A Framework for Abstractive Summarization of Conversational Meetings

Vincent Marklynn
Department of Computer Science
Seattle University, Seattle, USA
vmarklynn@seattleu.edu

Anjali Sebastian
Department of Computer Science
Seattle University, Seattle, USA
asebastian@seattleu.edu

Yong Long Tan
Department of Computer Science
Seattle University, Seattle, USA
ytan@seattleu.edu

Wan D. Bae
Department of Computer Science
Seattle University, Seattle, USA
baew@seattleu.edu

Shayma Alkobaisi*
College of Information Technology
United Arab Emirates University, Al Ain, UAE
shayma.alkobaisi@uaeu.ac.ae

Sada Narayanappa
Computer Science
University of Denver
sadananda.narayanappa@du.edu

***Abstract*—Unlike extractive summarization, abstractive summarization creates summaries by synthesizing new words and sentences that maintain the original meaning of the source. This presents new challenges that researchers and developers face when developing language processing models for text generation. Utilizing advanced models in automatic speech recognition and natural language processing such as OpenAI's Whisper and Meta's BART, we simplify the process of speech recognition and abstractive summarization of long meetings. We propose a system framework and conduct analysis of speech to text specifically in conversations with more than two speakers in a meeting environment. Through both quantitative and qualitative analysis we evaluate the proposed model performance compared to the BART base model, and show that with summarizing long meeting dialogues our model improved summarization by 139.6% over the base model in the ROUGE-LSUM metric. Our proposed framework for abstractive summarization is a practical and accurate solution providing accessibility accommodations to hard-of-hearing people and accurate and insightful analysis to industry and academia.**

***Index Terms*—abstractive summarization, speech to text conversion, text summarization, BART, Transfer Learning**

I. Introduction

Meetings with multiple speakers can be challenging for individuals who require accessibility accommodations, such as hard-of-hearing people. Audio-to-text conversion has been a helpful tool to assist these individuals. Still, it does not address the issue of identifying speakers in a meeting and generating a summary of conversation for the given meeting. This increases the necessity of a pipeline that can identify speakers and summarize a given audio file.

In addition to providing accessibility accommodations for individuals who are hard of hearing, it is useful to provide accurate and insightful analysis to both researchers and end users, which can help in understanding and utilizing the content of the meetings better. This can be especially beneficial for organizations that need to monitor multiple meetings and key information for decision-making and research purposes.

* Corresponding author

We propose a pipeline for audio analysis, which integrates automatic transcription, speaker diarization and summarization of conversational meetings. The development of our proposed framework consists of the following three phases:

- Speech to text conversion: An automatic speaker recognition (ASR) module was developed by utilizing OpenAI Whisper [1] and Pyannote [2]. OpenAI Whisper was used to automatically transcribe audio files and Pyannote was used to perform speaker diarization. Output texts from the two were merged into a resulting text.
- Text summarization: We developed a transfer learning (TL) based text summarization model from a BART base model [3], a sequence-to-sequence transformer pretrained on the CNN/Daily Mail and SAUSUM datasets. The base model was retrained on a machine transcribed AMI/ICSI corpus to improve summarization for longer meetings.
- User interface: A front-end web application was implemented using Django [4], where users can upload an audio file and receive a transcript and a summary. Modules of other functionalities, such as identification of speakers and meeting content analysis can be easily integrated.

II. Related Work

In recent years, the transformer encoder-decoder architecture has been widely used in the field of natural language processing (NLP). The essential components integrated into our proposed framework are described below.

Bidirectional and Auto-Regressive Transformer (BART) [5] developed by Facebook is a model that is pretrained by corrupting the input text using some noise schemes and learning a model to reconstruct the original input. It reads the corrupted input in both directions, left to right and vice versa. The bidirectional encoder produces a set of hidden states that capture the meaning and context of the input text. Then, this collection of hidden states will get pushed to the autoregressive decoder. The decoder is another component that generates the output text one element at a time, where each element

is conditioned on the previously generated elements. BART can be used for various NLP tasks, for example, generation, translation, text classification, summarization, etc.

QMSum [6] is a novel benchmark designed for evaluating the performance of summarization models in the context of multi-domain meetings. This benchmark focuses on query-based summarization, which aims to generate concise summaries in response to specific user queries rather than generic summaries of entire meetings. QMSum contains a diverse collection of meeting transcripts, queries, and reference summaries, which helps facilitate the development and evaluation of more advanced, context-aware models for meeting summarization. For our work, these reference summaries will be used for evaluation, as we assume that they can act as ground truth.

