Enhancing Meeting Quality with NLP: From Speech-to-Text to Summarization and Analysis

[1]Yu-Jui, Chen (Y.J.C) *Interdisciplinary Program of Management and Technology*
National Tsing Hua University
Hsinchu, Taiwan
[2]Ko-Qin, Mei (K.Q.M) *Interdisciplinary Program of Electrical Engineering and Computer Science*
National Tsing Hua University
Hsinchu, Taiwan
[3]Dong-Ting, Yao (Y.D.T) *Computer Science*
National Tsing Hua University
Hsinchu, Taiwan
[4]Yu-Chieh, Chiu (C.Y.C) *Computer Science*
National Tsing Hua University
Hsinchu, Taiwan
[5]Ya-Yu, Chien (C.Y.Y) *Computer Science*
National Tsing Hua University
Hsinchu, Taiwan
[6]Ping-Yen, Li (L.P.Y) *Computer Science*
National Tsing Hua University
Hsinchu, Taiwan

***Abstract*—Inefficient meetings are a common problem in team work, often accompanied by the troublesome task of organizing meeting records. With advancements in natural language processing, tasks such as transcription, summarization, and specific analyses can now be effectively handled by AI-driven models. In this study, we developed a comprehensive system that integrates automatic speech recognition (ASR) with speaker diarization through WhisperX, along with large language model Gemini, to transcribe meeting audio into text and perform detailed analyses. By converting meeting records into text and summaries, our system eases the burden of organizing meeting content manually. Additionally, by providing insights about the team, such as character analysis of each member and the strengths and weaknesses of the entire working group, users can better understand how to improve team performance. This study carefully selects models for each component of the system, ultimately implementing WhisperX, OpenCC, and Gemini to complete the system.**

***Index Terms*—Natural Language Processing, Automatic Speech Recognition, Speaker Diarization, Summary, Large Language Model, Prompt Engineering, Speaking Repetitive Pattern Detection, Belbin Team Role.**

I. Introduction

Whether students or organizational staff, everyone must participate in various meetings as part of their daily routines. However, the content of these meetings is often underutilized. Manually producing transcripts is both time-consuming and labor-intensive, and these transcripts or meeting notes frequently offer limited value. It is evident that many meetings suffer from inefficiency, low engagement, and repetitive discussions, all of which negatively impact team productivity. Faced with these challenges, there is an urgent need for a new approach to improve meeting quality and efficiency.

Natural Language Processing (NLP) provides a promising solution. Technologies such as Automatic Speech Recognition (ASR), text summarization, and the use of GPT models for targeted analysis of meeting content allow for a deeper understanding and evaluation of discussions, offering an objective, structured, and effective mechanism to enhance meeting quality and efficiency.

In this project, we aim to explore and develop a system that integrates ASR, speaker diarization, and large language models. We investigate various methods to optimize the output of these models to better meet our specific needs.

II. Background

A. *Overview*

First, we convert the audio or video files into WAV format and input them into WhisperX, producing a verbatim transcript with speaker annotations. This transcript then undergoes a series of processes, including conversion from Simplified Chinese to Traditional Chinese, text correction, and cleaning. Afterward, the attendee names provided by the user are matched with the speaker annotations in the transcript, resulting in an optimized version that includes actual names. Next, the enhanced transcript is compared and analyzed against the meeting objectives using the large language model Gemini, generating a meeting summary, a role analysis based on the Belbin Team Roles, and a repetitive pattern evaluation report.

