAN PEMBAHASAN

Pada bab ini berisi tentang pembahasan serta hasil yang diperoleh dalam penelitian.

BAB V : KESIMPULAN DAN SARAN

Pada bab ini berisi kesimpulan dari hasil penelitian dan saran-saran sebagai masukan untuk penelitian lanjutan di masa mendatang.

DAFTAR PUSTAKA

LAMPIRAN

II. TINJAUAN PUSTAKA

2.1 *Pipeline*

Secara fundamental, *pipeline* dalam komputasi adalah serangkaian elemen pemrosesan data yang terhubung secara sekuensial, dimana *output* dari satu elemen menjadi masukan untuk elemen berikutnya. Konsep itu berpusat pada penciptaan alur kerja yang otomatis dan modular untuk tugas-tugas komputasi yang kompleks.

Dalam konteks *machine learning* (ML), sebuah *pipeline* mengotomatiskan seluruh alur kerja, mulai dari pengumpulan dan *preprocessing* data, hingga pelatihan model, evaluasi, dan penerapan (*deployment*).

Arsitektur *pipeline* ML dapat dikategorikan ke dalam beberapa paradigma. Penelitian ini secara spesifik mengadopsi dan mengimplementasikan arsitektur modular atau *cascade*. Arsitektur ini membagi alur kerja yang kompleks menjadi komponen-komponen yang independen, dapat digunakan kembali, dan terspesialisasi. Keluaran dari satu model khusus (misalnya, transkripsi dari Whisper) "dialirkan" sebagai masukan ke model berikutnya (misalnya, Pegasus untuk peringkasan). Pendekatan ini menawarkan beberapa keuntungan:

- **Spesialisasi:** Setiap modul dapat menjadi model *state-of-the-art* untuk tugas spesifiknya (misalnya, Whisper untuk transkripsi dan Pegasus untuk ringkasan), memungkinkan sistem untuk memanfaatkan teknologi terbaik di setiap bidang [5].
- **Fleksibilitas:** Komponen dapat ditukar atau diperbarui secara independent tanpa perlu merancang ulang seluruh sistem. Sebagai contoh, model Whisper dapat diganti dengan model *speech-to-text* lain di masa depan jika ada yang model yang lebih baik [5].

- **Kemudahan Pemeliharaan:** Mengisolasi tugas ke dalam modul-modul yang terpisah menyederhanakan proses pencarian dan perbaikan serta pemeliharaan sistem secara keseluruhan [5].

Sebagai kontras, model *end-to-end* (E2E) bertujuan untuk mempelajari pemetaan langsung dari masukan awal ke keluaran akhir dalam satu jaringan saraf tiruan yang terpadu. Pendekatan ini merupakan area penelitian akademis yang signifikan. Dalam implementasi praktis, arsitektur *cascade* terbukti secara konsisten mengungguli model E2E dalam berbagai metrik, termasuk konsistensi dan relevansi. Keunggulan ini disebabkan oleh kemampuan arsitektur *cascade* untuk memanfaatkan kekuatan model peringkasan (seperti Pegasus) yang telah dilatih secara ekstensif pada korpus data yang sangat besar, yang seringkali tidak tersedia untuk melatih model E2E secara efektif. Dengan demikian, pilihan untuk membangun *pipeline cascade* kustom adalah keputusan yang didasarkan pada bukti kinerja empiris [6].

Kerangka kerja ML standar seringkali menyediakan objek *pipeline* generik, seperti kelas *pipeline* dari Hugging Face, yang dirancang untuk menyederhanakan penggunaan satu model [7]. Namun, kerangka kerja ini seringkali tidak memadai ketika mengintegrasikan beberapa model berskala besar dan boros sumber daya seperti versi *large* dari model Whisper dan model Pegasus. Oleh karena itu, *pipeline* generik yang abstrak menjadi tidak layak. Hal ini mengharuskan pembuatan *pipeline* kustom, sebuah skrip yang dirancang khusus untuk secara eksplisit mengelola aliran data, proses pemuatan dan pelepasan model (*loading/unloading*), serta alokasi memori di antara setiap langkah diskrit. Pilihan untuk membangun pipeline kustom bukanlah sekadar preferensi, melainkan konsekuensi logis dari keputusan untuk menggunakan model *state-of-the-art* berskala besar. Hal ini menyoroti sebuah tantangan dalam pengembangan AI modern: model-model yang paling kuat telah melampaui kapasitas kerangka kerja serbaguna yang dirancang untuk pendahulu mereka yang lebih kecil. Akibatnya, peneliti harus mengambil kendali manual atas alur kerja, memuat satu model, menjalankan inferensi, melepaskannya dari memori, membersihkan *cache*, lalu memuat model berikutnya. Dengan demikian, "*pipeline* kustom" ini merupakan solusi rekayasa yang

diperlukan oleh tuntutan komputasi dari model yang dipilih, yang mengangkat nilai penelitian ini dari sekadar penerapan model menjadi sebuah karya rekayasa sistem.

2.2 *Artificial Intelligence* (AI)

Artificial intelligence (AI) adalah bidang ilmu komputer yang bertujuan untuk menciptakan sistem atau mesin yang mampu melakukan tugas-tugas untuk memecahkan masalah kognitif yang umumnya terkait dengan kecerdasan manusia, seperti pembelajaran, penciptaan, dan pengenalan gambar. Kecerdasan ini melibatkan kemampuan untuk belajar, memahami, menganalisis, dan mengambil keputusan, serta dapat berinteraksi dengan lingkungan sekitar secara dinamis. Di dalam dunia yang semakin kompleks dan *data-driven*, AI menjadi komponen penting dalam berbagai aspek, mulai dari teknologi-teknologi sehari-hari hingga skala industri canggih.

Untuk memahami AI secara menyeluruh, kita perlu mengenal beberapa cabang utama dari AI, yaitu *learning*, *searching*, *reasoning*, dan *perception*. Masing-masing cabang ini memiliki peran dalam menciptakan sistem AI yang fungsional dan efektif.

1. *Learning* (Pembelajaran)

   Learning atau pembelajaran adalah salah satu cabang paling penting di dalam AI. Di sinilah sistem AI mendapatkan kemampuan untuk belajar dari data atau pengalaman, sehingga dapat meningkatkan kinerja dari AI itu sendiri seiring waktu. Proses pembelajaran ini melibatkan pengolahan data untuk menemukan pola atau hubungan yang berguna. *Machine learning* telah menjadi komponen penting dalam teknologi AI modern seperti pengenalan suara dan sistem rekomendasi [8].
2. *Searching* (Pencarian)

   Searching atau pencarian adalah proses di mana AI menemukan solusi terbaik atau optimal dari sejumlah besar kemungkinan yang ada. Pencarian merupakan bagian integral dari AI, terutama pada konteks problem-solving dan optimisasi.
   - *Breadth-First Search*: Algoritma ini menjelajahi semua kemungkinan solusi pada tingkat yang sama sebelum bergerak ke tingkat berikutnya. Ini berguna

dalam situasi di mana kita ingin menemukan solusi yang paling dekat (atau paling cepat) dari titik awal.

- *Depth-First Search*: Berbeda dengan *breadth-first search*, algoritma ini mendalami satu cabang solusi hingga ke ujung sebelum kembali dan mencoba cabang lain. Ini bisa lebih cepat dalam menemukan solusi, tetapi mungkin tidak selalu menemukan solusi terbaik.
- *A* Search*: Merupakan kombinasi dari *breadth-first search* dan *depth-first search*, yang dimana algoritma ini menggunakan heuristik untuk memperkirakan seberapa dekat sebuah solusi dari tujuan, sehingga dapat menemukan jalur yang paling optimal dengan efisiensi tinggi [9].

Searching digunakan dalam berbagai aspek, termasuk pencari informasi, pemecahan teka-teki, dan bahkan navigasi robot di lingkungan yang kompleks.

3. *Reasoning*

   Reasoning atau penalaran adalah kemampuan AI untuk membuat keputusan atau menarik kesimpulan berdasarkan data atau informasi yang tersedia. Penalaran memungkinkan AI untuk bisa berpikir secara logis dan bertindak sesuai dengan informasi yang dimiliki.

   - *Deductive Reasoning:* Dalam penalaran deduktif, AI menarik kesimpulan logis dari premis yang diketahui. Jika semua premis benar, maka kesimpulan yang dihasilkan juga akan benar. Contoh sederhana adalah penalaran dalam sistem pakar yang digunakan dalam diagnosa medis.
   - *Inductive Reasoning:* Dalam penalaran induktif, AI membuat generalisasi dari data spesifik. Meskipun kesimpulan yang dihasilkan mungkin tidak sepenuhnya benar, pendekatan ini berguna dalam membuat prediksi berdasarkan tren data yang ada.
   - Abductive Reasoning: Penalaran ini mencari penjelasan terbaik untuk suatu fakta atau observasi yang diberikan. Meskipun tidak selalu menghasilkan kesimpulan yang pasti, abductive reasoning penting dalam bidang seperti deteksi anomali dan investigasi kriminal.

   Penalaran merupakan komponen kunci dalam banyak aplikasi AI, termasuk pengambilan keputusan, sistem pakar, dan logika dalam permainan komputer

4. *Planning*

   Planning atau perencanaan adalah cabang AI untuk menentukan serangkaian tindakan yang diperlukan untuk mencapai tujuan tertentu. Sistem AI yang menggunakan perencanaan seringkali beroperasi dalam lingkungan yang dinamis, dimana mereka perlu mengantisipasi kemungkinan perubahan dan meresponsnya secara proaktif. Misalnya, robot otonom harus merencanakan jalur optimal untuk bergerak dari titik A ke titik B, dan menghindari rintangan di sepanjang jalan (sitasi). Perencanaan adalah salah satu pilar dalam AI yang digunakan dalam berbagai aspek, mulai dari robotika hingga *game* (sitasi).

   - Robotika: Dalam robotika, *planning* digunakan untuk menentukan langkah-langkah atau jalur optimal yang harus diambil oleh robot agar dapat mencapai tujuan tertentu. Misalnya, sebuah robot otonom yang bekerja di gudang harus merencanakan jalurnya untuk memindahkan barang dari satu titik ke titik lainnya sambil menghindari rintangan. Algoritma yang sering digunakan dalam hal ini seperti A*, yang memungkinkan robot untuk menemukan jalur terpendek dengan memertimbangkan berbagai kendala lingkungan [10]. Sistem perencanaan jalur telah diterapkan secara luas dalam aplikasi robotika di lingkungan industri dan medis, dimana robot harus bekerja secara independen tanpa campur tangan manusia [10].
   - Kendaraan Otonom: Pada kendaraan otonom, AI menggunakan *planning* untuk memilih rute optimal dengan mempertimbangkan kondisi lalu lintas, rambu jalan, dan cuaca. AI pada kendaraan ini menggunakan data *real-time* dari sensor untuk menentukan tindakan terbaik yang harus diambil, termasuk menghindari pejalan kaki dan kendaraan lain serta menyesuaikan kecepatan kendaraan berdasarkan kondisi jalan. Perencanaan jalur sangat penting dalam sistem otonom untuk menghadapi rintangan yang bergerak dan kondisi tidak terduga. Kendaraan ini biasanya menggunakan algoritma *trajectory planning*, yang memastikan kendaraan bergerak di jalur yang paling aman dan efisien sambil mempertimbangkan jarak tempuh dan keselamatan [11].
   - Sistem Logistik: Sistem logistik modern juga memanfaatkan *planning* untuk mengoptimalkan rute pengiriman. Dalam konteks ini AI

menggunakan algoritma perencanaan untuk menghitung rute tercepat dan paling efisien, mempertimbangkan faktor seperti lalu lintas, cuaca, dan ketersediaan sumber daya. Algoritma *vehicle routing problem* (VRP) dapat digunakan untuk merencanakan pengiriman dengan kapasitas terbatas dan permintaan stok yang tidak pasti [12].

2.3 Cara Kerja *Artificial Intelligence*

Artificial Intelligence (AI) bekerja dengan serangkaian tahapan sistematis yang memungkinkan sistem untuk dapat belajar dari data, mengenali pola, membuat prediksi dan mengambil keputusan secara otomatis tanpa ada campur tangan dari manusia secara langsung. Untuk memahami bagaimana AI bekerja, perlu diketahui langkah-langkah utama dalam prosesnya, mulai dari pengumpulan data hingga evaluasi model. Berikut adalah tahapan utama dalam cara kerja AI:

1. Pengumpulan Data dan Persiapan Data

   AI memerlukan data sebagai dasar untuk melakukan pembelajaran agar dapat menghasilkan hasil prediksi yang akurat. Data yang digunakan bisa dalam berbagai bentuk, seperti teks, gambar, suara, dan angka tergantung pada jenis masalah yang ingin diselesaikan.

   a. Pengumpulan Data

      Proses pertama dalam sistem AI adalah pengumpulan data dari berbagai sumber. Data ini dapat diperoleh dari database internal perusahaan, rekaman suara, dokumen, atau gambar yang diambil dari kamera [13]. Misalnya, pada sistem pengenalan wajah, data berupa ribuan gambar wajah dikumpulkan agar AI dapat mengenali pola unik pada struktur wajah manusia.

   b. Preprocessing Data

      Data yang dikumpulkan seringkali tidak dalam kondisi optimal untuk digunakan pada model AI. Oleh karena itu, tahap preprocessing sangat penting untuk dilakukan agar kualitas data meningkat. Berikut adalah beberapa langkah dalam preprocessing:

      - Pembersihan Data (*Data Cleaning*): Menghapus data yang tidak relevan, duplikasi data, atau data yang mengandung kesalahan. Misalnya dalam analisis sentimen, kata-kata yang tidak memiliki makna seperti “oh”,

"uh", atau karakter yang berupa simbol perlu dihapus agar analisis lebih akurat [14].

- Normalisasi: Mengubah data dalam format yang seragam. Misalnya dalam pemrosesan data numerik, normalisasi dilakukan dengan cara menyusun angka dalam rentang tertentu agar model bisa lebih mudah memahami perbedaan antar data [15].
- Transformasi Data: Mengonversi format data menjadi bentuk yang sesuai dengan model AI. Misalnya dalam pengolahan suara, sinyal audio dikonversi menjadi *log-Mel spectrogram*, yaitu representasi visual dari gelombang suara yang lebih mudah untuk diproses oleh model AI yang berbasi *deep learning*.