Whisper [7] is a system designed for automatic speech recognition, developed by OpenAI. It converts spoken language into written text using advanced speech recognition techniques. It has been trained on a vast dataset of multilingual and multitask supervised data gathered from the web. It is considered a state-of-the-art speech recognition system, providing highly accurate transcriptions promptly and outperforming many existing speech-to-text solutions. We utilize the base model version of Whisper and incorporate it into our system framework.

Pyannote [8] is a Python library that provides tools for various tasks in multimedia processing. We only utilize their speaker diarization tool to determine "who spoke when." Speaker diarization is the process of segmenting an audio stream into speaker-homogeneous sections. The Pyannote's speaker diarization contribution lies in its open-source nature, fostering the continuous improvement and expansion of its capabilities. It facilitates effective and flexible audio segmentation into speaker-homogeneous sections.

Our summarization of spoken dialogues is closely related to a method of summarization for spontaneous speech [9]. The authors employ a two-stage method for both speech-to-text and speech-to-speech automatic summarization. This involves sentence extraction and word-based sentence compaction, optimizing a balance of linguistic and grammatical likelihoods, information content, and confidence measures. Applied to spontaneous presentations, their approach proved effective in summarizing speech. However, their approach diverges from ours in that it is fundamentally extractive while we aim at abstractive summarization, generating new words and sentences to maintain the essence of the source material. Furthermore, we utilize the BART sequence-to-sequence transformer model, which is distinctly different from the speech unit extraction and concatenation method employed in their research. Building on these foundational methodologies, our work employs transfer learning, enhancing our model's performance by introducing more diverse and distinct datasets.

III. Methods

A. System Architecture Overview

The proposed framework integrates a pipeline of speech to text conversion and summarization into a front-end web application. An overview of the proposed framework is shown in Figure 1. The pipeline takes large audio files (approximately 60 minutes long), transcribes, and diarizes the conversation. The web interface enables users to upload a meeting recording to get the full meeting transcript and summary. It also provides the users with interactive tools to work with the system. A list of the existing NLP tools we used to implement the framework includes: Whisper Open AI [1], Pyannote [2], BART [3], FFMEG [10], Hugging Face Transformers [11], and Django [4].

B. Speech to Text Conversion

The system starts with a speech to text conversion model that forms a data ingestion pipeline and it converts audio files to text files, which are the inputs for the text summarization model. The speech-to-text conversion model works with audio files that have multiple speakers conversing. The audio file (WAV format) is processed using Whisper [1] and Pyannote [2] in parallel. The Whisper module generates a timestamped transcript but the transcript does not identify speakers while the Pyannote module identifies speakers and timestamp durations. The whisper output is in plain text format and the Pyannote output is in RTTM [12] format. We combine both to generate a transcript identifying speakers, timestamp duration, and dialogue.

The speech to text conversion model is trained using long meeting audio files such as the AMI [13], [14] and ICSI [15], [16] audio meeting files. This model enables scope to re-train the text summarization phase for domain specific meetings and other audio recordings aside from the AMI and ICSI audio meeting files. The model retraining is not restricted to the availability of transcripts.

C. Text Summarization

BART [5] has rich documentation that can guide users to develop NLP models for their specific needs. Users can take a BART base model and retrain it by feeding their own datasets. BART is one of the promising pretrained NLP models for abstractive summarization. However, BART has some limitations since it has not been trained on any long conversation datasets involving multiple speakers.

We develop a new variant of BART based on transfer learning (TL), BART_{ASC}, for abstractive summarization of conversation. We take the BART base model [3] pretrained on the news articles from CNN and the Daily Mail, and phone text messages from SAUSUM datasets. We then retrain the model with texts generated using AMI and ICIS audio files.

For faster prototyping, we utilize the HuggingFace Trainer API that allows us to use trained models and save the best. The Trainer API also allows us to use TensorBoard to evaluate the training and evaluation performances. Once uploaded to HuggingFace, we utilize the instant Inference API to use our model for demos. Both quantitative and qualitative analysis for the model performance are presented in Section IV.