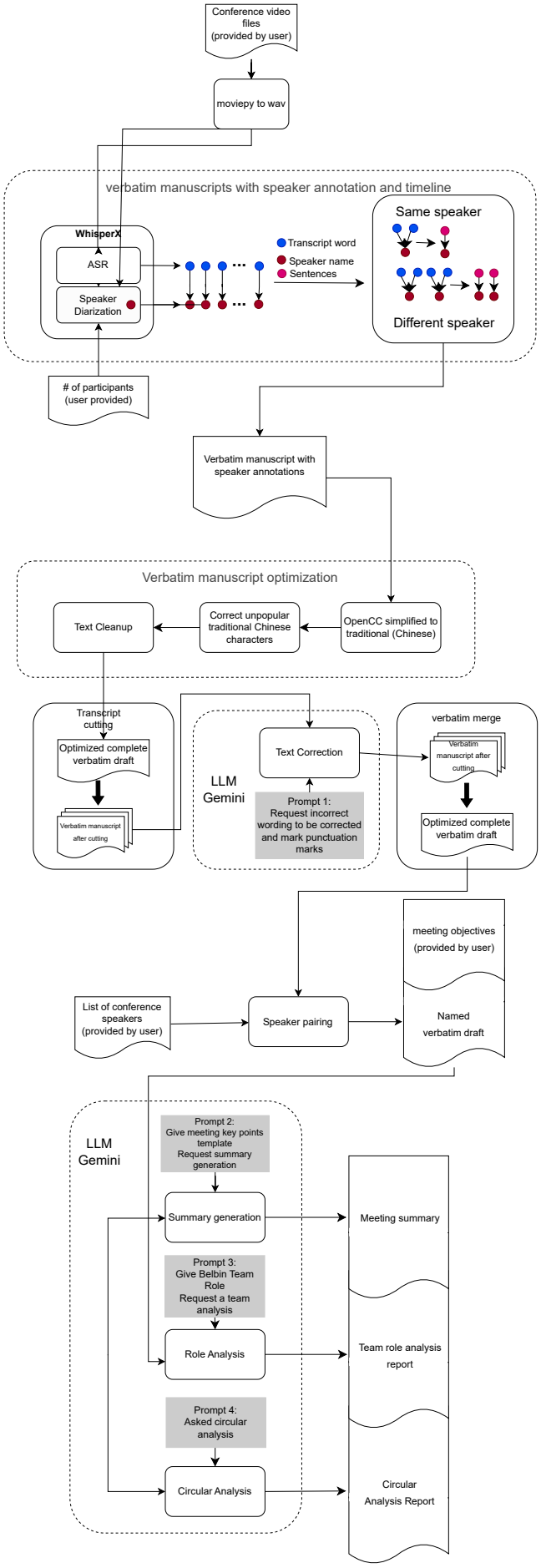

Fig. 1. Overall framework of system

B. ASR

We input the files into ASR model and Speaker Diarization model. On one hand, the files are transcribed into a verbatim transcript with the start and end timestamps for each word. On the other hand, speaker diarization is performed to generate speaker labels along with their respective start and end times. Next, we match the timestamp data of each word with the speaker labels, determining the speaker associated with each word. Finally, by grouping the words according to the same speaker, we merge them into sentences, while different speakers are separated accordingly. The result is a verbatim transcript that includes speaker information.

C. Transcripts Preprocessing

Before inputting the transcripts generated by ASR and speaker diarization into the summarization model we selected, we needed to perform several preprocessing steps to improve the readability and accuracy of the transcripts. This preprocessing aims to enhance the performance of the model in subsequent stages.

*1) **Conversion from Simplified to Traditional Chinese and Custom Dictionary Correction**:* To ensure that the output transcripts and report are presented in Traditional Chinese, we used opencc to convert the Simplified Chinese in the original transcripts to Traditional Chinese. Additionally, we compared Traditional Chinese texts, such as articles and novels, by converting them to Simplified Chinese and back to Traditional Chinese using opencc. This process allowed us to detect errors in conversion rates. We recorded common mistakes and rare words that opencc might misconvert, then created a custom dictionary for manual correction.

*2) **Noise Removal – Text Cleaning with Stopwords**:* For noise removal in the text, we ultimately opted for a more basic approach of cleaning the text using a stopword list, which included particles and filler words. Although this method is less effective in reducing redundancy compared to keyword extraction algorithms like TF-IDF and TextRank, it has specific advantages given our objective of summarizing meetings and analyzing team roles, especially when using large language models with strong comprehension abilities. The advantages of using stopword-based cleaning over keyword extraction include:

1) Preserving Contextual Integrity: After text cleaning, the model still receives relatively complete sentences and context. This helps the language model to better understand the content and key points through contextual inference.
2) More Accurate Semantic Analysis: By using cleaned text, the model can capture not only keywords but also the semantic roles of entire sentences within the text. This leads to more comprehensive and accurate summaries.
3) Flexibility: The language model is not limited to keywords; it can perform deeper understanding based on

context and semantics. This allows the summary to be driven not just by word frequency but by semantic importance.

Although the efficiency improvements are modest, this approach ensures both accuracy and completeness are retained.

*3) **Segmenting Transcripts for Input into the Model**:* Next, we divided the processed transcript into several sections and input them into the model, which was configured to correct wording and add punctuation. The outputs were then reassembled into a complete transcript. At this point, we had a fully processed transcript with text cleaning, punctuation added, and typos corrected. The reason for this approach is that while the model's input does not have a strict character limit (within a total upper limit), there is a restriction on the number of tokens it can output. Therefore, it was necessary to ensure that the length of each processed section stayed within the model's limit to prevent any content from being cut off. We set the maximum token limit to 8192, which is the output upper bound for the model we selected.

*4) **Speaker Name Mapping Based on User Input**:* Lastly, since the meeting summaries and analyses are derived from post-meeting audio recordings, we do not have prior knowledge of the speakers' identities. Therefore, speaker mapping must be completed by the user after reviewing the initial draft of the transcript. Once the user provides this information, we conduct the speaker mapping prior to finalizing the meeting summary, enhancing the readability of the meeting report.