2. Pemilihan dan Pelatihan Model AI

   Setelah data berhasil diproses, tahap selanjutnya adalah memilih model AI yang sesuai kemudian melatih model agar dapat memahami pola dalam data. Pemilihan model bergantung pada jenis masalah yang ingin diselesaikan.

   a. Pemilihan Model

      Terdapat berbagai metode pembelajaran dalam AI, berikut adalah beberapa metode pembelajarannya:

      - Supervised Learning: Model dilatih dengan data yang sudah berlabel, dimana setiap input memiliki output yang telah diketahui. Contoh penggunaannya adalah dalam klasifikasi email, dimana sistem AI belajar untuk membedakan email sebagai "spam" atau "bukan spam" berdasarkan contoh data yang ada [16].
      - Unsupervised Learning: Model belajar dari data yang tidak memiliki label dan berusaha menemukan pola tersembunyi. Contohnya adalah teknik *clustering* yang digunakan dalam sistem rekomendasi produk *e-commerce* untuk mengelompokkan pelanggan berdasarkan kebiasaan belanja [17].
      - Reinforcement Learning: Model AI belajar melalui interaksi dengan lingkungan dan menerima umpan balik dalam bentuk *reward* dan *penalty.* Teknik ini digunakan dalam robotika dan kendaraan untuk mengoptimalkan pengambilan keputusan secara mandiri.

b. Proses Pelatihan Model

Setelah menentukan model AI yang dipilih, tahap selanjutnya adalah pelatihan model. Tahap pelatihan dilakukan agar sistem dapat memahami pola dalam data. Pelatihan model terdiri dari beberapa langkah berikut:

- *Forward Propagation*: Model menerima input dari data, kemudian diproses melalui jaringan saraf tiruan atau algoritma yang digunakan, lalu menghasilkan output awal.
- *Loss Calculation*: Model membandingkan hasil prediksi dengan nilai sebenarnya untuk mengukur seberapa besar kesalahannya. Metrik seperti *Mean Squared Error* (MSE) sering digunakan untuk mengukur kesalahan dalam model regresi.
- *Backpropagation & Gradient Descent*: Model memperbarui bobot atau parameter berdasarkan kesalahan yang telah dihitung. Proses ini berulang hingga model mencapai akurai yang optimal [10].
- Iterasi Berulang (*Epochs*): Model dilatih dengan dataset yang sama berulang kali agar semakin memahami pola dalam data untuk menghasilkan prediksi yang lebih akurat.

3. Implementasi Model AI

Setelah model selesai dilatih, model dapat digunakan untuk menganalisis data baru dan membuat prediksi atau mengambil keputusan secara otomatis. Berikut adalah contoh implementasi AI dalam berbagai bidang, antara lain:

- *Computer* Vision: AI dapat mengenali objek baik dalam bentuk gambar atau video, seperti dalam sistem pengenalan wajah yang digunakan untuk keamanan pada perangkat smartphone.
- *Natural Language Processing* (NLP): AI mampu memahami bahasa manusi, menerjemahkan teks, atau melakukan analisis sentimen dalam ulasan pelanggan.
- *Autonomous* Systems: AI diterapkan dalam mobil tanpa pengemudi untuk mendeteksi rambu lalu lintas dan mengambil keputusan dalam berkendara.

4. Evaluasi Model dan Optimasi

Setelah melakukan tahap implementasi, model AI perlu dievaluasi untuk mengukur seberapa baik kinerjanya. Evaluasi dilakukan dengan menggunakan

metrik yang sesuai dengan tugas yang sedang dikerjakan. Berikut adalah beberapa metrik yang digunakan dalam melakukan evaluasi:

- Akurasi dan Presisi: Metrik ini digunakan dalam klasifikasi untuk mengukur seberapa banyak yang benar dibandingkan dengan total prediksi yang dibuat.
- *Mean Squared Error* (MSE): Digunakan dalam model regresi untuk mengukur seberapa jauh prediksi dari nilai yang sebenarnya.
- *Precision-Recall Curve*: Digunakan dalam model klasifikasi untuk mengevaluasi keseimbangan antara jumlah prediksi benar dan salah.

Jika model belum mencapai performa yang optimal, maka ada beberapa strategi yang dapat digunakan untuk meningkatkan hasilnya:

- Menambahkan lebih banyak data pelatihan agar model memiliki lebih banyak contoh untuk dipelajari.
- Menggunakan teknik *fine-tuning* dengan menyesuaikan parameter model agar lebih sesuai dengan karakteristik data yang digunakan.
- Meningkatkan kompleksitas model dengan menggunakan arsitektur yang lebih canggih untuk menangani pola yang lebih kompleks.

2.4 *Machine learning*

Machine learning adalah disiplin ilmu yang berfokus pada dua pertanyaan yang saling terkait: Bagaimana seseorang dapat membangun sistem komputer yang secara otomatis meningkat melalui pengalaman, dan apa hukum dasar teori yang mengatur semua sistem pembelajaran, termasuk komputer, manusia dan organisasi [8]. *Machine learning* membutuhkan data yang *valid* sebagai bahan belajar sebelum digunakan ketika testing untuk hasil *audio* yang optimal. *Machine learning* dikembangkan dengan berdasarkan disiplin ilmu seperti statistika, matematika dan data *mining* sehingga mesin dapat belajar dengan menganalisa data tanpa perlu diatur ulang atau diperintah. Istilah *machine learning* pertama kali dikemukakan oleh beberapa ilmuan matematika seperti Andrien Marie Legendre, Thomas Bayes dan Andrey Markov pada tahuhjn 1920-an dengan mengemukakan konsep dasar-

dasar *machine learning* dan konsepnya. Sejak saat itu banyak yang mengembangkan *machine learning*, salah satu contoh dari penerapan *machine learning* yang cukup populer adalah Deep Blue yang dibuat oleh IBM pada tahun 1996. Deep Blue merupakan *machine learning* yang dikembangkan agar bisa belajar dan bermain catur. Deep Blue juga telah diuji coba dengan bermain catur melawan juara catur profesional dan Deep Blue berhasil memenangkan pertandingan catur tersebut.

Untuk mempelajari *machine learning* terdapat beberapa teknik yang bisa digunakan, diantaranya sebagai berikut:

1. *Supervised Learning*

   Teknik ini merupakan salah satu pendekatan yang dapat diterapkan dalam pembelajaran mesin. Teknik ini memanfaatkan data berlabel untuk melatih model agar bisa memprediksi output berdasarkan input. Teknik ini sering digunakan dalam klasifikasi dan regresi, dimana data berlabel digunakan untuk memandu proses belajar mesin agar dapat meniru pola dari pengalaman belajar sebelumnya [13].
2. *Unsupervised Learning*

   Unsupervised learning adalah teknik yang digunakan untuk menganalisis data yang tidak memiliki label atau informasi yang jelas yang dapat diaplikasikan secara langsung. Tujuan utama dari teknik ini adalah untuk menemukan pola atau struktur tersembunyi dalam data tanpa adanya panduan atau label sebelumnya. Teknik ini sering digunakan dalam *clustering*, seperti pada analisis kelompok pelanggan dalam *e-commerce*, dimana model berusaha untuk mengidentifikasi kelompok dalam data tanpa petunjuk sebelumnya [18].
3. *Semi-Supervised Learning*

   Semi-supervised learning merupakan pendekatan gabungan antara *supervised learning* dan *unsupervised learning* dalam *machine learning*. Dalam teknik ini, sebagian data yang digunakan untuk pelatihan model sudah memiliki label, namun sebagian besar data tidak memiliki label. Dengan kata lain hanya sebagian kecil data yang telah diberi keterangan atau label, sedangkan sebagian besar data masih bersifat tidak berlabel. Teknik ini memungkinkan model untuk memanfaatkan informasi dari data yang diberi label untuk meningkatkan kinerja

dan keakuratannya, dan juga mengambil manfaat dari data yang tidak berlabel untuk menemukan pola atau struktur yang lebih luas dalam *dataset* [19].

4. *Self-Supervised Learning*

   Self-supervised learning merupakan sebuah bentuk dari teknik *unsupervised learning* dimana model belajar dengan membuat label atau target dari data mentahnya sendiri. Pendekatan ini sering digunakan dalam NLP dan *computer vision*, yang dimana model memprediksi bagian yang hilang atau tersembunyi dari data [20].

5. *Reinforcement Learning*

   Reinforcement learning merupakan salah satu jenis teknik dalam *machine learning* dimana proses pelatihan (*training*) dan pengujian (*testing*) model dilakukan secara bersamaan. *Reinforcement learning* melibatkan interaksi antara agen dan lingkungan, dimana agen mendapatkan umpan balik dalam bentuk penghargaan atau hukuman. Teknik ini digunakan dalam berbagai aplikasi, termasuk permainan dan robotik, dimana agen belajar untuk mengambil keputusan melalui pengalaman [21].

Cara kerja *machine learning* bervariasi tergantung pada teknik atau metode pembelajaran yang digunakan. Namun, secara umum prinsip dasar *machine learning* tetap sama, melibatkan tahapan pengumpulan data, eksplorasi data, pemilihan model atau teknik, pelatihan model yang dipilih, dan evaluasi.

2.5 *Deep Learning*

Deep learning merupakan salah satu sub-bidang dari *machine learning* yang mengacu pada penggunaan jaringan saraf tiruan (*Artificial Neural Network*) dengan banyak lapisan (*Deep Neural Network*). Teknologi ini meniru cara kerja dari otak manusia dalam memproses data dan kemudian menciptakan pola yang digunakan untuk pengambilan keputusan [14]. *Deep learning* telah digunakan dalam berbagai aplikasi, termasuk pengenalan gambar, pengenalan suara, pemrosesan data alami (*Natural Language Processing*), dan masih banyak lagi, dimana model *deep learning* mampu menangkap pola data yang kompleks dan mendalam [18].

Deep learning melibatkan penggunaan jaringan saraf tiruan yang terdiri dari lapisan input, lapisan tersembunyi (*hidden layers*), dan lapisan *output*. Setiap lapisan memiliki neuron-neuron yang saling terhubung, kemudian informasi diproses melalui koneksi tersebut dengan bobot yang disesuaikan selama proses pelatihan. Teknik yang digunakan untuk memperbarui bobot-bobot ini dikenal dengan *backpropagation*, yang memungkinkan jaringan untuk belajar dari kesalahan dan meningkatkan kinerja model dari waktu ke waktu [22].

Semakin banyak lapisan tersembunyi yang digunakan, semakin "dalam" jaringan tersebut, sehingga mampu menangkap pola yang lebih kompleks dalam data. Ini yang membuat *deep learning* unggul dalam berbagai aplikasi yang membutuhkan analisis data yang mendalam dan kompleks [23].

Deep learning telah merevolusi banyak hal dalam bidang teknologi, terutama dalam tugas-tugas yang melibatkan pengenalan pola, seperti:

- Pengenalan gambar: *Deep learning* memungkinkan komputer untuk mengenali objek dalam gambar dengan tingkat akurasi yang tinggi. Model seperti *Convolutional Neural Networks* (CNNs) digunakan secara luas dalam deteksi wajah, klasifikasi objek, dan analisis citra medis [24].
- Pemrosesan suara: Model *deep learning* juga digunakan dalam pengenalan suara, seperti dalam aplikasi *virtual assistant* atau transkripsi otomatis, di mana model dapat mengenali kata yang diucapkan dengan sangat baik [25].

2.6 *Natural Language Processing* (NLP)

Natural Language Processing (NLP) merupakan salah satu aplikasi utama dari *deep learning* yang berkaitan dengan pemahaman dan manipulasi bahasa manusia oleh komputer. NLP berfokus pada interaksi antara komputer dan bahasa alami, yang mencakup berbagai tugas seperti penerjemahan bahasa, analisis sentimen, pengenalan entitas, dan lain-lain. Dalam beberapa dekade terakhir, NLP telah berkembang dengan sangat pesat berkat adanya kemajuan dalam *deep learning* [26].

Dengan adanya kemajuan dalam deep learning, metode tradisional dalam NLP yang sebelumnya mengandalkan fitur-fitur buatan manusia mulai tergantikan oleh model *deep learning* yang mampu belajar langsung dari data mentah. Beberapa model *deep learning* terkenal yang digunakan dalam NLP meliputi:

- *Recurrent Neural Networks* (RNNs): RNN dan variannya seperti *Long Short-Term Memory* (LSTM) digunakan untuk memproses urutan data, seperti kalimat atau paragraf, dengan menangkap konteks dan dependensi jangka panjang dalam teks [27].
- Transformers: Model seperti BERT dan GPT menggunakan arsitektur Transformer untuk memproses data secara paralel, yang memungkinkan model untuk memahami konteks dalam teks yang lebih panjang dan kompleks. Model-model ini telah mencapai performa state-of-the-art dalam berbagai tugas NLP [28].

2.7 *Speech-To-Text*

Pengenalan ucapan atau suara (*Speech Recognition*) adalah suatu teknik yang memungkinkan sistem komputer untuk menerima *input* berupa kata yang diucapkan. Salah satu bagian dari *teknologi speech recognition* adalah *Speech-To-Text* (STT). STT merupakan satu bentuk teknologi yang bisa mengenali ucapan manusia untuk kemudian dikonversikan dalam bentuk teks [29]. Sistem STT sudah dilatih untuk mengenali suara manusia, mengolahnya, menginterpretasikannya, dan mengkonversi suara menjadi tulisan, sehingga memungkinkan berbagai perangkat seperti *smartphone*, *tablet*, komputer memahami kebutuhan manusia.

Pada prinsipnya, sistem *speech-to-text* akan menerima *input* berupa suara yang nantinya akan diubah menjadi bentuk teks. Meskipun konsep ini tampak sederhana, pada kenyataannya mekanisme kerja STT cukup kompleks. Kompleksitas sistem STT disebabkan oleh penggunaan yang luas oleh berbagai individu dengan variasi bahasa, dialek, aksen, dan ekspresi yang berbeda di seluruh dunia. Selain itu, sistem harus mampu membedakan antara suara manusia dan kebisingan latar belakang

seperti suara kendaraan atau keramaian, sehingga suara-suara tersebut dapat disaring dan tidak salah diinterpretasikan oleh sistem [30].

Secara garis besar, proses kerja teknologi STT dapat dijelaskan sebagai berikut:

- Terdapat 2 elemen penting dalam penggunaan teknologi STT, yaitu mikrofon untuk menangkap dan merekam suara, serta koneksi internet untuk menghubungkan perangkat dengan *server* atau basis data. Saat pengguna berbicara melalui mikrofon, perangkat akan merekam suara dan mengirimkannya ke *server*.
- Di dalam *server*, sistem akan memecah *audio* rekaman ucapan menjadi bagian-bagian kecil yang disebut fonem. Urutan, kombinasi, dan konteks fonem ini memungkinkan sistem untuk menganalisis konteks dan sintaksis kata. Sistem akan mencocokkan pola dan kata-kata di basis data dengan ucapan pengguna menggunakan algoritma dan data yang sudah di *input* sebelumnya [30].
- Selanjutnya sistem akan membuat hipotesis tentang apa yang sebenarnya dikatakan, sistem akan mentranskripsikan percakapan menjadi teks. Proses ini berlangsung sangat cepat, hanya dalam hitungan detik.

