Parrot: A Framework for Abstractive Summarization of Conversational Meetings

Vincent Marklynn
vmarklynn@seattleu.edu
Seatle University
Seattle, WA, USA

Anjali Sebastian
asebastian@seattleu.edu
Seattle University
Seattle, WA, USA

Yong Long Tan
ytan@seattle.edu
Seattle University
Seattle, WA, USA

Wan D. Bae
baew@seattleu.edu
Seattle University
Seattle, WA, USA

Sada Narayanappa
sada.narayanappa@lmco.com
Lockheed Martin Spaces
Denver, CO, USA

ABSTRACT

Around 83% of employees spend a third of their workweek attending meetings. We aim to conduct analysis from speech to text specifically in conversations with more than two speakers in a meeting environment. Besides granting accessibility accommodations to hard-of-hearing people, we also want to provide accurate and insightful analysis to customers and researchers. With recent advances in automatic speech recognition and natural language processing models such as OpenAI's Whisper and Meta's BART, we seek to simplify the process of speech recognition and abstractive summarization of long meetings. Unlike extractive summarization, abstractive summarization creates summaries by synthesizing new words and sentences that maintain the original meaning of the source. This is challenging due to the necessity of NLP models for text generation. Our solution consists of developing a pipeline that takes large audio files (approximately 1 hour) and transcribes and diarizes the conversation using OpenAI's Whisper and Pyannote. Our summarization solution involves retraining the BART model. Our retrained model makes the base-model compatible with summarizing long meeting dialogues and improves summarization by 148% (!!! should add results that show this number). The solution includes a front-end website built using Django that integrates our model and enables users to upload a meeting recording to get the full meeting transcript and summary.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**; • **Applied computing** → **Document management and text processing**.

KEYWORDS

abstractive summarization, natural language generation,

ACM xxxxx, April 01–04, 2022, City, State

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Vincent Marklynn, Anjali Sebastian, Yong Long Tan, Wan D. Bae, and Sada Narayanappa. 2023. Parrot: A Framework for Abstractive Summarization of Conversational Meetings. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (ACM xxxxx).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 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 the speaker in a meeting and generating a summary for the given meeting. Therefore, there is a need for a pipeline that can identify the speaker and summarize a given audio file. Besides granting accessibility accommodations to hard-of-hearing people, we also want to provide accurate and insightful analysis to customers and researchers, which can help them understand and utilize the content of the meetings better. In addition to providing accessibility accommodations for individuals who are hard of hearing, we want to offer precise and insightful analysis to both customers and researchers. This analysis can improve their comprehension and utilization of meeting content. This can be especially beneficial for organizations that need to monitor multiple meetings and extract key information for decision-making and research purposes.

We aim to conduct audio analysis from speech to text specifically in conversations with more than two speakers in a meeting environment. Besides granting accessibility accommodations to hard-of-hearing people, we also want to provide an accurate and insightful analysis for customers and researchers to leverage.

This audio analysis problem can be partitioned into four different parts:

- Automatic Transcription: We used OpenAI's Whisper to automatically transcribe audio files.
- Speaker Diarization: We used an opensource Python library called Pyannote to perform basic speaker diarization.
- Summarization: We used a pre trained BART model trained on the CNN/Dailymail and SAMSUM datasets.
- Visualization/Presentation: We developed a basic front-end web application where users can upload an audio file and receive a transcript and summary.

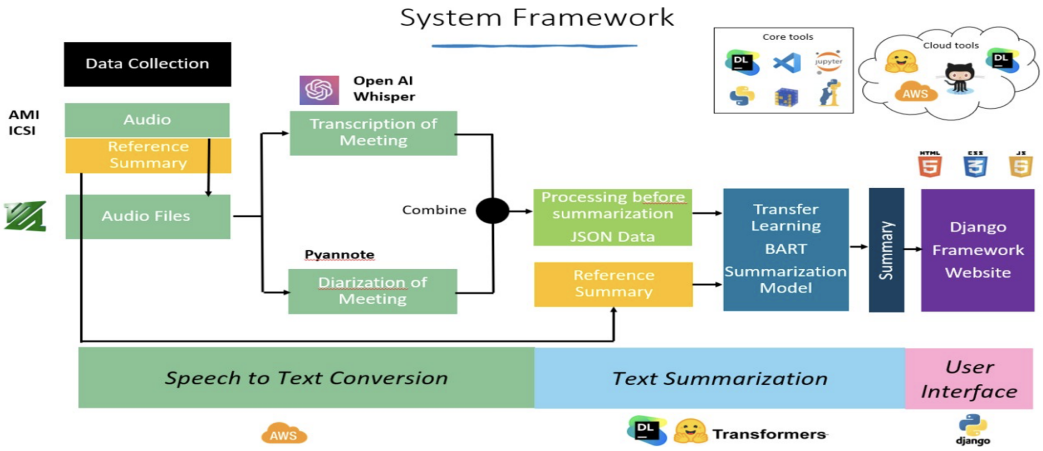

Figure 1: An overview of system framework

will need to replace with a new figure. remove title and tool figures. reduce # colors and bigger/clearer font