D. *Large Language Model*

To improve the efficiency of team meetings and gain a deeper understanding of team roles, we plan to use a large language model (LLM) to help us do the following things:

- *Summarization*: We plan to utilize large language models (LLMs) to automatically summarize meeting discussions. By doing so, we aim to streamline the process and improve overall meeting efficiency. These models will extract the key points from the conversation, helping participants quickly grasp the main ideas and reducing the time spent on reviewing lengthy transcripts. This approach not only optimizes workflow but also ensures that important details are highlighted, leading to more productive and focused decision-making.
- *Role Analysis Using Belbin Team Roles*: We process the input transcript to identify the roles of participants based on the Belbin Team Roles, proposed by Dr. Meredith Belbin, aiming to achieve a balanced team and increase productivity. [1]
    1) Shaper – Highly driven and motivated, thrives under pressure and pushes others to take action.
    2) Plant – Creative and innovative, often works alone and excels at solving complex problems.
    3) Co-ordinator – Brings the group together, identifies talents, and helps align the team towards shared goals.
    4) Monitor Evaluator – Analytical and critical thinker, makes shrewd judgments after carefully considering all factors.
    5) Resource Investigator – Great communicator and negotiator, adept at exploring opportunities and finding resources.
    6) Implementer – Organized and reliable, approaches tasks systematically and gets things done efficiently.
    7) Team Worker – Supportive and adaptable, fosters collaboration and avoids conflict within the team.
    8) Completer-Finisher – Detail-oriented, follows through on tasks with precision and dislikes carelessness.
    9) Specialist – Knowledgeable in a specific area, focused on maintaining high standards and providing expertise.
- *Circular Discussion (Ghost Discussion) Detection:* Circular discussions, often referred to as ghost discussions, occur when participants in a meeting repeatedly address the same topics without reaching a conclusion. Detecting such patterns is critical in summarization tasks, as they indicate redundancy and lack of progress in conversations. In this section, we describe the mathematical techniques used for detecting circular discussions in conference meetings.

III. METHODOLOGY

A. *Model for generating verbatim transcripts with speaker annotations*

In this section, we aim to obtain verbatim transcripts and speaker diarization text, and then merge them by aligning their timestamps. To achieve this, we have chosen two approaches: (1) Whisper combined with the Pyannote model, and (2) WhisperX, an integrated model based on Faster Whisper and Pyannote.

*1) **Whisper+Pyannote**:* Whisper is a multi-task, multilingual seq2seq Transformer model trained through weak supervision on 680,000 hours of annotated speech data. Its ASR capabilities are highly accurate across most languages, making it one of the best-performing ASR models available. While Whisper can also handle tasks like voice activity detection and translation, it does not natively support speaker diarization. To include speaker diarization in our system, we combine Whisper with the Pyannote model, which specializes in identifying different speakers in the audio. This allows us to accurately separate speakers and transcribe their speech. [2]

Whisper handles four types of tasks: (1) transcribing English speech into English text; (2) transcribing and translating non-English speech into English text; (3) transcribing non-English speech into text in the original language; and (4) detecting non-speech segments, such as audio containing only background sounds. [2]

Pyannote is a language model specifically designed for handling speaker diarization. Its task pipeline includes voice activity detection, overlapped speech detection, speaker change detection, and speaker embedding, ultimately producing speaker annotation data. [3]

We combine the verbatim transcript generated by Whisper with the speaker annotations from Pyannote. The process can be simplified into the following formula:

$$\text{chunk}(i) = \begin{cases} \text{chunk.text} + \text{whisper\_i[text]}, & \text{if spk}(i) = \text{spk\_now}(i) \\ \text{Append(chunk)}, & \text{if spk}(i) \neq \text{spk\_now}(i) \end{cases}$$

- chunk(i) represents the text and metadata (e.g., timestamps, speaker) for chunk i.
- whisper_i represents the text and metadata (e.g., timestamps, speaker) for a portion of the whisper output.
- spk(i) is the speaker associated with the chunk.
- spk_now(i) is the speaker of the data being processed.
- The first case appends the text if the speaker remains the same; the second case appends the chunk and resets when a new speaker is detected.

*2) **WhisperX**:* WhisperX is a model based on the faster-whisper backend, utilizing wav2vec2 technology, and integrated with pyannote for speaker diarization. Compared to the original Whisper model, WhisperX offers a significant improvement in processing speed, achieving up to 70 times real-time transcription. It also greatly reduces the burden on the GPU, requiring less than 8GB of memory to run efficiently. Additionally, WhisperX excels at reducing hallucination errors, with pre-processing using Voice Activity Detection to further enhance transcription accuracy. [4]

One key feature that distinguishes WhisperX from the original Whisper model is its ability to directly generate verbatim transcripts with speaker name annotations. Additionally, WhisperX can break down the transcript into individual words, each accompanied by detailed information such as timestamps, speaker labels, and other parameters. This provides greater flexibility for further research and processing.