Semakin sering sistem STT digunakan, maka semakin pintar, akurat dan cepat sistem dalam mengenali suara, serta semakin jarang melakukan kesalahan. Dengan demikian, teknologi STT menjadi lebih efektif seiring waktu dan penggunaan yang lebih luas dan mampu beradaptasi dengan berbagai variasi bahasa dan suara [31].

2.8 *Abstractive summarization*

Abstractive summarization adalah teknik *natural language processing* (NLP) yang bertujuan untuk menghasilkan ringkasan dari sebuah teks dengan menciptakan kalimat baru yang menyampaikan informasi utama dari teks asli. Kata sifat '*abstractive'* digunakan untuk menunjukkan ringkasan yang bukan hanya sekedar pilihan dari beberapa bagian atau kalimat yang sudah ada yang berasal dari sumbernya, tetapi merupakan paraphrase terkompresi dari isi utama sebuah dokumen, yang memiliki potensi untuk menggunakan kosakata yang tidak terlihat dari dokumen aslinya [32]. Berbeda dengan *extractive summarization*, yang hanya mengekstrak dan menyusun kembali kalimat-kalimat asli, *abstractive*

summarization menciptakan kalimat yang baru dan lebih ringkas yang tetap mempertahankan makna dan informasi penting dari teks asli.

Mekanisme dari *abstractive summarization*:

1. *Pre-processing*: Teks *input* perlu dibersihkan dan diproses untuk menghilangkan elemen yang tidak relevan seperti tanda baca yang berlebihan, spasi ekstra atau karakter khusus. Proses ini juga mencakup tokenisasi, yaitu memecah teks menjadi unit-unit kecil seperti kata atau frasa.
2. *Encoding*: Teks yang telah diproses kemudian diubah menjadi representasi *vector* oleh model *encoder*. Model *encoder* seringkali merupakan bagian dari arsitektur *Transformer* seperti BERT atau *encoder-decoder Transformer*, memahami dan menangkap konteks dan makna dari teks.
3. *Decoding*: Model *decoder* mengambil representasi vector yang dihasilkan oleh *encoder* dan menghasilkan teks ringkasan yang baru. Model ini mencoba menghasilkan kalimat yang koheren dan bermakna dengan merujuk pada informasi yang diperoleh selama tahap encoding.
4. *Post-processing*: Teks ringkasan yang dihasilkan akan diperiksa dan diperbaiki untuk memastikan bahwa hasil akhirnya koheren, gramatikal dan mudah dipahami. Dalam tahap ini bisa termasuk perbaikan tata bahasa, penyusunan ulang kalimat dan penghilangan informasi yang tidak relevan.

Kemudian ada beberapa model dan algoritma yang sering digunakan untuk *abstractive summarization* meliputi:

1. *Seq2Seq* (*Sequence-to-Sequence*) Models: Model ini menggunakan *encoder* dan *decoder* RNN (*Recurrent Neural Network*) atau LSTM (*Long Short-Term Memory*) untuk mengubah teks panjang menjadi teks pendek.
2. *Transformer-based Models*: Model seperti BERT, GPT dan T5 menggunakan arsitektur *Transformer* yang lebih efisien dalam menangkap hubungan jangka panjang dalam teks. T5 (*Text-to-Text Transfer Transformer*) khususnya dirancang untuk berbagai tugas NLP dengan mengubah semua tugas menjadi *text-to-text*.

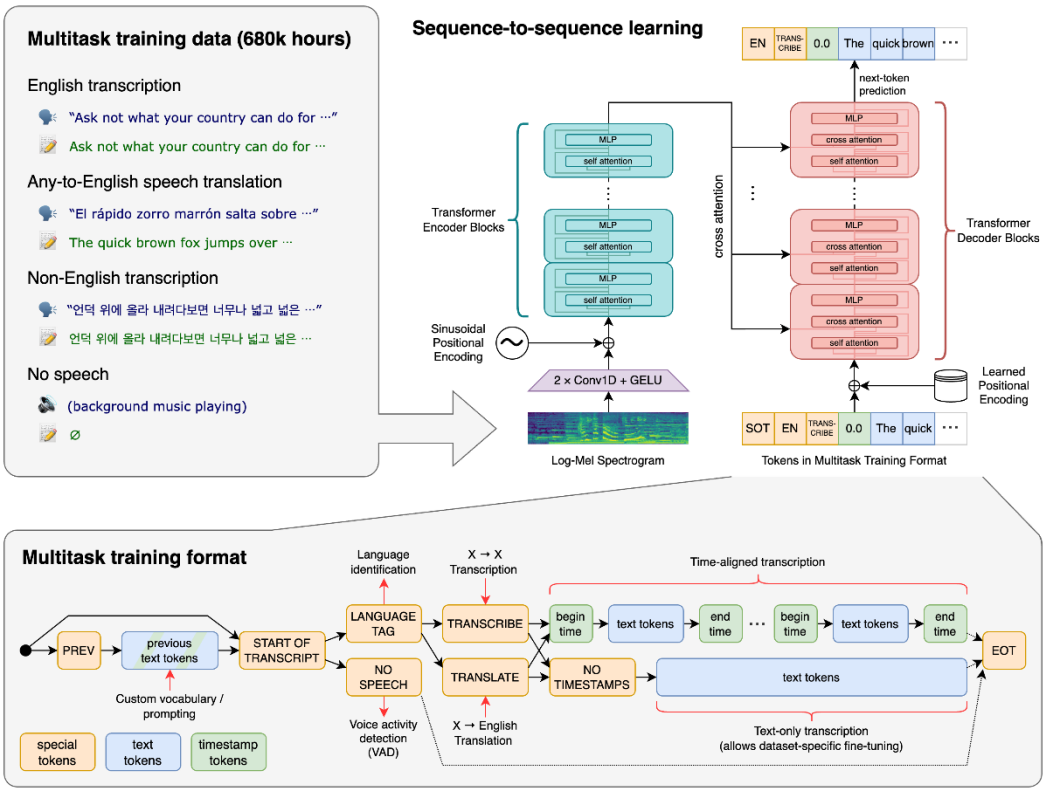
3. *Pointer-Generator Networks*: Model ini menggabungkan teknik *extractive* dan *abstractive* dengan memungkinkan model memilih antara menyalin kata langsung dari teks asli atau menghasilkan kata baru.

2.9 Model Whisper

Model Whisper adalah model *speech-to-text* dari OpenAi yang digunakan untuk menyalin file *audio*. Model ini dilatih dengan menggunakan kumpulan data *audio* dan teks berbahasa inggris yang besar. Model ini dioptimalkan untuk menyalin file *audio* yang berisi ucapan dalam *Bahasa Inggris*. Selain dalam Bahasa Inggris, model ini juga dapat digunakan untuk mentranskripsikan *audio* yang berisi ucapan dalam Bahasa lain. Dari total 680.000 jam *audio* berlabel yang digunakan oleh Whisper, sebanyak 117.000 jam berasal dari 96 bahasa selain Bahasa Inggris [33]. Penggunaan data yang besar dan beragam menghasilkan peningkatan ketahanan terhadap aksen, kebisingan latar belakang (*noise background*) dan bahasa teknis. Proses model *audio* melalui sistem *Transformer block* dengan residual koneksi dan normalisasi lapisan akhir. Model ini menggunakan format *multitask* untuk melakukan seluruh pemrosesan suara *pipeline*, termasuk *transkripsi*, terjemahan, deteksi aktivitas suara. Whisper menggunakan pendekatan *self-supervised learning*, dimana model dilatih untuk mengonversi data *audio* menjadi teks tanpa memerlukan pelabelan manual. Teknik ini memungkinkan model untuk belajar dari data mentah dan memprediksi output secara mandiri berdasarkan pola yang ditemukan di dalam data. Teknik ini mirip dengan yang digunakan dalam model Wa2Vec untuk pengenalan suara otomatis [34].

Arsitektur whisper adalah pendekatan *end-to-end* yang sederhana, diimplementasikan sebagai *Transformer encoder-decoder*. *Audio* yang telah dimasukan dibagi menjadi potongan-potongan berdurasi 30 detik, diubah menjadi *log-Mel spectrogram* dan kemudian diteruskan ke *encoder*. *Decoder* dilatih untuk memprediksi keterangan teks yang sesuai, yang kemudian dicampur dengan *token* khusus yang mengarahkan model tunggal untuk melakukan tugas seperti identifikasi bahasa, *timestamps* tingkat frasa, transkripsi ucapan multibahasa, dan terjemahan ucapan ke Bahasa Inggris.

Pendekatan lain yang ada seringkali menggunakan kumpulan data pelatihan *audio-text* yang lebih kecil dan berpasangan lebih erat, atau menggunakan *pre-training audio* yang yang luas namun tanpa pengawasan. Karena whisper dilatih pada kumpulan data yang besar dan beragam serta tidak disesuaikan dengan kumpulan data tertentu. Whisper tidak dalam mengungguli model yang berspesialisasi dalam performa *LibriSpeech*, sebuah tolak ukur kompetitif yang terkenal dalam pengenalan ucapan. Namun, ketika whisper diukur melalui kinerja *zero-shot* di banyak kumpulan data yang berbeda, kinerja dari whisper lebih kuat dan menghasilkan kesalahan 50% lebih sedikit dibandingkan model lain. Sekitar sepertiga dari kumpulan data *audio* whisper adalah Bahasa non-Inggris, dan secara bergantian diberi tugas untuk menyalin dalam Bahasa asli atau menerjemahkan ke Bahasa Inggris. Pendekatan ini sangat efektif dalam mempelajari terjemahan ucapan teks dan mengungguli SOTA yang diawasi CoVoST2 ke terjemahan bahasa Inggris *zero-shot*.



Gambar 1. Gambaran Umum Model Whisper

Sumber : A. Radford., J. W. Kim., T. Xu., G. Brockman., C. McLeavey., dan I. Sutskever. 2022. “Robust Speech Recognition via Large-Scale Weak Supervision,” OpenAI Research Paper. Tersedia : whisper.pdf

Gambar 1 merupakan gambaran keseluruhan alur proses pada model Whisper yang mengintegrasikan beberapa tugas pemrosesan suara ke dalam satu sistem terpadu. Berikut adalah penjelasannya:

1. Pengolahan Sinyal Audio Mentah
    - Input Audio & *Resampling*: Audio asli berupa gelombang (*waveform*) dengan amplitudo yang berbeda dan berubah-ubah pertama-tama diubah laju samplingnya menjadi 16.000 Hz agar konsisten.
    - Pemisahan *frame* dan Transformasi *Fourier*: Sinyal audio dibagi menjadi potongan-potongan pendek (*frame*) dengan jendela 25 milidetik dan *stride* 10 milidetik. Setiap *frame* diubah ke *domain* frekuensi menggunakan teknik *Short-Time Fourier Transform* (STFT).
    - *Filter Bank Mel & Log Transform*: Hasil STFT, yang berupa spektrum frekuensi, kemudian dipetakan ke dalam 80 filter *bank* yang disusun berdasarkan skala *Mel* (skala yang meniru cara pendengaran manusia). Setelah itu, diambil logaritma dari nilai magnitude tiap *filter* untuk menghasilkan *log-mel spectrogram*. Transformasi ini mengompres rentang dinamis data sehingga perbedaan kecil pada frekuensi yang penting bagi pendengaran manusia menjadi lebih mudah untuk dideteksi.
2. Ekstraksi Fitur Awal dengan *Convolutional Stem*
    - Normalisasi: *Log-mel spectrogram* yang dihasilkan dinormalisasi sehingga nilai-nilainya berada dalam rentang antara -1 dan 1 dengan rata-rata mendekati nol. Normalisasi ini membantu stabilitas dalam pelatihan model.
    - Lapisan Konvolusional: Data yang telah dinormalisasi kemudian diproses oleh dua lapisan Conv1D. Lapisan konvolusional ini bertugas mendeteksi pola-pola lokal dalam spektrum, seperti fitur fonetik (misalnya, suara vokal atau konsonan) yang muncul pada potongan-potongan pendek audio.

- Fungsi Aktivasi GELU: Setiap lapisan konvolusional diikuti oleh fungsi aktivasi GELU yang memperkenalkan non-linearitas, sehingga model dapat mempelajari fitur yang lebih kompleks dari sinyal suara.

3. Pembentukan Representasi Konstektual dengan Transformer *Encoder*
   - Penambahan *Positional Encoding*: Dikarenakan data audio bersifat sekuensial, informasi urutan sangat penting. Oleh karena itu, setelah melalui lapisan konvolusional, vector representasi ditambahkan dengan *sinusoidal positional encoding* yang memberi tahu model posisi tiap *frame* dalam urutan.
   - Proses *Self-Attention*: Data yang telah di-*enrich* dengan informasi posisi diproses oleh beberapa lapisan Transformer *encoder*. Di sini, mekanisme *self-attention* memungkinkan model untuk memperhatikan semua bagian dari urutan audio sekaligus, sehingga dapat memahami konteks global, misalnya hubungan antara bunyi yang terjadi terpisah secara waktu.
4. Konversi Representasi Audio Menjadi Teks dengan Transformer *Decoder*
   - Tokenisasi Awal dan Spesifikasi Tugas: *Decoder* melalui proses dengan token khusus, misalnya token <|startoftranscript|> yang menandai awal transkripsi. Selain itu, token-token lain disisipkan untuk menentukan tugas yang diinginkan, seperti:
     - Token bahasa untuk mengidentifikasi bahasa audio (misalnya, token khusus untuk Bahasa Indonesia dan Bahasa Inggris).
     - Token tugas seperti <|transcribe|> untuk transkripsi atau <|translate|> untuk penerjemahan.
     - Token <|nospeech|> digunakan jika tidak ada suara yang terdeteksi.
     - Token untuk penentuan *timestamp* disisipkan jika diperlukan, membatu model memberikan informasi waktu untuk setiap segmen kata.
   - Proses *Autoregresif*: *Decoder* bekerja secara *autoregresif*, artinya pada setiap langkah, model menggunakan output token sebelumnya sebagai konteks untuk memprediksi token selanjutnya. Mekanisme *cross-attention* di *decoder* memungkinkan informasi konstektual dari *encoder* (yang telah meringkas audio) digunakan untuk menghasilkan token teks yang akurat.