We created a working summarization pipeline starting with an Automatic Speaker Recognition (ASR) module consisting of OpenAI Whisper and Pyannote. For our summarization module, we performed transfer-learning on a pre trained BART sequence-to-sequence transformer model by training the model on a machine transcribed AMI/ICSI corpus to improve summarization for long meetings.

2 RELATED WORK

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

BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension [10]: BART 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 or vice versa. The bidirectional encoder produces a set of hidden states that capture the meaning and context of our 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.

QMSum [12]: A new benchmark for query-based multi-domain meeting summarization. A short summary of the paper's main ideas and contributions needs to be added. A New Benchmark for Query-based Multi-domain Meeting Summarization. QMSum 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, we only reuse their reference summaries. These reference summaries will be used for evaluation, as we assume that they can be served as ground truth.

Whisper [11]: A short summary of the paper's main ideas and contributions needs to be added. Whisper 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 [5]: A short summary of the paper's main ideas and contributions needs to be added. Pyannote 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.

*** A few reference papers are related to including audio summarization? Are there any work similar to our work?

3 METHODS AND INITIAL ANALYSIS

3.1 System Architecture Overview

Our architecture can be separated into three parts:

(1) Speech to Text Conversion: Using the AMI and ICSI corpus, we transcribed the audio files using Whisper. Next, we diarized the audio file using Pyannote and combined the two files into a single transcription.
(2) Text Summarization: Using the human reference and the dataset that we have created we combined both to form a single JSON file. We then fed the JSON file into the pretrained BART model.
(3) User Interface: Using the Django Framework, we created a simple web application that allows users to upload an audio file and receive a transcription and a summary of the meeting.

The proposed pipeline is implemented in Python 3.8 using various tools. A complete list of tools we used as follows: BART [open source code cite?], FFMEG [4], Whisper Open AI [2], Pyannote [1], Hugging Face Transformers [open source code cite?], and Django [3] .

3.2 Speech to Text Conversion

Need to explain the methods for speech to text conversion

3.3 Text Summarization

3.3.1 BART. Our pretrained model is BART (Bidirectional Auto-Regressively Transformer). BART was developed by Facebook AI Research team. BART can handle various NLP tasks, for example, translation, text classification, summarization, etc. BART can be fine-tuned for any text generation and comprehension tasks. In short, BART is pretrained by corrupting the input text using some noise schemes and learning a model to reconstruct the original input.

BART has three big components: 1) noise transformation, 2) bidirectional encoder, and 3) autoregressive decoder. i. Noise transformation happens right before the encoder. Noise transformation will add some noises onto the inputs before feeding the data into the encoder. Having such action is to prevent the overfitting problem. There are five ways of corrupting the input: Token Masking, Sentence Permutation, Document Rotation, Token Deletion, and Text Infilling. The authors examined all of them and found that BART with text infilling could give the most consistently strong performance. ii. Next, the corrupted data will feed into the encoder that is able to read in both directions. This encoder produces a set of hidden states that capture the meaning and context of the input text. iii. Lastly, the set of hidden states will then get pushed to the autoregressive decoder that reads from left to right. This decoder basically generates a probability distribution over all possible tokens, given the hidden states and the masked input text. It then samples a token from the distribution and appends it to the output text in a step-by-step manner, where each token is conditioned on the previously generated tokens.

One reason why we choose BART is that it has richer documentation on how users can do fine- tuning and feed their own dataset to suit their own needs, compared to other novel pretrained models. Also, BART can establish abstractive summarization which is our objective. The evidence shown in the paper is highly abstractive with only a few phrases copied from the input.

3.3.2 Transfer Learning in BART. As mentioned above, BART hasn't been trained on any long conversation dataset that includes multiple speakers. We plan to feed some of the AMI and ICIS corpus to BART and utilize another portion of the data to validate the outcome.