B. Simplified and Traditional Chinese conversion model - OpenCC

OpenCC (Open Chinese Convert) is a well-known open-source tool used for converting Simplified Chinese characters to Traditional Chinese and vice versa. Given the increasing demand for automated text processing in various fields, it is crucial to assess the accuracy of such tools to ensure the correct representation of Chinese characters in different writing systems. The test was conducted using a variety of texts, including formal and informal written materials. A total of 188,257 words were analyzed, covering different domains such as news articles, business documents, and colloquial conversations. The primary testing tool used was OpenCC's default configuration, without any custom modifications. The conversion output was manually compared to the expected Traditional Chinese output by human reviewers to evaluate the error rate. [5]

C. Models for Summarization, Role Analysis and Circular Discussion

As discussed in the Background section, there is a need for a model capable of performing summarization, role analysis, and detecting circular discussions. To address these requirements, we will evaluate five models: TextRank, PEGASUS, BERT, TAIDE, and Gemini. To assess the effectiveness of each model, we will utilize ChatGPT-4 to objectively grade their performance. The following sections provide a detailed overview of each model's functionality and approach.

*1) **TextRank**:* TextRank is an unsupervised graph-based algorithm for text summarization and keyword extraction, inspired by Google's PageRank. The algorithm represents text as a graph where sentences (or words) are nodes, and edges are formed based on similarities or co-occurrences between them. The centrality of each node is calculated iteratively using the following formula:

$$h(V_i) = (1-d) + d \cdot \sum_{V_j} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} h(V_j)$$

- V_i is a node
- $h(V_i)$ is the TextRank score of a certain node
- d is the damping factor, a constant value between 0 and 1, often set to 0.85
- W_{ji} is the weight of the connection between nodes

In this study, we utilize TextRank4ZH, a variant of the TextRank model specifically designed for Chinese text summarization. TextRank4ZH adapts the original algorithm to account for the unique linguistic structure of the Chinese language, allowing for more accurate extraction of key phrases and summarization. By leveraging this model, we ensure that the summarization process is optimized for Chinese text, improving the overall performance in handling Chinese meeting transcripts and documents. [6]

*2) **LexRank**:* LexRank is an unsupervised graph-based algorithm for text summarization. Inspired by the PageRank algorithm, LexRank constructs a graph where nodes represent sentences and edges represent the similarity between sentences. By iteratively calculating the PageRank values of nodes, LexRank determines the importance of each sentence and selects the most important sentences to form the summary. The PageRank value of a node is calculated as follows:

$$PR(V_i) = (1-d) + d \cdot \sum_{j \in in(V_i)} \frac{PR(V_j)}{C(V_j)}$$

where:

- $PR(V_i)$ is the PageRank value of node V_i
- d is the damping factor, typically set to 0.85
- $in(V_i)$ represents the set of all nodes pointing to node V_i
- $C(V_j)$ is the sum of the weights of all outgoing links from node V_j

[7]

*3) **PEGASUS**:* PEGASUS is a Transformer-based model specifically designed for abstractive summarization, focusing on the generation of concise and informative summaries. The model employs a novel pre-training objective known as Gap Sentences Generation (GSG). In this method, the model removes important sentences from the document, masking them,

and then generates those sentences based on the remaining text. This setup mimics the process of creating an abstractive summary. The key formula for the PEGASUS model, capturing the essence of the GSG pre-training objective, is as follows:

$$\text{GSG Loss} = -\sum_{i=1}^{m} \log P(y_i | x_{\text{masked}})$$

where:

- y_i represents the important (gap) sentences to be generated
- $y_m asked$ is the input document with the important sentences masked

[8]

*4) **BERT**:* BERT (Bidirectional Encoder Representations from Transformers) is a language model introduced by researchers at Google in 2018. It provides deep bidirectional representations of text. BERT is pre-trained on large amounts of unlabeled text data using two main tasks: Masked Language Modeling and Next Sentence Prediction. This pre-training allows BERT to capture contextual information and semantic relationships in language.

The core algorithm for BERT-based extractive summarization can be described as follows:

1) Let D be the input document consisting of sentences $[s_1, s_2, ..., s_n]$.
2) For each sentence s_i, obtain its BERT embedding $E_i = \text{BERT}(s_i)$.
3) Pass each E_i through a feed-forward neural network to obtain a relevance score:

$$score(s_i) = \sigma(W \cdot E_i + b)$$

Where W and b are learnable parameters and σ is the sigmoid function.

4) Rank sentences by their scores and select the top k sentences as the summary.