- Rekonstruksi Teks: Pada akhirnya, deretan token yang diprediksi disusun kembali menjadi kalimat utuh, sehingga menghasilkan transkripsi yang berasal dari sinyal audio asli.

2.10 Model Pegasus

Pegasus adalah sebuah model bahasa alami yang dikembangkan oleh tim peneliti Google Research. Model ini pertama kali diperkenalkan dalam sebuah makalah penelitian yang diterbitkan pada tahun 2020 oleh tim Google, dengan judul "Pegasus: *Pre-training with Extracted Gap-sentences for Abstractive summarization*". Pendekatan yang digunakan dalam pengembangan model Pegasus serta hasil eksperimen yang menunjukkan kinerja model tersebut dalam tugas-tugas pemrosesan bahasa alami terutama dalam pembuatan ringkasan teks.

Pegasus menggunakan *self-supervised learning* dengan teknik *gap-sentence-generation* (GSG). Teknik GSG memungkinkan model untuk lebih efektif dalam tugas-tugas peringkasan teks abstraktif [35]. Dalam model Pegasus pendekatan yang digunakan disebut sebagai "*Pre-training with Extracted Gap-sentences*" atau *Pre-training* dengan kalimat-kalimat celah yang diekstraksi. Model ini secara khusus didesain untuk tugas ringkasan teks yang bersifat abstraktif, artinya model ini tidak hanya mengekstrak potongan-potongan teks tertentu dari dokumen sumber, tetapi juga memahami dan menghasilkan ringkasan yang lebih bersifat konseptual.

Proses *pre-training* Pegasus melibatkan dua tahap utama, pertama model diberi masukan teks yang panjang dan kemudian dipilih sebagai beberapa kalimat sebagai "kalimat-kalimat celah" yang dianggap penting untuk diingat. Selanjutnya model dilatih untuk mengisi celah-celah ini dengan konten yang relevan dan informatif untuk menghasilkan ringkasan yang akurat. Dalam *pre-training* Pegasus, beberapa kalimat utuh dihapus dari dokumen dan model ditugaskan untuk memulihkannya [35]. Contoh masukan untuk *pre-training* adalah dokumen dengan kalimat-kalimat yang hilang, sedangkan *audio* yang dihasilkan terdiri dari kalimat-kalimat hilang yang digabungkan menjadi satu. Ini adalah tugas yang sangat sulit dan mungkin tampak mustahil, bahkan bagi manusia. Tapi, tugas yang menantang tersebut

mendorong model untuk mempelajari bahasa dan fakta umum tentang dunia, serta cara menyaring informasi yang diambil dari seluruh dokumen untuk menghasilkan keluaran yang sangat mirip dengan tugas penyempurnaan peringkasan.

Pegasus telah menunjukkan kinerja yang sangat baik dalam beberapa tugas pemrosesan bahasa alami, terutama dalam tugas ringkasan teks. Model ini telah berhasil menghasilkan ringkasan yang informatif dan relevan dari teks yang panjang dan kompleks.

2.11 Model GPT

GPT (*Generative Pre-trained Transformer*) adalah model Bahasa berbasis arsitektur *Transformer* yang dikembangkan oleh OpenAI. GPT menggunakan *unsupervised learning*, dimana model dilatih untuk memprediksi kata berikutnya dalam urutan teks tanpa memerlukan label eksplisit. Teknik ini memungkinkan GPT untuk mempelajari pola bahasa secara alami dari data teks yang besar. GPT dirancang untuk menghasilkan teks yang manusiawi yang koheren berdasarkan *input* yang diberikan, hal tersebut membuat model ini sangat berguna dalam berbagai aplikasi *Natural Language Processing* (NLP) seperti penerjemahan, penulisan teks otomatis, *chatbots*, dan banyak lagi [36].

Karakteristik Utama Model GPT:

1. Arsitektur Transformer:
   GPT didasarkan pada arsitektur Transformer. Transformer mengandalkan mekanisme perhatian (*attention mechanism*) untuk memproses urutan input dan output, memungkinkan model untuk mempertimbangkan hubungan antara semua kata dalam kalimat secara bersamaan, tidak hanya secara berurutan seperti pada model RNN (*Recurrent Neural Networks*).
2. Pre-training dan Fine-tuning:
   - Pre-training: Pada tahap ini, GPT dilatih secara tidak diawasi (unsupervised) pada kumpulan data teks yang sangat besar untuk memprediksi kata berikutnya dalam sebuah kalimat. Selama pre-training, model belajar memahami struktur bahasa dan pola-pola umum dalam teks.

- Fine-tuning: Setelah pre-training, model kemudian di-*finetune* pada *dataset* yang lebih kecil dan lebih spesifik dengan supervisi (*supervised learning*), disesuaikan dengan tugas tertentu seperti analisis sentimen, penjawaban pertanyaan, atau terjemahan.

3. Versi GPT:
   - GPT-1: Model GPT pertama yang dirilis pada tahun 2018, memiliki 117 juta parameter dan menunjukkan kemampuan dalam berbagai tugas NLP.
   - GPT-2: Diperkenalkan pada tahun 2019, dengan 1,5 miliar parameter, GPT-2 mengejutkan dunia dengan kemampuannya menghasilkan teks yang sangat realistis.
   - GPT-3: Dirilis pada tahun 2020, GPT-3 memiliki 175 miliar parameter, menjadikannya salah satu model bahasa terbesar dan paling canggih pada masanya, mampu melakukan berbagai tugas dengan sedikit contoh (*few-shot learning*).
   - GPT-3.5-turbo: Merupakan varian khusus dari GPT-3.5 yang dioptimalkan untuk penggunaan dalam aplikasi komersial dan produk OpenAI seperti ChatGPT. Versi ini dirancang untuk memberikan respons yang lebih cepat dan lebih efisien, sehingga ideal untuk digunakan dalam layanan yang memerlukan interaksi *real-time* dengan pengguna.
   - GPT-4: Merupakan lanjutan dari GPT-3 dan dirilis pada Maret 2023. GPT-$ memiliki kemampuan yang lebih canggih, dengan peningkatan dalam memahami konteks yang lebih komplek dan kemampuan yang lebih baik dalam menyelesaikan tugas-tugas yang memerlukan penalaran lebih dalam. GPT-4 juga mendukung *input* berupa teks dan gambar, yang memperluas fungsionalitas model dibandingkan pendahulunya.
4. Kemampuan Generatif:

   GPT tidak hanya dapat memproses teks tetapi juga menghasilkan teks baru berdasarkan konteks yang diberikan. Ini mencakup menulis esai, puisi, kode pemrograman, dan banyak lagi, menjadikannya alat yang sangat fleksibel untuk aplikasi kreatif dan praktis.

Aplikasi Model GPT:

- Penulisan Otomatis: Menulis artikel, cerita, atau konten pemasaran.
- Penerjemahan Bahasa: Menerjemahkan teks dari satu bahasa ke bahasa lain.
- *Chatbots* dan Asisten *Virtual*: Memberikan tanggapan otomatis dalam percakapan berbasis teks.
- Penjelasan Kode: Membantu dalam menulis dan menjelaskan kode pemrograman.

2.12 *Audio*

Audio adalah bentuk sinyal merepresentasikan gelombang suara dan dapat didengar oleh manusia. *Audio* merupakan bagian yang dapat didengar pada spektrum frekuensi suara, berbeda dengan suara yang digunakan oleh hewan tertentu atau digunakan dalam ilmu sains dan kedokteran [37].Berdasarkan Kamus Besar Bahasa Indonesia edisi ketiga, *audio* merupakan alat peraga yang bersifat dapat didengar. *Audio* merupakan alat peraga yang bersifat dapat didengar, *audio* berasal dari kata *audible*, yang berarti suaranya dapat didengar secara wajar oleh telinga manusia. Ini melibatkan pengukuran dan representasi dari variasi tekanan udara sepanjang waktu. Sinyal *audio* dapat direkam, disimpan, dan direproduksi menggunakan berbagai perangkat, seperti *mikrofon*, perekam *audio*, dan pemutar *audio*. Berikut adalah beberapa aspek penting mengenai *audio*:

1. Gelombang suara: *Audio* merupakan representasi dari gelombang suara yang ditangkap melalui mikrofon dan diproses kemudian diputar kembali melalui speaker atau headphone.
2. Digitalisasi: Suara dapat direkam dan disimpan dalam format digital menggunakan teknologi seperti *sampling* dan *quantization*, yang memungkinkan untuk *editing,* distribusi dan *playback* yang lebih fleksibel.
3. Kualitas *audio*: Kualitas *audio* dapat diukur dengan berbagai parameter seperti *bit rate*, *sample rate* dan format file misalnya, MP3, WAV, FLAC.

2.13 ROUGE

ROUGE (*Recall-Oriented Understudy for Gisting Evaluation*) adalah sekumpulan metrik yang digunakan untuk mengevaluasi kualitas ringkasan teks dengan membandingkannya dengan satu atau lebih ringkasan referensi (*gold-standard*). Metrik ini dirancang untuk mengukur tumpang tindih antara *n-grams*, kata-kata, dan subsekuensi dari teks yang dihasilkan dan teks referensi [38].

Recall, Precision dan F-Score.

1. *Recall*

   Recall mengukur seberapa baik model dalam menemukan semua item relevan dalam *dataset*. Dalam konteks ROUGE, *recall* adalah proporsi *n-grams* dalam teks referensi yang ditemukan di dalam teks yang dihasilkan.
2. *Precision*

   Precision mengukur seberapa baik model dalam menghasilkan hanya item-item yang relevan. Dalam konteks ROUGE, *precision* adalah proporsi *n-grams* dalam teks yang dihasilkan, yang ditemukan di dalam teks referensi.
3. *F-Score*

   F-Score atau *F1-Score* adalah rata-rata harmonis dari *precision* dan *recall*. Ini digunakan untuk menyeimbangkan keduanya dan memberikan satu angka evaluasi.

ROUGE memiliki beberapa varian yang masing-masing mengukur aspek berbeda dari kesamaan antara teks yang dihasilkan dengan teks referensi, berikut adalah variannya :

1. ROUGE-N

   ROUGE-N mengukur tumpang tindih *n-grams* antara teks yang dihasilkan dengan teks referensi. *N-grams* adalah sekuensi dari n kata berturut-turut dalam teks.

   - Rumus ROUGE-N Recall

     ROUGE-N Recall =

$$\frac{\sum_{S \in RefSummaries} \sum_{ngrams \in S} Count_{match}(ngrams)}{\sum_{S \in RefSummaries} \sum_{ngrams \in S} Count\ (ngrams)}$$

- Rumus ROUGE-N Precision

  ROUGE-N Precision =

$$\frac{\sum_{S \in RefSummaries} \sum_{ngrams \in S} Count_{match}(ngrams)}{\sum_{S \in RefSummaries} \sum_{ngrams \in S} Count\ (ngrams)}$$

- Rumus ROUGE-N F1-Score

  ROUGE-N F1-Score =

$$\frac{2\ x\ ROUGE-N\ Precision\ x\ ROUGE-N\ Recall}{ROUGE-N\ Precision + ROUGE-N\ Recall}$$

2. ROUGE-L

   ROUGE-L mengukur panjang subsekuensi umum terpanjang (*Longest Common Subsequence*, LCS) antara dua teks. LCS mempertahankan urutan kata asli dan mengukur berdasarkan subsekuensi terpanjang yang muncul di kedua teks.

   - Rumus ROUGE-L Recall

     ROUGE-L Recall =

$$\frac{LCS(X,Y)}{m}$$

   - Rumus ROUGE-L Precision

     ROUGE-L Precision =

$$\frac{LCS(X,Y)}{n}$$

   - Rumus ROUGE-L F1-Score

     ROUGE-L F1-Score =

$$\frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}}$$

3. ROUGE-W

   ROUGE-W adalah varian ROUGE-L yang memberikan bobot lebih pada LCS yang lebih panjang. Ini memperhitungkan pentingnya subsekuensi panjang dengan memberikan bobot pada kecocokan panjang.

4. ROUGE-S

   ROUGE-S mengukur tumpang tindih *skip-bigram*, yang memperhitungkan dua kata yang muncul dalam urutan yang sama di kedua teks, meskipun ada kata lain yang terpisah di antara keduanya.

5. ROUGE-SU

   ROUGE-SU adalah kombinasi ROUGE-S dan *unigram*, yang memperhitungkan *skip-bigram* serta tumpang tindih *unigram*.

Berikut adalah contoh implementasi dari ROUGE-N dan ROUGE-L:

A. Contoh Implementasi ROUGE-N

Misalkan kita memiliki dua teks seperti berikut:

- Teks referensi (X): "*The cat is on the mat.*"
- Teks yang dihasilkan (Y): "*The cat sat on the mat.*"

Langkah 1: Mengidentifikasi *N-grams*

Untuk ROUGE-1 (unigram), identifikasi setiap kata sebagai unigram.

- Unigram dalam teks referensi (X):
  "*The*", "*cat*", "*is*", "*on*", "*the*", "*mat*".
- Unigram dalam teks yang dihasilkan (Y):
  "*The*", "*cat*", "*sat*", "*on*", "*the*", "*mat*".

Langkah 2 : Menghitung *Recall*

Recall untuk ROUGE-1 dihitung dengan membagi jumlah unigram yang cocok antara Y dan X dengan total unigram dalam teks referensi.

- Jumlah unigram yang cocok:
  "*The*", "*cat*", "*on*", "*the*", "*mat*" (5 *unigram* cocok).
- Total *unigram* dalam teks referensi: 6

$$Recall_{ROUGE-1} = \frac{5}{6} = 0.833$$

Langkah 3 : Menghitung *Precision*

Precision untuk ROUGE-1 dihitung dengan membagi jumlah unigram yang cocok dengan total unigram dalam teks yang dihasilkan.

- Jumlah unigram yang cocok: 5
- Total unigram dalam teks yang dihasilkan: 6

$$Precision_{ROUGE-1} = \frac{5}{6} = 0.833$$

Langkah 4: Menghitung *F1-Score*

F1-Score untuk ROUGE-1 dihitung dengan menggunakan rumus:

- $F1 - Score_{Rouge-1} = \frac{2\ x\ ROUGE-N\ Precision\ x\ ROUGE-N\ Recall}{ROUGE-N\ Precision+ROUGE-N\ Recall}$
- $F1 - Score_{Rouge-1} = \frac{2\ x\ 0.833\ x\ 0.833}{0.833+0.833} = 0.833$

Menghitung ROUGE-2

Untuk menghitung ROUGE-2, identifikasi pasangan kata berturut-turut sebagai *bigram*.