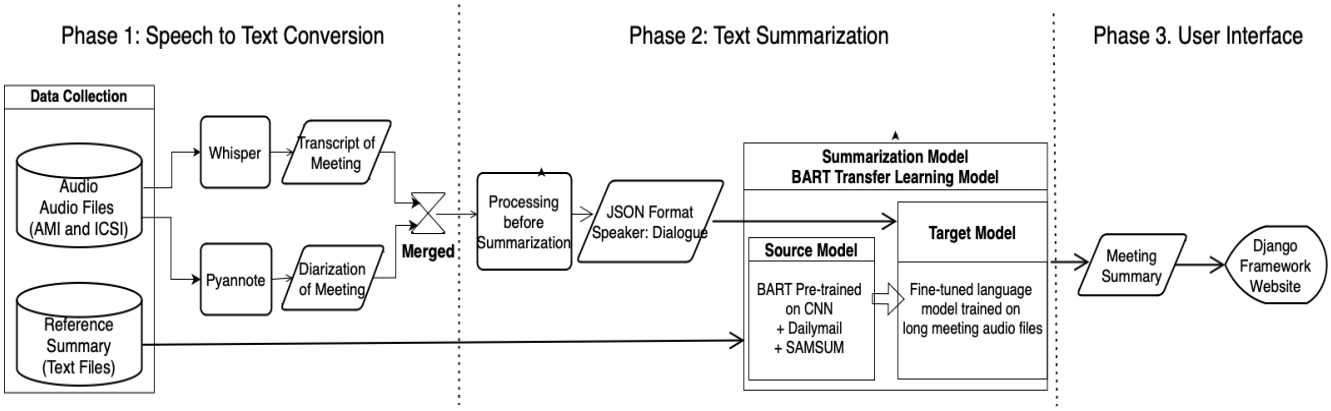

Fig. 1. An overview of the system framework

D. User Interface

We implemented a web application using the Django [4] framework to demonstrate our proposed NLP pipeline with the BART$_{ASC}$ model for abstractive summarization of conversational meetings. This web application includes two major components, transcription and summarization, but note that the proposed approach can be used in various NLP applications.

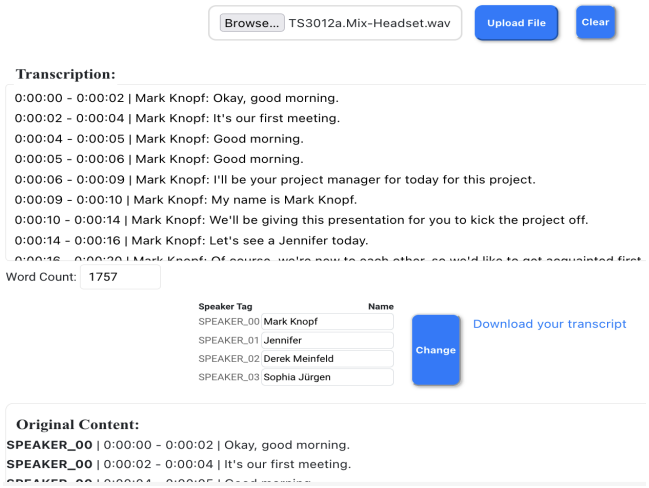

Fig. 2. A snapshot of web user interface

The first part of our user interface provides users with the following functionalities for transcription: (1) upload a audio file stored in a local folder, (2) transcribe the audio file and display the transcribed content, (3) edit the content if needed to change words, numbers, or sentences. It is an essential functionality in real applications because the accuracy of converting audio to text heavily depends on the quality of the microphone input, (4) enter the name of each speaker and this updates the transcribed texts with the speakers' names. The system can be easily updated with an additional functionality that takes pre-recorded voice of the meeting attendees and tags each speaker's name to the transcribed texts, and (5) download the final transcript. Our proposed system also incorporates an audio player and displays an unmodifiable original content box, allowing users to listen and track the progress. Users can hover over and click on any point in the timeline, and the audio player will automatically jump to the selected time. A snapshot in Figure 2 demonstrates the interface components that support these functionalities.

The second part of the user interface includes several functionalities for summarization. The system allows users to obtain a summary of the final transcript processed in the first part. It also provides users keywords extracted from the transcript, arranged from top to bottom, the first list contains single words, the second list contains pairs of words, and the third list contains groups of three consecutive words. These keywords can help users better comprehend the conversation.

IV. Experiments

A. Datasets and Experiment Setup

AMI Corpus [13], [14] and ICSI Corpus [15], [16] are publicly available datasets that contain 100 hours and 70 hours of meeting conversations recorded respectively. The audio files consist of recording of multiple speakers interacting in workplace meetings. There are 169 files in total which are in .wav format. The duration of the files is 20 - 90 minutes.