The objective function for training this model often uses binary cross-entropy loss:

$$L = -\sum (y_i \cdot \log(score(s_i)) + (1 - y_i) \cdot \log(1 - score(s_i)))$$

Where y_i is 1 if the sentence is in the reference summary and 0 otherwise.

This approach leverages BERT's powerful language understanding capabilities to identify key sentences, resulting in more coherent and context-aware summaries compared to traditional frequency-based methods. [9]

*5) **TAIDE**:* TAIDE-LX-7B is a large language model built upon the LLaMA2-7B architecture, developed specifically for Traditional Chinese language tasks. These tasks include, but are not limited to, text generation, multi-turn dialogue, summarization, and translation. The model has been extensively fine-tuned, with a strong focus on instruction tuning, allowing it to excel in task-oriented scenarios. This fine-tuning process enhances the model's ability to perform in professional settings such as office automation, interactive Q and A, and specialized tasks. TAIDE-LX-7B also integrates the unique linguistic and cultural characteristics of Traditional Chinese, making it a valuable tool for applications in Taiwan and other Chinese-speaking regions.

*6) **Gemini**:* To efficiently summarize the meeting transcripts, we leveraged Gemini 1.5 Flash via the Gemini API. The API processes raw meeting transcripts to generate concise and coherent summaries, aimed at capturing the core discussion points, decisions made, and action items without losing critical details. Gemini incorporates key innovations from Transformer models and Mixture of Experts (MoE) architectures to efficiently process large-scale data, making it highly suitable for tasks such as conference summarization. The details are listed in the fowllowing subsections:

- *Transformer and Attention Mechanism*
  The core component of Gemini's architecture is the Transformer, which is built around the attention mechanism. The Transformer processes input data using *self-attention*, a mechanism that allows the model to weigh the importance of different parts of the input. The basic equation for scaled dot-product attention is given by:

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

  Here, Q, K, and V represent the *queries*, *keys*, and *values*, respectively, which are derived from the input data. The dot product of Q and $K^\top$ computes a similarity score between different elements of the input, which is then scaled by the dimension d_k and passed through a softmax function to produce attention weights. These weights are then applied to the values V to generate the final output.
- *Multi-Head Attention*
  To capture different aspects of the data, the Transformer uses *multi-head attention*, where multiple attention mechanisms are applied in parallel. Each head focuses on different parts of the input. The formula for multi-head attention is:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O$$

  Each attention head is computed as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

  where W_i^Q, W_i^K, and W_i^V are learned projection matrices. The outputs of all heads are concatenated and linearly transformed using W^O.
- *Mixture of Experts (MoE)*
  To further scale the model and improve its efficiency, Gemini integrates Mixture of Experts (MoE) layers. MoE allows only a subset of the model's layers to be activated for a given input, which reduces computation while maintaining performance. The gating mechanism in MoE selects which experts to activate based on the input:

$$y = \sum_{i=1}^{n} G(x)_i \cdot E_i(x)$$

Here, x is the input, $E_i(x)$ represents the output of the i-th expert, and $G(x)_i$ is the gating function, which determines the weight assigned to the i-th expert. Only a few experts with the highest gating weights are activated, leading to sparse computation. [?]

- *Combining Transformer with MoE in Gemini*
  The combination of Transformers and MoE in Gemini allows the model to balance capacity and efficiency. The Transformer provides a robust mechanism for understanding context and relationships in the input data, while MoE enhances scalability by selecting specialized experts to handle different parts of the task, such as summarizing various sections of a conference. [10]

IV. Results

A. Analysis of ASR Transcript Performance and Error Types

In the task of Automatic Speech Recognition (ASR), we selected Whisper Large-v3 and WhisperX Large-v2 as the candidate models. To compare the performance and common error types of these models, we searched for 15 videos covering various topics, including popular science, psychological analysis, social issues, interpersonal relationships, and stand-up comedy. These videos featured between 1 to 8 speakers and ranged in duration from 5 minutes to 1 hour, from which we conducted verbatim transcription followed by manual correction and recording. Based on the errors observed during the experiments, we categorized the error types and established a "Transcript Error Type Reference Table," as shown in the table below, ultimately identifying the error types most likely to occur with each model.

Since this study focuses on the performance of Chinese verbatim transcripts, we will use the Character Error Rate (CER) [11] as the standard for comparing model performance, with the CER formula provided below. Furthermore, we will also consider the time taken by the models to process the audio and video files as an additional reference, using the ratio of the processing time of the models to the duration of the audio and video files as a comparison standard. Below are our experimental results:

$$\text{CER} = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

- S is the number of substitutions.
- D is the number of deletions.
- I is the number of insertions.
- C is the number of correct characters.
- N is the number of characters in the reference (N=S+D+C).