- *Bigram* dalam teks referensi (X):
  "*The cat*", "*cat is*", "*is on*", "*on the*", "*the mat*".
- *Bigram* dalam teks yang dihasilkan:
  "*The cat*", "*cat sat*", "*sat on*", "*on the*", "*the mat*".
- Jumlah *bigram* yang cocok:
  "*The cat*", "*on the*', "*the mat*" (3 *bigram* cocok)
- Total *bigram* dalam teks referensi: 5
- Total *bigram* dalam teks yang dihasilkan 5

$$Recall_{ROUGE-2} = \frac{3}{5} = 0.6$$

$$Precision_{ROUGE-1} = \frac{3}{5} = 0.6$$

$$F1 - Score_{Rouge-2} = \frac{2\ x\ 0.6\ x\ 0.6}{0.6+0.6} = 0.6$$

B. Contoh Implementasi ROUGE-L

Misalkan kita memiliki dua teks seperti berikut:

- Teks referensi (X): "*The cat is on the mat*."
- Teks yang dihasilkan (Y): "*The cat sat on the mat*."

Langkah 1: Mengidentifikasi *Longest Common Subsequence* (LCS)

Pada contoh ini, subsekuensi umum terpanjang "LCS" adalah "*The cat on the mat*".

- Panjang LCS (Y, X): 4

Langkah 2: Menghitung *Recall*

$$Recall_{ROUGE-L} = \frac{LCS(X,Y)}{m} = \frac{4}{6} = 0.667$$

Dimana total kata dalam teks referensi (X) adalah 6.

Langkah 3: Menghitung *Precision*

$$Precision_{ROUGE-L} = \frac{LCS(X,Y)}{m} = \frac{4}{6} = 0.667$$

Dimana total kata dalam teks yang dihasilkan (Y) juga adalah 6.

Langkah 4: Menghitung *F1-Score*

Karena *recall* dan *precision* sama-sama bernilai 0.667, maka *F1-Score* untuk ROUGE-L akan sama dengan nilai tersebut:

$$F1 - Score_{Rouge-1} = \frac{2\ x\ 0.667\ x\ 0.667}{0.667 + 0.667} = 0.667$$

2.14 *Likert scale*

Likert scale adalah salah satu metode yang paling umum digunakan dalam penelitian survei guna mengukur sikap, opini, atau persepsi individu terhadap suatu subjek atau pernyataan. Metode ini diperkenalkan oleh Rensis Likert pada tahun 1932 dalam makalahnya yang berjudul "*A Technique for the Measurement of Attitudes*", yang dimana beliau mengusulkan cara untuk mengukur sikap secara kuantitatif melalui serangkaian pernyataan yang dapat direspon dengan tingkat persetujuan yang berbeda-beda.

A. Struktur *Likert scale*

Likert scale biasanya terdiri dari sejumlah pernyataan yang berkaitan dengan topik yang ingin diukur. Setiap pernyataan disertai dengan skala ordinal yang biasanya memiliki lima sampai dengan sembilan tanggapan. Berikut adalah contoh skala 5 titik yang biasa digunakan:

1. Sangat tidak setuju
2. Tidak setuju
3. Netral
4. Setuju
5. Sangat setuju

Skala 7 titik atau 9 titik juga sering digunakan untuk memberikan variasi lebih lanjut dalam tanggapan, memberikan responden pilihan yang lebih detail untuk mengekspresikan tingkat persetujuan mereka [39].

B. Tahapan Penggunaan *Likert scale*

Berikut adalah tahapan yang perlu dilakukan dalam penggunaan metode *likert scale*:

1. Penyusunan Pernyataan

   Penyusunan pertanyaan adalah tahap awal dan paling penting. Pernyataan harus relevan dengan topik yang diteliti dan dirancang untuk mengukur aspek tertentu dari sikap atau opini yang ingin dieksplorasi. Misalnya, dalam penelitian kepuasan pelanggan, pernyataannya bisa berupa "Layanan sangat memuaskan" [40].
2. Distribusi Kuesioner

Setelah pernyataan berhasil disusun, pertanyaan-pertanyaan tersebut dikumpulkan dalam bentuk kuesioner yang akan disebarkan kepada responden. Responden kemudian diminta untuk menilai setiap pernyataan sesuai dengan skala likert yang telah disediakan. Tanggapan ini kemudian digunakan untuk mengukur sikap atau opini secara keseluruhan.

3. Pengkodean Data

   Setelah tanggapan dari responden dikumpulkan, tanggapan tersebut kemudian dikodekan menjadi nilai numerik. Misalnya, dalam skala 5 titik, "Sangat tidak setuju" diberi kode 1, dan "Sangat setuju" diberi kode 5. Pengkodean ini memungkinkan data kualitatif yang dikumpulkan untuk dianalisis secara kuantitatif.

4. Analisis Data

   Data yang telah berhasil dikodekan kemudian dianalisis menggunakan teknik statistik. Analisis deskriptif seperti mean (rata-rata), median, dan moden sering digunakan untuk menggambarkan distribusi tanggapan. Teknis analisis lebih lanjut, seperti uji-t atau ANOVA, dapat digunakan untuk membandingkan kelompok atau menguji hipotesis.

5. Interpretasi dan Pelaporan

   Hasil dari analisis data kemudian diinterpretasikan untuk menjawab pertanyaan penelitian. Temuan ini dilaporkan dalam format yang sesuai, seperti laporan penelitian atau jurnal. Pelaporan seringkali disertai dengan grafik atau tabel yang menunjukkan distribusi tanggapan serta analisis statistik yang mendukung kesimpulan dari penelitian.

C. Kelebihan *Likert scale*

Likert scale memiliki kelebihan sebagai berikut:

1. Fleksibilitas

   Likert scale dapat digunakan dalam berbagai konteks penelitian, mulai dari psikologi, Pendidikan, pemasaran, hingga penelitian sosial lainnya. Hal ini membuat metode *likert scale* menjadi alat yang sangat berguna dalam melakukan penelitian.

2. Kemudahan Dalam Penggunaan

Skala ini mudah dipahami oleh responden dan mudah diadministrasikan oleh peneliti. Hasil yang diperoleh juga mudah dikodekan dan dianalisis, memungkinkan untuk melakukan pengolahan data yang efisien.

3. Data Kuantitatif

   Likert scale mengubah data kualitatif menjadi data kuantitatif yang dapat dianalisis secara statistik, memungkinkan peneliti untuk melakukan analisis lebih lanjut yang dapat memberikan wawasan mendalam tentang sikap dan opini responden.

D. Kekurangan *Likert scale*

Selain kelebihan, metode *likert scale* juga memiliki kekurangan. Berikut kekurangannya:

1. Bias Tengah

   Responden mungkin cenderung menjawab dengan memilih tanggapan yang berada di titik tengah skala misalnya, “Netral” untuk menghindari memberikan jawaban yang ekstrem. Hal ini dapat mengurangi akurasi pengukuran sikap yang sebenarnya.

2. Bias Sosial

   Responden cenderung memberikan tanggapan yang mereka anggap lebih dapat diterima secara sosial daripada yang benar-benar mereka rasakan. Hal tersebut dapat menyebabkan distorsi dalam hasil penelitian.

3. Tidak Mengukur Intensitas Sikap

   Meskipun *likert scale* dapat mengukur arah sikap, metode ini tidak dapat selalu menangkap intensitas perasaan atau sikap responden dengan baik. Misalnya, dua orang yang memilih “Setuju” mungkin memiliki tingkat persetujuan yang berbeda, namun hal ini tidak dapat tercermin dalam hasil skala.

Dengan memahami cara kerja dan penerapan dari metode *likert scale*, peneliti dapat lebih efektif dalam mengukur dan menganalisis sikap, opini, atau persepsi dari populasi yang diteliti. Skala ini memberikan metode yang andal dan efisien untuk mendapatkan data yang dapat ditindaklanjuti, baik dalam penelitian akademik maupun aplikasi praktis.

2.15 Streamlit

Streamlit adalah sebuah kerangka kerja (*framework*) *open-source* berbasis python yang dirancang secara khusus untuk memudahkan dan mempercepat proses pembuatan aplikasi web interaktif. Tujuan utama dari Streamlit adalah untuk memungkinkan para ahli di bidang data dan kecerdasan buatan (AI) untuk mengubah skrip analisis data atau model komputasi menjadi sebuah aplikasi web yang fungsional tanpa memerlukan keahlian mendalam dalam membuat website.

Paradigma yang diusung Streamlit berbeda secara fundamental dari kerangka kerja web tradisional seperti Django atau Flask. Streamlit mengadopsi model eksekusi berbasis skrip (*script-based*) dan reaktif [41]. Artinya, seluruh logika aplikasi ditulis dalam satu skrip python secara *linear*. Tidak ada konsep *routing* URL yang kompleks atau arsitektur *Model-View-Controller* (MVC) yang perlu dipelajari.

Fungsionalitas Streamlit dibangun di atas serangkaian API yang sederhana namun kuat, yang dapat dikategorikan ke dalam beberapa kelompok utama: elemen tampilan, *widget* interaktif, manajemen tata letak, dan optimasi kinerja.

1. Komponen Tampilan (*Display Elements*)

   Komponen ini berfungsi untuk menyajikan informasi kepada pengguna secara terstruktur. Fungsi-fungsi seperti *st.tittel()*, *st.header()*, *st.subheader()* dan *st.text()* digunakan untuk membuat hierarki teks, sementara *st.markdown()* memungkinkan penggunaan sintaks *markdown* untuk format teks yang lebih kaya [42]. Selain teks, Streamlit juga mendukung tampilan berbagai jenis media secara *native*. Misalnya *st.audio()* dapat digunakan untuk memutar *file audio* yang diunggah, dan *st.video()* untuk menampilkan video.
2. *Widget* Interaktif (*Input Widgets*)

   Ini adalah inti dari kemampuan interaktif Streamlit, yang memungkinkan aplikasi untuk menerima *input* dari pengguna dan mengubah perilakunya secara dinamis. *St.button()* menciptakan tombol yang saat ditekan akan mengembalikan nilai *true* untuk satu kali eksekusi skrip, ideal untuk memicu proses komputasi. *St.file_uploader()* menyediakan antarmuka bagi pengguna untuk mengunggah *file* dari lokal, yang sangat krusial dalam penelitian ini untuk mengunggah *file audio* rekaman rapat, perkuliahan, atau seminar.

3. Manajemen Tata Letak

   Untuk membangun aplikasi yang lebih kompleks dan terorganisir, Streamlit menyediakan beberapa fungsi untuk mengelola tata letak. Seperti *st.sidebar* yang berfungsi untuk menempatkan *widget* di panel samping yang dapat disembunyikan, berguna untuk menempatkan kontrol dan opsi tanpa mengganggu area konten utama[43].

4. Optimasi Kinerja

   Mengingat paradigma eksekusi ulang skrip, komputasi yang berat dapat membuat web menjadi lambat. Streamlit mengatasi ini melalui mekanisme *caching* yang cerdas dengan menggunakan decorator python. Dekorator *@st.cache_data* dirancang untuk menyimpan *output* dari fungsi yang mengembalikan data (seperti memuat *dataset* dari disk atau API). Jika fungsi dipanggil lagi dengan argument *input* yang sama, maka Streamlit akan melewatkan eksekusi dan langsung mengembalikan hasil yang tersimpan di *cache*.

2.16 Ngrok

Dalam alur pengembangan perangkat lunak, seringkali muncul kebutuhan untuk mengekspos layanan yang berjalan di lingkungan lokal ke publik. Kebutuhan ini krusial untuk berbagai skenario, seperti mendemonstrasikan prototipe kepada klien, menguji integrasi dengan layanan pihak ketiga melalui *webhook*, atau seperti dalam konteks penelitian ini yang membuat aplikasi web yang berjalan di lingkungan komputasi terisolasi seperti Google Colab agar dapat diakses secara public. Ngrok hadir sebagai teknologi pendukung yang esensial untuk mengatasi tantangan ini.

Ngrok secara fundamental adalah sebuah layanan *reverse proxy* terdistribusi global yang menciptakan "terowongan" (*tunnel*) aman dari sebuah *endpoint* publik di internet ke layanan jaringan yang berjalan di mesin lokal [44]. Tujuan utamanya adalah untuk menyederhanakan proses yang secara tradisional sangat kompleks, yaitu melewati rintangan jaringan seperti *Netwrok Address Translation* (NAT), *firewall*, dan konfigurasi *port forwarding* yang rumit. Dengan ngrok, pengembang dapat memperoleh URL publik yang dapat diakses secara global untuk layanan

lokal mereka hanya dengan satu perintah di terminal, tanpa perlu mengubah konfigurasi jaringan apa pun [45].

Arsitektur ngrok memiliki keunikan yang membedakannya dari *reversi proxy* tradisional. Alih-alih mengkonfigurasi server public untuk meneruskan lalu lintas ke alamat IP tujuan yang statis, ngrok mengadopsi model *agent-cloud*. Model ini terdiri dari dua komponen utama:

1. Ngrok Agent: Sebuah program klien yang diunduh dan dijalankan oleh pengguna di mesin lokal mereka, berdampingan dengan layanan yang ingin diekspos (misalnya, dalam penelitian ini adalah Streamlit).
2. Ngrok Cloud Service: Jaringan server global yang dikelola oleh ngrok, yang berfungsi sebagai titik akhir publik.

Mekanisme kerjanya dimulai ketika ngrok agent dieksekusi. Agent tersebut akan memulai koneksi *outbound* yang aman dan persisten menggunakan *Transport Layer Security* (TLS) ke ngrok *cloud service*. Koneksi ini menciptakan terowongan yang aman. Ketika ngrok *cloud service* menerima permintaan HTTP atau TCP pada URL publik yang telah dialokasikan, ia tidak mencoba untuk terhubung langsung ke alamat IP mesin lokal. Sebaliknya, ia meneruskan (*relays*) lalu lintas tersebut melalui terowongan *outbound* yang sudah ada ke ngrok agent. Akhirnya, *agent* meneruskan lalu lintas tersebut ke port layanan lokal yang telah ditentukan (misalnya, port 8501 untuk Streamlit) [46].