We also used reference summaries obtained from the paper [6] to train the model. The reference summaries are made by humans for the same meetings found in the AMI and ICSI Corpus and are in text-format. The summaries are around a paragraph in length with 80 to 150 words.

The proposed framework was implemented in Python 3.8 using various tools. We used 1-GPU for carrying out the processing on the audio files. It took approximately 3 minutes. With 1-GPU, the time to generate a transcript for a audio file was 50 - 60 minutes long. Table I shows training hyperparameters used in our experiments. Python source code for the framework is available in a public GitHub repository [17] and the proposed models are on Hugging Face [18].

B. Performance Metrics

Our evaluations on the performance of the models are based on Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [19]. ROUGE compares automatically produced

TABLE I
MODEL TRAINING HYPERPARAMETERS

Parameter	Value
Learning rate	5e-05
Optimizer	Adam with betas=(0.9, 0.009) and ϵ=1e-08
Learning rate scheduler	Linear
Batch size	training: 1, validation: 1
Validation split rate	0.20
The number of epochs	10

summaries against reference summaries which are human produced. For example, if a model produces a summary of "the cat was found under the bed," it is compared against "the cat was under the bed." In order to use ROUGE evaluations, there must be reference summaries by humans. In our experiments, we used datasets that include their summaries done by humans [6]. ROUGE calculates the following three metrics:

1) Precision = Number of overlapping N-grams / Number of tokens in the generated text. This measures how much of the model summary was relevant.
2) Recall = Number of overlapping N-grams / Number of tokens in the reference text. It measures how much of the reference summary is recovered from the reference.
3) F_1-score = $\frac{2*Precision*Recall}{Precision+Recall}$. F_1-score symmetrically represents both precision and recall in one metric.

In our analysis, we used the following ROUGE methods:

- ROUGE-N measures the overlap between the generated text and the reference text in terms of n-grams. To compute the Rouge-N score, (1) tokenize the generated text and the reference text into n-grams of size N, (2) compute the number of overlapping n-grams between the generated text and the reference text, and (3) compute the precision, recall, and F_1-score using the formulae above.
- ROUGE-L measures the overlap between the generated text and the reference text in terms of longest common subsequences. To compute the Rouge-L score, (1) tokenize the generated text and the reference text into individual words, (2) compute the longest common subsequence between the generated text and the reference text, and (3) compute the precision, recall, and F1-score using the following formulae.
- ROUGE-LSUM computes the Rouge-L score for multiple sentences, and then averages the scores. Rouge-Lsum is the same as Rouge-L, but it is calculated at the document level, considering all sentences together, rather than calculating it per sentence and then averaging the scores.

While models with higher ROUGE values are considered better models, the optimal ranges of the scores vary depending on the datasets. Moreover, these metrics have limitations as they primarily focus on the syntactical overlap and disregard semantic aspects. For instance, if the model effectively rewords and restructures the original text, Rouge scores might not fully capture its performance. To overcome these limitations, it is beneficial to involve multiple experts with diverse backgrounds who are skilled at summarizing. By incorporating their feedback along with the performance metrics, the quality of the output generated by the model can be significantly enhanced. This approach offers a more comprehensive evaluation that considers both syntactical and semantic aspects of summarization.

C. Quantitative Analysis

In this section, we present quantitative analysis on the performance of the BART$_{ASC}$ model that was trained on the ACSI and AMI datasets using the HuggingFace Trainer API. The model is a transfer learning model of the pretrained BART base model that was trained on the CNN Daily Mail and SAMSUM dataset. Based on how these models were trained, the BART base model is more suitable for summarizing articles and short chat messages while BART$_{ASC}$ is more suitable for longer dialogues.

We configured the HuggingFace Trainer API to save the model with the lowest validation loss. We increased the number of epochs from 1 to 10. The validation loss continues to increase and our HuggingFace model automatically saved the model from the first epoch. The training results throughout the epochs are shown in Table II.

Table III shows the performance improvement that our model BART$_{ASC}$ provides over the BART base model. For our experiments, each meeting is considered a single document; therefore, ROUGE-L and ROUGE-LSUM are most appropriate for the model evaluation [19]. The improvements are 68.4% in ROUGE-1, 165.6% in ROUGE-2, 52.0% in ROUGE-L and 139.6% in ROUGE-LSUM.