Although the performance and processing time ratios of both models are similar, WhisperX Large-v2 (hereafter referred to as WhisperX) slightly outperforms Whisper Large-v3 (hereafter referred to as Whisper).

Comparison table of error type tags:
- repetition looping: dark red
- Translate English into Chinese: light green
- Missing paragraph (missing more than 3 words (inclusive) or obviously missing a sentence): Orange, auxil-color 2
- Missing characters (within 2 consecutive missing characters): plum red, auxil-color 5, 40% lighter
- Typo - homophone recognition: light blue
- Typo - specific noun: purple
- Wrong name: green, auxil-color 6, 50% darker
- Unclear speech or words with very similar sounds or other errors: orange

p.s. If there are other errors, please describe them a little bit. ex: Illusion words (non-existent/extra words)

Video type
- Video length:
  - **Short** : within 10 minutes
  - **Medium** : 10~30 minutes
  - **Long** : 30 minutes or more
- Number of people in the video: **Single, two, multiple**

Ratio of test data:
- Video length (short : medium : long): (1:2:3)
- Number of people in the video (single : two : multiple): (1:2:3)

Fig. 2. Transcript Error Type Reference Table

TABLE I
Comparison of Different Models for ASR Transcript

	Whisper Large-v3	**WhisperX Large-v2**
Average CER(%)	13.58%	9.94%
Average ratio of Execution Time	2.253	2.311

When we analyze the data by speaker count and audio length, the performance gap between the two models is most pronounced in audio files with six or seven speakers. These recordings frequently feature overlapping speech and interruptions, resulting in noisy segments that make accurate transcription challenging. This often leads to errors like misinterpretations and missing segments.

TABLE II
Compare CER with people

Number of people	**Whisper CER(%)**	**WhisperX CER(%)**
1	6.52%	3.42%
2	11.12%	9.66%
3	16.06%	19.66%
4	6.26%	7.35%
6	18.39%	3.76%
7	22.04%	14.63%

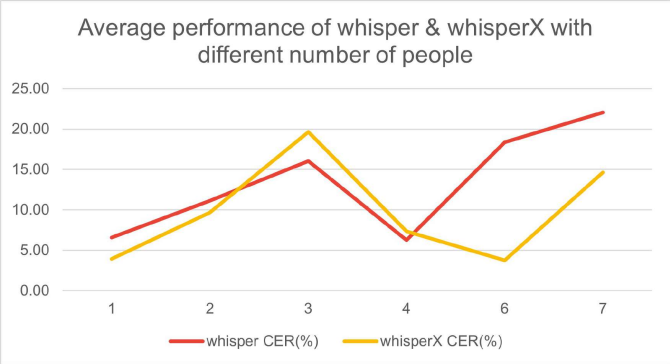

Fig. 3. CER with people curve

In the analysis segmented by duration, we observed the largest performance difference between the two models in audio files longer than 30 minutes. Initially, we assumed that audio length affected model stability. However, further cross-referencing of speaker count and duration revealed that most longer audio files involved more than six speakers, where Whisper performed less effectively. Conversely, when audio files had fewer speakers or only one, the performance difference between Whisper and WhisperX was minimal. A comprehensive review of audio length and error rate data showed no significant relationship between audio duration and error rate; rather, the main factors affecting model performance were the number of speakers and the presence of overlapping speech.

TABLE III
COMPARE CER WITH TIME

Length of time	**Whisper CER(%)**	**WhisperX CER(%)**
Under 10 minutes	17.82%	15.65%
Within 10 to 30 minutes	12.86%	11.43%
More than 30 minutes	14.35%	3.56%

During the proofreading process, we observed that unclear speech, background noise, and overlapping or interrupted speech were the primary factors affecting transcription accuracy. Both models struggled with homophones or sentences with similar sounds, often failing to produce the correct text. Examples include: as/us. Additionally, both models had difficulty recognizing proper names, specific terms, idioms, or phrases.

The most significant difference between Whisper and WhisperX, and what we consider Whisper's most serious drawback, is its tendency to produce repetition looping, where previously transcribed content is randomly and repeatedly inserted into later parts of the transcript. WhisperX does not have this issue. Moreover, Whisper is more prone to hallucinations in silent audio segments, where non-existent speech is transcribed.

However, WhisperX has its limitations as well. When multiple people speak simultaneously or when background noise is more prominent, the model tends to misidentify speech as noise, leading to missing sections in the transcript. After thorough evaluation and careful discussion, we ultimately selected WhisperX Large-v2 as the model to handle our ASR tasks.