Arsitektur yang mengutamakan koneksi *outbound* ini adalah kunci mengapa ngrok sangat efektif dan menjadi solusi yang sangat diperlukan untuk lingkungan pengembangan *cloud* modern. Platform seperti Google Colab menawarkan sumber daya komputasi (GPU/TPU) yang kuat dan seringkali gratis, menjadikannya sangat menarik untuk penelitian AI yang intensif secara komputasi. Namun, platform ini pada dasarnya adalah lingkungan komputasi yang terisolasi dan sementara. Sebuah *notebook* Colab tidak memiliki alamat IP publik yang stabil dan tidak dirancang untuk menerima koneksi *inbound* dari internet karena alasan keamanan. Ini menciptakan "masalah isolasi" di mana aplikasi web yang berjalan di dalamnya tidak dapat diakses dari luar. Ngrok secara elegan memecahkan masalah ini. Karena *agent* ngrok di dalam Colab yang *memulai* koneksi ke luar, ia tidak melanggar

kebijakan keamanan jaringan Colab. *Ngrok cloud service* kemudian bertindak sebagai perantara publik yang stabil, secara efektif mengubah lingkungan komputasi yang terisolasi dan sementara seperti Google Colab menjadi server pengembangan yang dapat diakses secara publik untuk tujuan demonstrasi dan pengujian [47]. Tanpa teknologi jembatan (*bridging technology*) seperti ngrok, manfaat menggunakan GPU Colab untuk menjalankan aplikasi Streamlit interaktif akan terbatas hanya pada sesi *browser* pengembang itu sendiri.

2.17 Penelitian Terkait

Terdapat beberapa penelitian terkait yang dijadikan sebagai perbandingan serta rujukan mengenai metode serta hasil yang dicapai pada penelitian ini. Berikut merupakan ulasan dari beberapa penelitian terkait:

2.17.1 *Pengembangan Sistem Manajemen Notulensi dan Dokumentasi Rapat Berbasis Web (Studi Kasus: Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya)*

Penelitian yang dilakukan oleh Ilyas Abdi Nugraha., Fajar Pradana, dan Achmad Arwan. Dalam penelitian yang berjudul "*Pengembangan Sistem Manajemen Notulensi dan Dokumentasi Rapat Berbasis Web (Studi Kasus: Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya)*" telah menghasilkan sebuah sistem manajemen notulensi dan dokumentasi rapat dimana memiliki fungsi utama, membuat agenda rapat, membuat notula rapat, membuat dokumentasi rapat, membuat dokumentasi *video* rapat, dan membuat notula usai rapat. Tujuan dari perancangan sistem manajemen ini adalah untuk memberikan solusi terkait permasalahan yang dihadapi oleh peserta rapat, yaitu menciptakan sistem manajemen notulensi dan dokumentasi rapat berbasis *web* sehingga dapat mempermudah proses notulensi dan dokumentasi ketika rapat berlangsung [48].

Penelitian yang dilakukan oleh Ilyas Abdi Nugraha., Fajar Pradana, dan Achmad Arwan. Dalam penelitian yang berjudul "*Pengembangan Sistem Manajemen Notulensi dan Dokumentasi Rapat Berbasis Web (Studi Kasus: Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya)*". Tujuan dari

penelitian ini adalah untuk mempermudah proses notulensi dan dokumentasi pada rapat. Penelitian yang akan dilakukan mengenai "Rancang bangun *pipeline* notulensi otomatis dengan integrasi model Whisper (*speech-to-text*) dan model Pegasus (*abstractive summarization*)", akan meningkatkan hal-hal yang perlu ditingkatkan pada penelitian terkait ini.

2.17.2 *Meeting Assistant System Berbasis Teknologi Speech-to-Text*

Penelitian yang dilakukan oleh Daniel Soesanto., Budi Hartanto, Melisa. Dalam penelitian yang berjudul "*Meeting Assistant System Berbasis Teknologi Speech-to-Text*" berhasil menerapkan algoritma *TextRank* untuk mengambil 50% dari total kalimat pembahasan yang memiliki nilai *similarity* antar kalimatnya dengan peringkat tertinggi pada setiap agenda rapat. Tujuan dari pengembangan sistem ini adalah untuk mempermudah proses pembuatan notulen dengan melakukan ekstraksi kata-kata penting dari suatu rapat [49].

Penelitian yang dilakukan oleh Daniel Soesanto., Budi Hartanto, Melisa. Dalam penelitian yang berjudul "*Meeting Assistant System Berbasis Teknologi Speech-to-Text*" sistem yang dirancang ini masih bisa ditingkatkan pada bagian fitur, dengan meningkatkan fitur perekaman suara otomatis pada perancangan sistem meeting assistant. Hal inilah yang kemudian akan ditambahkan dalam penelitian yang akan dilakukan, dengan menambahkan fitur perekaman suara otomatis dimaksudkan untuk memudahkan proses perekaman suara pada proses notulensi.

2.17.3 *Optimalisasi Hasil Rapat Melalui Aplikasi E-NOT*

Penelitian yang dilakukan oleh Sunardi, Hersatoto Listiyono, dan Yunus Anis dalam jurnal berjudul "*Optimalisasi Hasil Rapat Melalui Aplikasi E-Not*" berhasil mengembangkan aplikasi *e-not* untuk mempermudah pencatatan dan monitoring hasil rapat. Aplikasi ini dilengkapi dengan fitur pencatatan agenda, notulensi, presensi, serta monitoring hasil keputusan rapat, yang membantu pimpinan dalam mengevaluasi pelaksanaan keputusan rapat secara real-time. Sistem ini dibangun menggunakan metode waterfall dan diuji dengan metode black box, menghasilkan validasi fungsional dengan tingkat keakuratan 100% [50].

Penelitian yang dilakukan oleh Sunardi, Hersatoto Listiyono, dan Yunus Anis dalam jurnal berjudul "*Optimalisasi Hasil Rapat Melalui Aplikasi E-Not*" sistem yang dirancang ini masih bisa ditingkatkan pada fitur notulensi yang masih dilakukan dengan *input* data manual. Hal inilah yang kemudian akan ditambahkan dalam penelitian yang akan dilakukan, dengan menambahkan fitur notulensi otomatis dimaksudkan untuk memudahkan proses notulensi.

2.17.4 *Rancang Bangun Manajemen Pertemuan Tingkat Eselon 1 di Kementerian Kesehatan Berbasi Web*

Penelitian yang dilakukan oleh Dodi Angga Kusuma dan Kemal Nazaruddin Siregar dalam jurnal berjudul "*Rancang Bangun Manajemen Pertemuan Tingkat Eselon I di Kementerian Kesehatan Berbasis Web*" berhasil merancang sistem informasi manajemen pertemuan berbasis *web* untuk memfasilitasi pengelolaan rapat daring dan luring di Kementerian Kesehatan. Sistem ini mengintegrasikan proses pemesanan ruang rapat, pengiriman undangan, pembuatan notula, serta persetujuan dan distribusi notula secara otomatis. Dengan menggunakan metode System Development Life Cycle (SDLC) dan pendekatan prototipe, sistem ini dirancang untuk meningkatkan efisiensi dan efektivitas pengelolaan rapat serta mengurangi biaya operasional terkait penggunaan lisensi aplikasi rapat daring [51].

Penelitian yang dilakukan oleh Dodi Angga Kusuma dan Kemal Nazaruddin Siregar dalam jurnal berjudul "*Rancang Bangun Manajemen Pertemuan Tingkat Eselon I di Kementerian Kesehatan Berbasis Web*" sistem yang dirancang ini masih bisa ditingkatkan pula pada fitur notula rapat. Pada sistem ini pencatatan notula rapat atau notulensi masih dilakukan secara manual. Hal inilah yang kemudian akan ditambahkan dalam penelitian yang akan dilakukan, dengan menambahkan fitur notulensi otomatis yang dimaksudkan untuk memudahkan.

III. METODE PENELITIAN

3.1 Waktu dan Tempat Penelitian

Waktu dan tempat pelaksanaan penelitian dilakukan pada:

1. Waktu penelitian : Mei 2024 sampai dengan September 2024
2. Tempat penelitian : Universitas Lampung

3.2 Waktu Penelitian

Jadwal yang direncanakan pada penelitian ini adalah sebagai berikut:

Tabel 3.1 Jadwal Penelitian

No	Aktivitas	Maret 2025	April 2025	Mei 2025	Juni 2025	Juli 2025
1	Studi Literatur					
2	Pemilihan Komponen dan Teknologi Pipeline					
3	Pengumpulan Data					
4	Preprocessing Data					
5	Desain Arsitektur Pipeline					
5	Implementasi Pipeline					
6	Pengujian					
7	Penyusunan Laporan					

3.3 Alat dan Bahan Dalam Penelitian

3.3.1 Alat Penelitian

Alat- alat yang digunakan dalam penelitian ini adalah sebagai berikut:

Tabel 3.2 Alat yang digunakan dalam penelitian

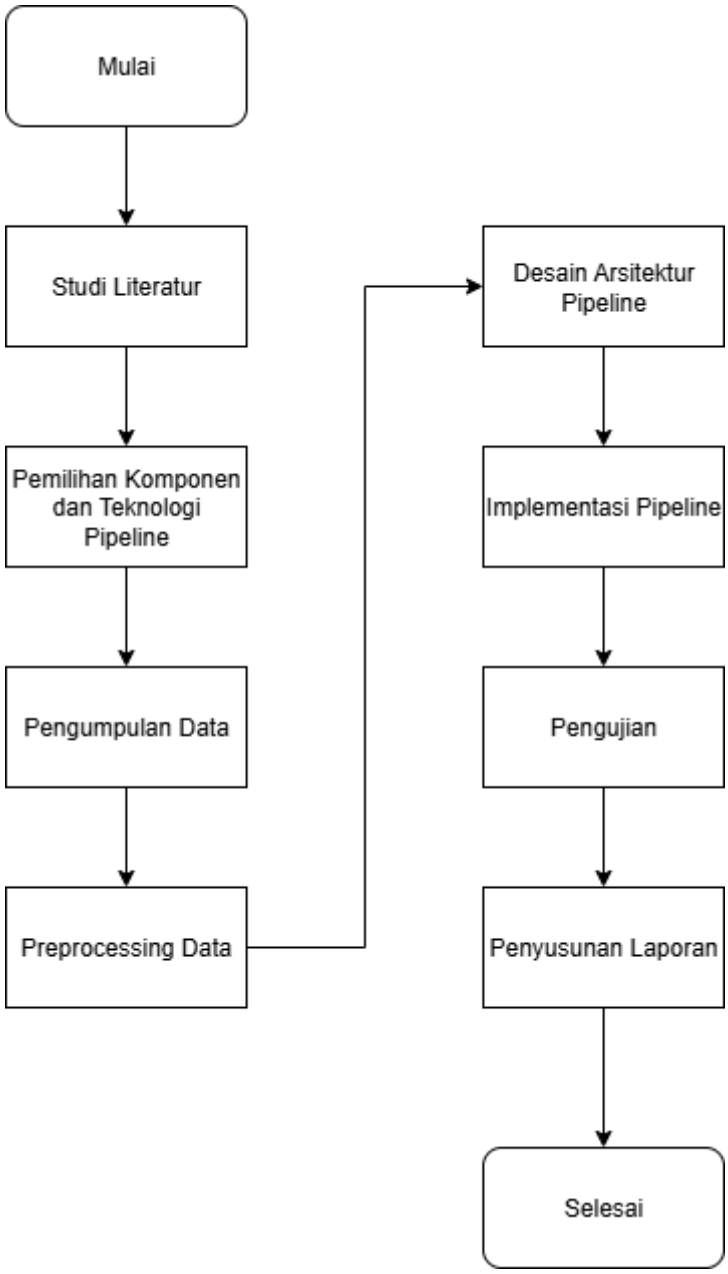
No	Nama Alat	Spesifikasi	Deskripsi
1	Laptop	AMD Ryzen 5 4500u, RAM 12GB, Sistem Operasi Windows 11 64-bit	Perangkat keras yang digunakan untuk membuat sistem notulensi otomatis
2	Google Colab	Versi Google Colab Pro	Layanan Jupyter Notebook yang dihosting dan tidak memerlukan pengaturan untuk digunakan dan menyediakan akses ke sumber daya komputasi, termasuk GPU dan TPU.

3.3.2 Bahan Penelitian

Bahan yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Jurnal nasional dan internasional yang digunakan sebagai sumber referensi dalam melakukan penelitian.
2. Beberapa penelitian terdahulu yang relevan.

3.4 Tahapan Penelitian



Gambar 2. Tahapan Penelitian

Gambar 2 merupakan tahapan penelitian yang akan dilaksanakan dalam penelitian ini. Tahapan tersebut terdiri dari Studi Literatur, Pemilihan Jenis AI, Mempelajari Detail Jenis AI yang Dipilih, Pengumpulan Data, Pemrograman Model, Pengujian, dan Penyusunan Laporan.

3.4.1 Studi Literatur

Pada tahap ini akan dilakukan studi literatur yang mendalam tentang notulensi otomatis, teknologi yang terkait, dan kerangka kerja yang relevan. Tahap ini berguna sebagai dasar penelitian. Dilakukan pencarian dan memakai beberapa teori yang relevan dengan pembahasan tentang penelitian ini. Dasaran teori dimanfaatkan pada penelitian melalui sekumpulan jurnal, artikel, buku, dan beberapa penelitian yang sudah dilakukan sebelumnya sebagai panduan dari penelitian ini.

3.4.2 Pemilihan Komponen dan Teknologi *Pipeline*

Pada tahap ini akan dilakukan proses pemilihan model yang akan digunakan di dalam *pipeline.* Model yang dipilih ada tiga, yaitu model Whisper, Pegasus, dan GPT. Model Whisper dipilih sebagai komponen transkripsi (*speech-to*-text) karena kemampuannya yang teruji dalam menyalin *file audio* dengan tingkat akurasi yang tinggi. Model ini memiliki ketahanan yang baik terhadap variasi aksen, kebisingan latar belakang, dan bahasa teknis. Model Pegasus dipilih sebagai komponen peringkasan (*abstractive summarization*) karena memiliki kinerja yang baik dalam menghasilkan ringkasan yang informatif dan relevan. Kemampuan abstraktifnya memungkinkan model untuk menciptakan kalimat baru yang lebih ringkas namun tetap mempertahankan makna inti dari teks asli, bukan sekedar mengekstrak kalimat yang sudah ada. Model GPT dipilih sebagai komponen penyempurnaan akhir karena kemampuannya dalam melakukan parafrase dan penataan ulang teks. Model ini digunakan untuk meningkatkan keterbacaan, kejelasan, dan profesionalitas dari ringkasan yang dihasilkan oleh model Pegasus, sehingga output akhir notulensi memiliki gaya bahasa yang natural dan terstruktur.

3.4.3 Pengumpulan Data

Tahap ini mencakup proses pengumpulan data dari dua sumber utama. Pertama, data didapatkan dari pihak yang sering menyelenggarakan rapat, perkuliahan dan seminar. Kedua, data diperoleh melalui rekaman rapat, perkuliahan dan seminar

dari platform digital seperti Youtube. Data audio yang dikumpulkan, akan diolah di dalam model notulensi otomatis.