D. Qualitative Analysis

This section presents qualitative analysis on the performance of the BART$_{ASC}$ model on three different audio files: (1) short prose/dialogue and (2) long meeting. We compared the results of our proposed pipeline to the human reference summary. Summaries from BART$_{ASC}$ and their corresponding summaries by humans are shown in Table IV.

As shown in Table IV, our model can select important details from short prose and dialogues. The result from our model testing on short prose/dialogue is more comprehensive, accurately identifying the specific locations visited by the individuals. However, due to a lack of diversity in the reference data, it has a tendency to pick meeting lingos, even when the dialogue is not from a meeting setting.

For the long meeting, our model covers about 60% of the content of the reference summary and performs well in most parts. However, in the second sentence, the phrase "project manager" is unnecessarily repeated at the end.

E. Evaluation Remarks

The proposed system poses several limitations due to the quality of the human reference and the availability of data. Expanding the scope and performance of our summarizer to cover different meeting domains requires access to more diverse and comprehensive datasets. This would enable the model to learn

TABLE II
BART$_{ASC}$ MODEL TRAINING PERFORMANCE THROUGHOUT EPOCHS

Epoch	Validation loss	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-LSUM
1	**3.2308**	**38.43**	**13.65**	**22.37**	**35.24**
2	3.3026	40.18	11.79	23.27	36.47
3	3.5199	39.82	12.16	22.78	36.50
4	4.0508	40.43	11.66	23.00	36.88
5	4.6988	38.41	9.83	20.39	34.20
6	4.9590	38.58	9.63	20.87	35.03
7	5.4264	38.28	9.58	21.14	34.60
8	5.4887	38.30	9.69	21.24	34.16
9	5.8044	39.96	10.43	22.69	36.24
10	5.8987	39.09	10.84	21.91	35.51

TABLE III
PERFORMANCE COMPARISON BETWEEN BART BASE MODEL AND BART$_{ASC}$ (EPOCH = 1)

Model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-LSUM
BART base model	22.82	5.14	14.71	14.71
BART$_{ASC}$	38.43	13.65	22.37	35.24
Improvement (%) through retraining	**68.4%**	**165.6%**	**52.0%**	**139.6%**

TABLE IV
QUALITATIVE ANALYSIS OF BART$_{ASC}$

Short prose/dialogue [20]	
BART$_{ASC}$	The whole vacation was spent in Paris with my parents. **They saw the Eiffel Tower, Nortre-Dame Cathedral, Seine River, and the famous art museums.** They also went a boat tour around the city, and they had **a lot of fun sitting in cafes and watching people.**
Human	#Person1# asks Person2# what #Person2# did during the vacation to Paris. #Person2# saw **all the most famous places and enjoyed sitting in cafes most.** #Person1#'s looking forward a vacation.
Long meeting [6]	
BART$_{ASC}$	This is the **first meeting of the new remote control project.** Project manager introduced the team members to each other and the project manager. The team first **got acquainted with each other by drawing their favourite animals.** Next, the team discussed the working design of the remote control device. They decided to **make it more user-friendly** than the previous one, so they made some changes to the design. **The final product would be a remote control with** a fancy look, **ease of use**, and **cost 25 Euros**.
Human	Manager gave the team members **a brief introduction of the new remote control project**, then the team **started an ice breaking game by drawing their animals. The expected selling price of the remote is 25 Euros** and the production costs would be limited to twelve and a half Euro per unit. Marketing plan would depend on overhead costs. The team agreed that the remote needs to be robust, and they wanted **it to be a device for all with accessible**. **buttons**. Project manger suggested it to be only a television remote control instead of a multi-functional one

and adapt to various meeting scenarios, ultimately improving its accuracy and generalizability.

It is also worth mentioning that additional tasks to deal with audio files were required due to the unsuitability of the format of the obtained dataset for summarization tasks. A typical dataset for summarization requires at least three columns: id, original text, and reference summary. However, since the obtained dataset is a transcription of audio corpuses, we had to perform data wrangling procedures to obtain reference human summaries for the ICSI and AMI datasets.

Finally, due to the fact that a particular language was not specified for the Whisper model, it was inferring languages from accents. This may have caused errors in the transcriptions where Whisper switched to another language before reverting to English.

V. CONCLUSIONS AND FUTURE WORK

We introduced a pipeline that transcribes, diarizes, and summarizes conversation meetings using Whisper, Pyannote, and a TL based BART model, resulting in text summarization improvement of 52.0% and 139.6% over the BART base model in the ROUGE-L and ROUGE-LSUM metrics, respectively. The proposed solution is integrated into an user friendly web application built with Django, allowing users to upload meeting recordings and obtain comprehensive transcripts, summaries and keywords. This work paves the way for more effective meeting analysis and information accessibility in various contexts and industries.