B. OpenCC model testing

The results show that the average error rate is approximately 1.98%, with some cases peaking at no more than 5%. Since our testing method involved first converting Traditional Chinese text into Simplified Chinese, then converting it back to Traditional Chinese, and finally comparing the result with the original Traditional Chinese text, this error rate also includes errors from the first conversion. Therefore, the actual error rate in real-world use may be slightly lower. This indicates that OpenCC is highly reliable for most text processing tasks. Errors identified during testing were categorized as follows:

- **Radical changes:** Characters where the radical changes.
- **Character composition changes:** Characters where the composition changes.
- **Radical reduction:** Characters with reduced radicals in the Traditional version.
- **Radical addition:** Characters with additional radicals.
- **Interchangeable characters:** Characters used interchangeably.
- **Synonyms:** Characters that are semantically equivalent.
- **Homophones:** Characters that sound the same but have different meanings.

The errors were consistent across various texts, suggesting that adjustments to the OpenCC conversion logic could further reduce the error rate. Below is a table summarizing the error rates across different test samples:

TABLE IV
ERROR RATE & COMMON MISCONVERSIONS

number	**word count**	**error rate**
Part 1	42316	2.37%
Part 2	43359	1.83%
Part 3	40188	2.24%
Part 4	30718	1.70%
Part 5	31676	1.62%
Total	188257	1.98%

C. Summarization Models

We tested PEGASUS, BERT, TextRank, and LexRank summarization models on the meeting transcripts from our group. PEGASUS is a generative summarization model, while BERT, TextRank and LexRank are extractive models. After initial testing, BERT, TextRank and LexRank were quickly eliminated, and PEGASUS also showed poor performance after further tests on several transcripts, leading to its elimination as well.

The reason for eliminating these models is closely related to the nature of transcripts. As a form of text, transcripts are conversational, informal, and have a scattered focus, unlike

carefully crafted written articles. Extractive summarization models rely on calculating the importance of sentences in the text and using them to create a summary, which is not suitable for transcripts. This led to the elimination of BERT, TextRank and LexRank. PEGASUS, though generative, was trained mainly on regular articles, making it unable to handle the conversational and informal nature of transcripts effectively. Therefore, we ultimately chose large language models with stronger language comprehension to complete the task.

D. LLM selection

- MMLU Performance: Gemini Flash scored 77.9, outperforming Claude and Haiku combined, which scored 73.8 and 69.8, respectively. However, it still lagged behind GPT-3.5 Turbo, which achieved a score of 88.7. This indicates that Gemini Flash demonstrates strong capabilities in understanding and generating responses to a diverse range of questions, yet GPT-3.5 Turbo clearly excels in this area, reflecting its more advanced comprehension and contextual awareness.
- GPQA Performance: In the GPQA benchmark, Gemini Flash scored 38.6, which is higher than the combined score of Claude and Haiku at 35.7 and 30.8, respectively. Nevertheless, it is significantly below GPT-3.5 Turbo's 53.6. This result highlights that while Gemini Flash is competitive in answering questions accurately, GPT-3.5 Turbo displays superior performance, particularly in addressing complex inquiries that require deeper analytical thinking.
- DROP Performance: On the DROP benchmark, Gemini Flash achieved a score of 79.7, slightly ahead of the combined score of Claude and Haiku at 78.4 and 70.2, respectively, yet still trailing behind GPT-3.5 Turbo's 83.4. This shows that Gemini Flash has a commendable ability to understand and extract relevant information from text, indicating its effectiveness in data retrieval and reasoning tasks, although GPT-3.5 Turbo remains the leader in this domain.
- MGSM Performance: In the MGSM benchmark, Gemini Flash received a score of 75.5, which surpasses the combined score of Claude and Haiku at 71.7 and 56.3, respectively. However, it is still quite far from GPT-3.5 Turbo's impressive score of 90.5. This suggests that while Gemini Flash has solid capabilities in generating diverse and creative outputs, there is still room for improvement in its innovative potential when compared to GPT-3.5 Turbo.
- MMMU Performance: In the MMMU benchmark, Gemini Flash scored 56.1, while the combined score of Claude and Haiku was 50.2 and 68, respectively. This disparity highlights the challenges faced by Gemini Flash in certain specific tasks, while GPT-3.5 Turbo's high score of 90.2 underscores its expertise in various analytical challenges.
- MathVista Performance: For the MathVista benchmark, Gemini Flash scored 58.4, which is higher than the combined score of Claude and Haiku at 46.4 and 0, respectively. This result showcases Gemini Flash's potential in solving mathematical problems. However, GPT-3.5 Turbo's score of 63.8 illustrates its superior abilities in mathematical reasoning, a critical skill for many applications.

Based on this comprehensive evaluation, the decision to select Gemini Flash as our primary model stems from its competitive performance across multiple benchmarks. While it may not lead in every specific aspect, its overall capabilities in understanding and generating relevant responses make it a strong candidate, especially for tasks requiring data extraction and reasoning.