Tabel 3.3 Pengumpulan Dataset

No	Sumber Data	Jumlah File	Total Durasi (Jam)	Label
1.	Rapat (Zoom, Google Meet)	25	25	Bahasa Indonesia
2.	Perkuliahan (Zoom, Google Meet, Youtube)	25	25	Bahasa Indonesia
3.	Seminar (Zoom, Google Meet, Youtube)	25	25	Bahasa Indonesia
4.	Podcast (Youtube)	25	25	Bahasa Indonesia
Total		**100**	**100**	**Bahasa Indonesia**

3.4.4 *Preprocessing* Data

Pada tahap ini, *dataset* yang telah dikumpulkan pada proses pengumpulan data akan melalui proses *preprocessing* data. Tahap ini merupakan langkah yang krusial dalam alur tahapan penelitian, karena kualitas data input secara langsung mempengaruhi performa dan akurasi model Pegasus dan model Whisper yang digunakan. Tujuan utama dari *preprocessing audio* adalah untuk membersihkan, menstandarisasi, dan mengubah format *audio* mentah agar siap dan optimal untuk dianalisis oleh model Whisper. Proses *preprocessing* data *audio* dalam penelitian ini mencakup beberapa langkah teknis yang sistematis, yaitu:

1. Konversi Format dan Standarisasi *Sample Rate*

   Langkah pertama adalah memastikan semua *file audio* dalam *dataset* memiliki format dan karakteristik yang seragam. Seluruh *file* audio, terlepas dari format aslinya (misalnya mp3, m4a, atau lainnya), akan dikonversi menjadi format WAV (*Waveform Audio File Format*). Format WAV dipilih karena bersifat *lossless* dan menjadi standar dalam banyak aplikasi pemrosesan *audio*. Secara bersamaan, *sample rate* dari setiap *file audio* akan distandarisasi menjadi 16.000

Hz (16 kHz). Penyeragaman ini wajib dilakukan karena model Whisper secara spesifik dilatih dan dioptimalkan untuk memproses audio dengan *sample rate* tersebut, sehingga memastikan kompatibilitas dan performa transkripsi yang maksimal.

2. Pembersihan *Noise* (*Noise Reduction*)

   Dataset audio yang berasal dari rekaman rapat, perkuliahan, seminar, dan podcast seringkali mengandung *noise* atau kebisingan latar belakang, seperti suara pendingin ruangan, gema, atau percakapan lain yang tidak relevan. *Noise* ini dapat mengganggu proses transkripsi dan menurunkan akurasi model. Karena itu, teknik reduksi *noise* diterapkan untuk menyaring dan mengurangi suara-suara yang tidak diinginkan, sehingga model dapat lebih fokus terhadap sinyal suara utama (ucapan manusia).

3. Penyesuaian Volume (Normalisasi Amplitudo)

   Koleksi *dataset audio* dari berbagai sumber dan sesi rekaman yang berbeda seringkali memiliki tingkat volume yang tidak konsisten. Ada rekaman yang suaranya sangat keras, dan ada pula yang sangat pelan. Untuk mengatasi permasalahan ini, dilakukanlah normalisasi amplitudo pada setiap *file audio*. Proses ini menyesuaikan dan menyeragamkan tingkat *volume* di seluruh *dataset*, sehingga tidak ada data yang terlalu dominan atau terlalu lemah hanya karena terdapat perbedaan *volume*. Hal ini membantu model untuk bisa memproses semua data secara stabil.

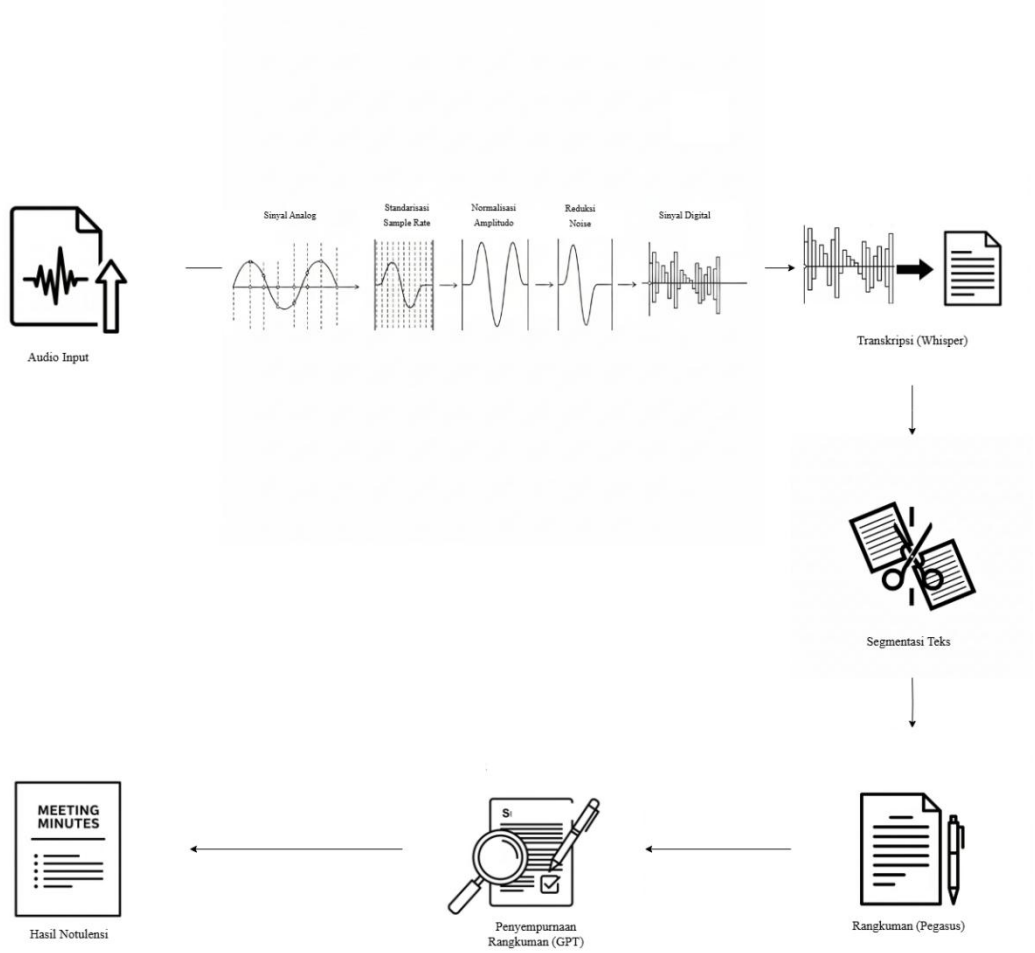
4. Transformasi ke *Log-Mel Spectogram*

   Tahap terakhir dari proses *preprocessing audio* sebelum dimasukkan ke dalam model Whisper adalah mengubah sinyal *audio* menjadi representasi *visual* dalam bentuk *log-Mel spectrogram*. Model Whisper, yang berbasis arsitektur Transformer tidak memproses gelombang *audio* mentah secara langsung. Sebaliknya, sinyal *audio* dibagi menjadi segmen-segmen pendek berdurasi 30 detik, yang kemudian diubah menjadi *spectrogram*, representasi frekuensi suara terhadap waktu. *Spectogram* ini selanjutnya dipetakan ke skala Mel yang meniru persepsi pendengaran manusia, dan nilainya diubah ke skala logaritmik. Hasil akhir yang didapatkan berupa *log-Mel spectrogram* inilah yang menjadi input sebenarnya bagi encoder model Whisper untuk dianalisis dan ditranskripsikan.

3.4.5 Desain Arsitektur *Pipeline*

Pada tahap ini, akan dilakukan perancangan arsitektur *pipeline* sistem notulensi otomatis secara menyeluruh. Desain ini akan memetakan alur kerja sistem dari awal hingga akhir, mulai dari *input* pengguna hingga *output* notulensi, serta menentukan komponen teknologi yang akan digunakan untuk membangun sistem tersebut.

Arsitektur *pipeline* akan dirancang untuk mengintegrasikan beberapa komponen utama secara sekuensial, dikenal sebagai pendekatan *cascade*. Pendekatan ini umum digunakan untuk tugas peringkasan *speech-to-text* dengan mengintegrasikan model pengucapan (ASR) yang diikuti oleh model peringkasan teks (T2T) [6]. Infrastruktur komputasi pada penelitian ini akan memanfaatkan Google Colab yang menyediakan sumber daya GPU untuk mempercepat proses komputasi model-model *deep learning* yang besar seperti model Whisper dan Pegasus. Untuk antarmuka pengguna (UI), akan dikembangkan sebuah aplikasi web interaktif menggunakan *framework* Streamlit, yang memungkinkan pengguna untuk mengunggah *file audio* dengan mudah. Agar aplikasi yang berjalan di Google Colab dapat diakses secara publik, sistem akan menggunakan *tunnelling* Ngrok.



Gambar 3. Alur *Pipeline* Sistem Notulensi Otomatis

Seperti yang terlihat pada gambar 3, alur data pada sistem dirancang sebagai sebuah *pipeline* sekuensial yang dimulai saat pengguna mengunggah *file* audio. Seperti yang terlihat pada gambar 3, proses ini dapat diuraikan sebagai berikut:

1. ***Audio Input*:** Pengguna mengunggah *file audio* melalui antarmuka Streamlit.
2. ***Preprocessing Audio*:** Konversi *audio* ke format WAV 16kHz dan pembersihan *noise*.
3. **Transkripsi:** *Audio* akan diubah menjadi teks menggunakan model Whisper.
4. **Segmentasi Teks:** Teks transkripsi yang panjang akan dipecah menjadi segmen-segmen yang lebih kecil agar sesuai dengan batasan input model Pegasus.
5. **Rangkuman:** Setiap segmen teks akan diringkas secara abstraktif menggunakan model Pegasus.

6. **Penyempurnaan Rangkuman:** Hasil ringkasan dari model Pegasus akan disempurnakan dan distrukturkan ulang menggunakan model GPT untuk meningkatkan keterbacaan dan profesionalitasnya.
7. **Hasil Notulensi:** *Output* akhir dari *pipeline* ini adalah hasil notulensi yang terstruktur dan siap digunakan.

Hasil notulensi kemudian akan dikirim kembali dan ditampilkan kepada pengguna melalui antarmuka Streamlit. Desain arsitektur *pipeline* sistem ini memastikan bahwa setiap model (Whisper, Pegasus, dan GPT) terintegrasi secara berurutan dan otomatis untuk menghasilkan notulensi yang akurat dan terstruktur dari *audio input*.

3.4.6 Implementasi *Pipeline*

Tahap implementasi ini merupakan realisasi teknis dari arsitektur *pipeline* yang telah dirancang sebelumnya. Pada tahap ini, alur kerja yang telah dibuat diwujudkan dalam bentuk kode program Python yang berjalan di lingkungan Google Colab. Implementasi untuk setiap tahapan sebagai berikut:

1. ***Audio Input*:** Pengguna mengunggah file audio melalui antarmuka sistem. Implementasi tahap ini menggunakan *widget st.file_uploader* dari *framework* Streamlit yang menerima *file audio* dari perangkat lokal pengguna dan menyiapkannya untuk diproses oleh *backend*.
2. ***Preprocessing Audio*:** Tahap ini diimplementasikan dengan memanfaatkan *library librosa* untuk melakukan konversi format *audio* ke WAV, standarisasi *sample rate* menjadi 16 kHz, dan normalisasi amplitudo. Untuk pembersihan noise, digunakan *library noisereduce* untuk meningkatkan kualitas sinyal *audio* sebelum dimasukkan ke model transkripsi.
3. **Transkripsi:** Transkripsi *audio* diimplementasikan dengan memanfaat *library faster-whisper*, yang merupakan versi optimasi dari model Whisper. Penggunaan *library* ini memungkinkan proses inferensi yang lebih cepat dan efisien pada GPU yang tersedia di lingkungan Google Colab.
4. **Segmentasi Teks:** Fungsi kustom di program untuk memecah teks transkripsi yang panjang menjadi segmen-segmen yang saling tumpang tindih (*overlapping chunks*), memastikan setiap segmen sesuai dengan batas token model Pegasus.

5. **Rangkuman:** Untuk mengimplementasikan peringkasan secara efisien, segmen-segmen teks diproses oleh model Pegasus dalam bentuk *batch*. Pemrosesan secara *batch* ini bertujuan untuk mengoptimalkan penggunaan memori GPU dan mempercepat waktu komputasi saat menghasilkan ringkasan dari setiap segmen.
6. **Penyempurnaan Rangkuman:** Hasil ringkasan dari model Pegasus akan digabungkan dan dikirim ke model GPT melalui pemanggilan API. Sebuah *prompt* yang telah dirancang secara spesifik akan digunakan untuk menginstruksikan model GPT agar melakukan parafrase dan menata ulang teks menjadi format notulensi yang lebih profesional dan terstruktur.
7. **Hasil Notulensi:** Teks *final* yang diterima dari model GPT akan disimpan sebagai *final output*. *Output* ini kemudian ditampilkan kembali kepada pengguna melalui antarmuka Streamlit dan disediakan fungsionalitas untuk mengunduhnya dalam format dokumen.

3.4.7 Pengujian

Tahap pengujian dilakukan untuk mengevaluasi kinerja *pipeline* notulensi otomatis secara keseluruhan. Pengujian dilakukan untuk mengukur efektivitas sistem sebagai satu kesatuan dalam menghasilkan notulensi yang berkualitas, bukan untuk menilai performa masing-masing model secara terpisah. Evaluasi ini dilakukan dengan berbagai skenario *audio* untuk menilai apakah *output* akhir dari *pipeline* telah sesuai dengan kriteria yang diharapkan. Pengujian dilakukan dengan dua metode pengujian, yaitu:

1. ROUGE (*Recall-Oriented Understudy for Gisting Evaluation*): Metode ini digunakan untuk pengujian kuantitatif dengan membandingkan ringkasan yang dihasilkan oleh pipeline (*output* dari tahap Pegasus) terhadap transkripsi lengkap (*output* dari tahap Whisper) sebagai teks referensi. ROUGE akan memberikan skor numerik yang objektif untuk mengukur sejauh mana kesamaan konten, tumpang tindih kata, dan urutan kalimat antara output model dan referensi. Skor ini menjadi indikator keakuratan dan relevansi notulensi yang dihasilkan sistem.

2. *Likert Scale*: Metode ini digunakan untuk pengujian kualitatif dengan melibatkan responden manusia untuk menilai kualitas transkripsi dan notulensi berdasarkan beberapa kriteria, seperti keterbacaan, koherensi, dan kelengkapan informasi. Responden akan memberikan penilaian dalam skala (misalnya, 1 hingga 5, dari "sangat tidak setuju" hingga "sangat setuju"). Umpan balik subjektif ini sangat penting untuk memahami persepsi pengguna dan mengidentifikasi aspek-aspek yang mungkin tidak tertangkap oleh metrik otomatis, seperti alur logika dan kemudahan pemahaman.