For future work, first we will focus on the evaluation of models. Although the ROUGE methods have been widely used in NLP applications, we found their limitations to evaluate

models for abstractive summarization, which do not necessarily have strong word overlap with the reference summaries. Our evaluation will be complemented with alternative methods to assess the models. Secondly, we will conduct comparative experiments with various other meeting summarization tools that have been reported in [21]. Lastly, we will make our framework more applicable by improving the summarization capabilities and insightful analysis of workplace meetings. Additional application features could include visualizations of meeting using colors and waveforms, speaker recognition that allows the model to learn overtime to recognize previously encountered voices, and sentiment analysis.

ACKNOWLEDGMENT

This study received support from United Arab Emirates University - National Faculty Research Program (Grant No. G00004281) and Seattle University Thomas J. Bannan Endowed Chair.

REFERENCES

[1] Whisper, "Whisper," accessed in 2023. [Online]. Available: https://github.com/openai/whisper

[2] PyannoteWhisper, "Pyannotewhisper," accessed in 2023. [Online]. Available: https://github.com/yinruiqing/pyannote-whisper

[3] P. Schmid, "bart-large-cnn-samsum," 2004. [Online]. Available: https://huggingface.co/philschmid/bart-large-cnn-samsum

[4] Django, "Django," accessed in 2023. [Online]. Available: https://www.djangoproject.com

[5] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7871–7880.

[6] M. Zhong, D. Yin, T. Yu, A. Zaidi, M. Mutuma, R. Jha, A. H. Awadallah, A. Celikyilmaz, Y. Liu, X. Qiu *et al.*, "Qmsum: A new benchmark for query-based multi-domain meeting summarization," *arXiv preprint arXiv:2104.05938*, 2021. [Online]. Available: https://github.com/Yale-LILY/QMSum

[7] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," *arXiv preprint arXiv:2212.04356*, 2022.

[8] H. Bredin, R. Yin, J. M. Coria, G. Gelly, P. Korshunov, M. Lavechin, D. Fustes, H. Titeux, W. Bouaziz, and M.-P. Gill, "Pyannote. audio: neural building blocks for speaker diarization," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7124–7128.

[9] S. Furui, T. Kikuchi, Y. Shinnaka, and C. Hori, "Speech-to-text and speech-to-speech summarization of spontaneous speech," *IEEE Transactions on Speech and Audio Processing*, vol. 12, no. 4, pp. 401–408, 2004.

[10] FFmpeg, "Ffmpeg," accessed in 2023. [Online]. Available: https://ffmpeg.org

[11] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.6

[12] R. T. Evaluation, "Rich transcription evaluation," accessed in 2023. [Online]. Available: https://www.nist.gov/itl/iad/mig/rich-transcription-evaluation

[13] E. Commission, "Ami corpus," accessed in 2023. [Online]. Available: https://groups.inf.ed.ac.uk/ami/corpus/

[14] W. Kraaij, T. Hain, M. Lincoln, and W. Post, "The ami meeting corpus," 2005.

[15] U. B. International Computer Science Institute Speech Group, "Icsi corpus," accessed in 2023. [Online]. Available: https://groups.inf.ed.ac.uk/ami/icsi/

[16] A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke *et al.*, "The icsi meeting corpus," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, vol. 1. IEEE, 2003, pp. I–I.

[17] V. Marklynn, A. Sebastian, Y. L. Tan, W. D. Bae, S. Alkobaisi, and S. Narayanappa, "parrot," 2023. [Online]. Available: https://github.com/vmarklynn/parrot

[18] ——, "bart-large-cnn-samsum-icsi-ami-v2," 2023. [Online]. Available: https://huggingface.co/vmarklynn/bart-large-cnn-samsum-icsi-ami-v2

[19] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013

[20] Y. Chen, Y. Liu, L. Chen, and Y. Zhang, "DialogSum: A real-life scenario dialogue summarization dataset," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, Aug. 2021, pp. 5062–5074. [Online]. Available: https://aclanthology.org/2021.findings-acl.449

[21] V. Rennard, G. Shang, J. Hunter, and M. Vazirgiannis, "Abstractive meeting summarization: A survey," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 861–884, 2023.