Ultimately, choosing Gemini Flash aligns with our goal of producing accurate, insightful, and relevant outputs, ensuring that end-users receive the highest quality of information possible. The selection is justified by Gemini Flash's strong performance in key areas, even when facing tough competition from models like GPT-3.5 Turbo.

TABLE V
PERFORMANCE COMPARISON OF GEMINI FLASH, CLAUDEHAIKU, AND GPT-3.5 TURBO ON MULTIPLE BENCHMARKS

Benchmark	Gemini Flash	ClaudeHHaiku	GPT-3.5 Turbo
MMLU	77.9	73.8	69.8
GPQA	38.6	35.7	30.8
DROP	79.7	78.4	70.2
MGSM	75.5	71.7	56.3
MMMU	56.1	50.2	68
MathVista	58.4	46.4	0

Table V presents a comparative analysis of Gemini and the other models across various evaluation criteria. The following points elaborate on each criterion and provide insights into the decision-making process regarding model selection.

V. CONCLUSION

Through testing, we found that the performance of WhisperX in ASR slightly surpasses that of the original Whisper. WhisperX addresses the repetition looping issue present in the original Whisper, enhancing the readability of transcripts. Additionally, WhisperX improves the original Whisper's stability in multi-speaker recordings. However, WhisperX still has some unresolved issues, leaving room for further improvement. In the future, we plan to use error type statistics and prompt engineering to train LLMs to help correct errors, with the aim of further reducing the error rate.

Additionally, from the experimental results, we observed that ASR processing is quite time-consuming, taking more than twice the length of the original audio file. This is mainly because the current speaker diarization model is limited to pyannote. We will continue to explore faster and more cost-effective computational methods to improve processing efficiency.

In conclusion, the evaluation of various models for summarization tasks revealed that the Gemini model consistently outperformed others, particularly in key areas such as summary quality, repetitive content detection, and linguistic diversity. While TAIDE excelled in adaptability, Gemini's superior performance in generating coherent, precise, and linguistically rich summaries makes it the optimal choice for the task at

hand. Although Gemini has some limitations in adaptability, its strengths in producing high-quality and engaging outputs outweigh those drawbacks. Therefore, it was selected as the most suitable model for this project.

Moving forward, we plan to further refine the model's performance, particularly in handling more diverse meeting transcript scenarios. Integrating Gemini with advanced prompt engineering techniques and continuously improving error correction will ensure even more accurate, efficient, and context-aware summarization, meeting the growing demands for high-quality automatic summarization in practical applications.

REFERENCES

[1] Belbin, "The Nine Belbin Team Roles," Belbin, 2024. https://www.belbin.com/about/belbin-team-roles

[2] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision," arXiv:2212.04356 [cs, eess], Dec. 2022, Available: https://arxiv.org/abs/2212.04356

[3] H. Bredin et al., "pyannote.audio: neural building blocks for speaker diarization," arXiv:1911.01255 [cs, eess], Nov. 2019, Available: https://arxiv.org/abs/1911.01255

[4] M. Bain, J. Huh, T. Han, and A. Zisserman, "WhisperX: Time-Accurate Speech Transcription of Long-Form Audio," arXiv.org, Jul. 11, 2023. https://arxiv.org/abs/2303.00747

[5] C. Kuo, "Open Chinese Convert ," GitHub, May 05, 2022. https://github.com/BYVoid/OpenCC

[6] Chirantana, M., Ajit, K., Madhurima, D., Asit, Apurba, S., Graph-Based Text Summarization Using Modified TextRank, August 2018, doi: 10.1007/978-981-13-0514-6_14.

[7] B. University, "Understanding LexRank Text Summarization Algorithm - Rishabh Singh, Bennett University - Medium," Medium, May 06, 2021.

[8] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization," arXiv:1912.08777 [cs], Jul. 2020, Available: https://arxiv.org/abs/1912.08777 https://rishabh71510.medium.com/understanding-lexrank-text-summarization-algorithm-fb2c5415e0b6 (accessed Oct. 11, 2024).

[9] D. Miller, "Leveraging BERT for Extractive Text Summarization on Lectures," arXiv:1906.04165 [cs, eess, stat], Jun. 2019, Available: https://arxiv.org/abs/1906.04165

[10] T. R. McIntosh, T. Susnjak, T. Liu, P. Watters, and M. N. Halgamuge, "From Google Gemini to OpenAI Q* (Q-Star): A Survey of Reshaping the Generative Artificial Intelligence (AI) Research Landscape," arXiv.org, Dec. 17, 2023. https://arxiv.org/abs/2312.10868

[11] "Char Error Rate — PyTorch-Metrics 1.4.0.post0 documentation," lightning.ai. https://lightning.ai/docs/torchmetrics/stable/text/char$_{e}rror_{r}ate.html$