Kombinasi dari skor ROUGE yang objektif dan penilaian *Likert Scale* yang subjektif akan memberikan gambaran lengkap mengenai kekuatan dan kelemahan sistem. Hasil dari kedua pengujian ini akan menjadi dasar untuk menentukan apakah sistem telah memenuhi standar yang diharapkan atau memerlukan optimasi lebih lanjut.

3.4.9 Penyusunan Laporan

Tahap akhir dari penelitian ini adalah pelaporan hasil dan temuan penelitian mengenai Rancang Bangun *Pipeline* Notulensi Otomatis Dengan Integrasi Model Whisper (*Speech-to-Text*) dan model Pegasus (A*bstractive Summarization*). Dari data yang dihasilkan dan telah dianalisis kemudian dilakukan pengambilan kesimpulan dan saran. Hasil temuan yang ada kemudian digunakan sebagai skripsi pada Universitas Lampung.

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari penelitian yang dilakukan berupa penelitian rancang bangun *pipeline* notulensi otomatis dengan integrasi model Whisper (*speech-to-text)* dan model Pegasus (*abstractive summarization*) didapatkan kesimpulan sebagai berikut:

1. Telah berhasil dibangun sebuah arsitektur *pipeline* untuk notulensi otomatis dengan mengintegrasikan tiga model *deep learning*. *Pipeline* ini memetakan alur kerja sekuensial yang dimulai dari *preprocessing audio*, dilanjutkan dengan transkripsi menggunakan model Whisper, segmentasi teks, peringkasan menggunakan model Pegasus, dan diakhiri dengan penyempurnaan oleh model GPT. Berdasarkan pengujian, *pipeline* sistem ini mencapai skor rata-rata ROUGE-1 sebesar 0,5103, ROUGE-2 sebesar 0,4575, dan ROUGE-L sebesar 0,4743, serta memperoleh skor rata-rata 4.30 dari skala 5 pada evaluasi *Likert Scale*.
2. Penentuan parameter optimal untuk model Whisper dalam aristektur *pipeline* menunjukkan bahwa penggunaan *beam_size=10* menghasilkan transkripsi dengan kualitas terbaik. Berdasarkan hasil pengujian, parameter ini secara konsisten memberikan peningkatan nilai *recall* dan mencapai skor rata-rata F1-Score tertinggi pada metrik ROUGE-1 (0,5362), ROUGE-2 (0,4856), dan ROUGE-L (0,5003). Pengaturan ini menawarkan kompromi terbaik antara kedalaman pencarian dan beban komputasi, menghasilkan transkripsi yang akurat dan detail.
3. Strategi segmentasi teks dan parameter model Pegasus yang optimal untuk meningkatkan kualitas hasil notulensi adalah dengan menggunakan segmen berukuran 1024 token (Parameter Set 3). Konfigurasi ini terbukti mampu menghasilkan notulensi yang paling detail, dan komprehensif, serta mencapai

skor F1-Score ROUGE tertinggi (rata-rata ROUGE-1 sebesar 0,5135). Dengan mengutamakan kualitas dan kelengkapan informasi, parameter 1024 token dipilih sebagai strategi yang paling optimal.

4. Evaluasi performa dan kualitas *pipeline* notulensi otomatis menunjukkan adanya peningkatan yang signifikan pada aspek efisiensi, akurasi, dan keterbacaan. Peningkatan efisiensi dibuktikan melalui evaluasi kualitatif, dimana pengguna memberikan skor kepuasan keseluruhan yang sangat tinggi (rata-rata 4,58 dari 5) dan menilai notulensi yang dihasilkan dapat langsung digunakan sebagai dokumen resmi dengan sedikit suntingan (rata-rata 4,07 dari 5). Dari segi akurasi, sistem divalidasi secara kuantitatif dengan skor ROUGE yang memuaskan (rata-rata ROUGE-1 sebesar 0,5103) dan didukung persepsi pengguna yang menilai notulensi mampu merepresentasikan inti pembahasan secara akurat (rata-rata 4,54 dari 5). Sementara itu, keterbacaan notulensi juga dinilai sangat baik, tercermin dari skor rata-rata yang tinggi untuk aspek struktur yang logis (rata-rata 4,49 dari 5), kejelasan kalimat (rata-rata 4,43 dari 5), dan penggunaan bahasa formal yang konsisten (rata-rata 4,47 dari 5)

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, terdapat beberapa peluang pengembangan yang dapat dilakukan pada penelitian selanjutnya. Adapun saran yang dapat diberikan adalah sebagai berikut:

1. Integrasi Modul *Speaker Diarization* ke dalam *Pipeline*: Arsitektur *pipeline* saat ini belum dapat mengidentifikasi pembicara dalam rekaman. Pengembangan selanjutnya disarankan untuk mengintegrasikan modul *speaker diarization* sebagai salah satu tahapan baru di dalam *pipeline*. Penambahan komponen ini akan secara signifikan meningkatkan nilai guna notulensi untuk keperluan rapat formal dan pembuatan risalah yang membutuhkan atribusi pernyataan kepada individu yang bersangkutan.
2. Optimasi Kinerja Komponen *Pipeline* melalui *Fine-Tuning*: *Pipeline* yang dikembangkan dalam penelitian ini menggunakan model pra-terlatih (*pre-trained*) yang bersifat umum. Untuk meningkatkan akurasi pada konteks yang

sangat teknis atau spesifik (misalnya, rapat medis, persidangan hukum, atau diskusi ilmiah), disarankan untuk melakukan *fine-tuning* pada model Whisper dan Pegasus menggunakan *dataset* khusus dari domain tersebut. Langkah ini berpotensi meningkatkan kemampuan model dalam mengenali terminologi dan jargon yang unik.

DAFTAR PUSTAKA

[1] A. Dailey-Hebert, *MAXIMIZING INTERACTIVITY IN ONLINE LEARNING: MOVING BEYOND DISCUSSION BOARDS*. Journal of Educators Online, 2018.

[2] T. Bates, *Teaching in a Digital Age Guidelines for designing teaching and learning*. BCcampus, 2015.

[3] W. A. Green and H. Lazarus, "Are Today's Executives Meeting with Success?," 1991.

[4] D. Soesanto, B. Hartanto, and Melisa, "Meeting Assistant System Berbasis Teknologi Speech-to-Text," *Teknika*, vol. 10, no. 1, pp. 1–7, Jan. 2021, doi: 10.34148/teknika.v10i1.307.

[5] "Google SRE - Managing Data Processing Pipelines: Challenges." Accessed: Jul. 29, 2025. [Online]. Available: https://sre.google/sre-book/data-processing-pipelines/

[6] R. Monteiro and D. Pernes, "Towards End-to-end Speech-to-text Summarization," Jun. 2023, [Online]. Available: http://arxiv.org/abs/2306.05432

[7] "Pipeline." Accessed: Jul. 29, 2025. [Online]. Available: https://huggingface.co/docs/transformers/pipeline_tutorial

[8] E. Horvitz and D. Mulligan, "Data, privacy, and the greater good," *Science (1979)*, vol. 349, no. 6245, pp. 253–255, Jul. 2015, doi: 10.1126/science.aac4520.

[9] M. J. D Powell *et al.*, “The gradient projection method for nonlinear programming, pt. I, linear constraints,” 1968.

[10] S. M. Lavalle, “PLANNING ALGORITHMS.” [Online]. Available: http://planning.cs.uiuc.edu/

[11] C. Urmson *et al.*, “Autonomous driving in urban environments: Boss and the urban challenge,” *J Field Robot*, vol. 25, no. 8, pp. 425–466, Aug. 2008, doi: 10.1002/rob.20255.

[12] G. Laporte, F. V. Louveaux, and L. Van Hamme, “An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands,” *Oper Res*, vol. 50, no. 3, pp. 415–423, 2002, doi: 10.1287/opre.50.3.415.7751.

[13] K. P. . Murphy, *Machine learning : a probabilistic perspective*. MIT Press, 2012.

[14] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” May 27, 2015, *Nature Publishing Group*. doi: 10.1038/nature14539.

[15] J. Han, M. Kamber, and J. Pei, *Data Mining Third Edition*. Morgan Kaufmann, 2012.

[16] C. M. . Bishop, *Pattern recognition and machine learning*. Springer Science + Business Media, 2009.

[17] S. J. Russell and P. Norvig, *Artificial Intelligence A Modern Approach Fourth Edition*. Pearson, 2020.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[19] X. Zhu and A. B. Goldberg, *Introduction to Semi-Supervised Learning Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers, 2009.

[20] “Self-supervised learning: The dark matter of intelligence.” Accessed: Sep. 30, 2024. [Online]. Available: https://ai.meta.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/

[21] R. S. Sutton and A. G. Barto, *Answers to Exercises for Reinforcement Learning: An Introduction 2nd Edition*. MIT Press, 2018.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[23] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int J Comput Vis*, vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/s11263-015-0816-y.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun ACM*, vol. 60, no. 6, pp. 84–90, Jun. 2017, doi: 10.1145/3065386.

[25] A. Graves, A. Mohamed, and G. Hinton, *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. 2013.

[26] Y. Goldberg, “A Primer on Neural Network Models for Natural Language Processing,” 2016.

[27] S. Hochreiter and J. ¨ Urgen Schmidhuber, *Long Short-Term Memory*, vol. 9. Neural Computation, 1997.

[28] J. Devlin, M.-W. Chang, K. Lee, K. T. Google, and A. I. Language, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” [Online]. Available: https://github.com/tensorflow/tensor2tensor

[29] I. Kadek Suryadharma, G. Budiman, and B. Irawan, “PERANCANGAN APLIKASI SPEECH TO TEXT BAHASA INGGRIS KE BAHASA BALI MENGGUNAKAN POCKETSPHINX BERBASIS ANDROID (Design Application Speech to Text English to Balinese Language Using PocketSphinx Base On Android),” vol. 1, no. 1, p. 229, 2014.

[30] L. Deng and X. Li, “Machine learning paradigms for speech recognition: An overview,” *IEEE Trans Audio Speech Lang Process*, vol. 21, no. 5, pp. 1060–1089, 2013, doi: 10.1109/TASL.2013.2244083.

[31] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, “FASTTEXT.ZIP: COMPRESSING TEXT CLASSIFICATION MODELS.” [Online]. Available: https://github.com/facebookresearch/fastText

[32] R. Nallapati, B. Zhou, C. N. dos santos, C. Gulcehre, and B. Xiang, “Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond,” Feb. 2016, [Online]. Available: http://arxiv.org/abs/1602.06023

[33] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. Mcleavey, and I. Sutskever, “Robust Speech Recognition via Large-Scale Weak Supervision.” [Online]. Available: https://github.com/openai/

[34] S. Schneider, A. Baevski, R. Collobert, and M. Auli, “wav2vec: Unsupervised Pre-training for Speech Recognition,” Apr. 2019, [Online]. Available: http://arxiv.org/abs/1904.05862

[35] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, “PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization,” Dec. 2019, [Online]. Available: http://arxiv.org/abs/1912.08777

[36] A. R. Openai, K. N. Openai, T. S. Openai, and I. S. Openai, “Improving Language Understanding by Generative Pre-Training.” [Online]. Available: https://gluebenchmark.com/leaderboard

[37] R. Sheldon, “What is audio? – TechTarget Definition.” Accessed: Jul. 11, 2024. [Online]. Available: https://www.techtarget.com/whatis/definition/audio

[38] C.-Y. Lin, “ROUGE: A Package for Automatic Evaluation of Summaries.”

[39] A. Joshi, S. Kale, S. Chandel, and D. Pal, “Likert Scale: Explored and Explained,” *Br J Appl Sci Technol*, vol. 7, no. 4, pp. 396–403, Jan. 2015, doi: 10.9734/bjast/2015/14975.

[40] I. E. Allen and C. A. Seaman, “Likert Scales and Data Analyses.” [Online]. Available: www.sloan-c.org,

[41] “Getting Started with Streamlit for Machine Learning Deployment | by Alidu Abubakari | Medium.” Accessed: Jul. 08, 2025. [Online]. Available: https://medium.com/@alidu143/getting-started-with-streamlit-for-machine-learning-deployment-532e468567ce

[42] “Python Tutorial: Streamlit | DataCamp.” Accessed: Jul. 08, 2025. [Online]. Available: https://www.datacamp.com/tutorial/streamlit

[43] “A Straightforward Tutorial of Streamlit - viso.ai.” Accessed: Jul. 08, 2025. [Online]. Available: https://viso.ai/deep-learning/streamlit-tutorial/

[44] “What is ngrok? | ngrok documentation.” Accessed: Jul. 08, 2025. [Online]. Available: https://ngrok.com/docs/what-is-ngrok/

[45] “Ngrok Secrets: Instantly Share Your Local Host Safely 2025.” Accessed: Jul. 08, 2025. [Online]. Available: https://www.outrightcrm.com/blog/ngrok-securely-share-localhost/

[46] “How does ngrok work? | ngrok documentation.” Accessed: Jul. 08, 2025. [Online]. Available: https://ngrok.com/docs/how-ngrok-works/

[47] “Google Colab | ngrok documentation.” Accessed: Jul. 08, 2025. [Online]. Available: https://ngrok.com/docs/using-ngrok-with/googleColab/

[48] I. A. Nugraha, F. Pradana, and A. Arwan, “Pengembangan Sistem Manajemen Notulensi dan Dokumentasi Rapat Berbasis Web (Studi Kasus: Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya),” 2020. [Online]. Available: http://j-ptiik.ub.ac.id

[49] D. Soesanto, B. Hartanto, and Melisa, “Meeting Assistant System Berbasis Teknologi Speech-to-Text,” *Teknika*, vol. 10, no. 1, pp. 1–7, Jan. 2021, doi: 10.34148/teknika.v10i1.307.

[50] H. Listiyono and Y. Anis, “OPTIMALISASI HASIL RAPAT MELALUI APLIKASI E-NOT,” *Dinamika Informatika*, vol. 14, no. 2, pp. 67–77.

[51] D. A. Kusuma and K. N. Siregar, “RANCANG BANGUN MANAJEMEN PERTEMUAN TINGKAT ESELON I DI KEMENTERIAN KESEHATAN BERBASIS WEB,” *Jurnal Indonesia : Manajemen Informatika dan Komunikasi*, vol. 4, no. 1, pp. 97–109, Jan. 2023, doi: 10.35870/jimik.v4i1.124.