We are using transfer learning to train our model. In transfer Learning we need select a baseline model which acts as the source model, and we fine-tuned using the dataset. For each of the experiments we evaluated the experiment outcomes using both qualitative and quantitative metrics which are discussed in the nest section.

For faster prototyping, we used the HuggingFace Trainer API which allows us to use train 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 can utilize the instant Inference API to use our model for demos.

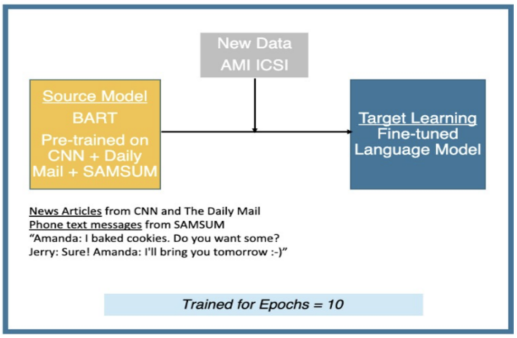

Figure 2: Transfer learning model

Notes: the figure will be replaced with a new TL

3.4 User Interface

The application consists of two pages, with the first page dedicated to transcription and the second to summarization. Starting with the first page, at the top of our user interface, we offer the browse and upload buttons for users to locate and upload their local files. A clear button is also provided to reset the current application state to the initial state, enhancing visibility.

Beneath these buttons, we offer a transcription box that contains the transcribed content from the audio file. The transcription box enables users to edit the transcribed content if they find it necessary to change a word, number, or sentence. The reason for providing such editing functionality is that the accuracy of converting audio to text heavily depends on the quality of the microphone input.

Currently, our system is unable to identify speaker names mentioned in any conversation for accurate speaker labeling. Therefore, below the transcription box, we provide a table with the first column displaying the speaker tags identified by our system, and the second column allowing users to change the corresponding speaker tags to names of their choice. To apply all changes made, a change button is provided next to the table. We also provide a download

link that allows users to download the updated transcript if they wish to save it locally.

At the bottom of our user interface, we incorporate an audio player and display 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.

Navigating to the second page of our application, users can read the summary of the updated transcript. To obtain the system-generated summary, users must click the "Get my summary" button to initiate the summarization process. After a certain amount of processing time, the summary will appear in the box located below the button. Lastly, keywords from the meeting are also provided. These keywords are extracted from the transcription box on the first page. 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 are offered to help users better comprehend the conversation.

Need to explain the methods for user interface & figures???. its above in black

4 EXPERIMENTS

4.1 Datasets

AMI corpus and ICSI corpus are publicly available datasets that contain 100 hours and 70 hours of meeting conversations recorded respectively. The duration of the audio files is between 20 minutes to 1 hour and 30 mins and there are 169 files in total. All the files are in .wav format. The audio files consist of recording of multiple speakers interacting in workplace meeting.

AMI Corpus [6, 9], Augmented Multi-party Interaction Project, funded by the European Commission. ICSI Corpus [7, 8], International Computer Science Institute Speech Group, UC Berkley. Reference Summary - Yale-LILY/QMSum: Dataset for NAACL 2021 paper: "QMSum: A New Benchmark for Query-based Multi-domain Meeting Summarization" (github.com)

We are also using reference summaries 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.

4.1.1 Data Preprocessing. The audio files are relatively clean but quite lengthy, each conversation is around 50 mins long. One difficulty that we are facing is the length of the audio files. We are relying on OpenAI's Whisper to transcribe the audio files and also using pyannote to diarize the sudio before combine the two to create the complete transcript. We used 1-GPU for carrying out the processing on the audio files. It takes approximately 3 mins. With 1-GPU to generate a transcript for a audio file that is 50 mins to 1 hour long.

4.1.2 Exploratory Data Analysis. Because we are dealing with NLP, we do not perform traditional data analysis. However, because the format of our dataset was not suitable for Summarization tasks, we had to perform data wrangling procedures.

i. Errors in Transcription using Whisper Due to the fact that we did not specify a specific language for the Whisper model, it was inferring languages from accents. This caused several errors in the transcriptions where Whisper switched to another language before reverting to English.

ii. Data Wrangling A typical dataset for summarization will have at least three columns: id, original text, and reference summary. However, since our dataset is a transcription of audio corpuses, we had to find reference human summaries. Fortunately, we were able to find reference summaries for the ICSI and AMI datasets.

Next, we had to combine all three features together so that they will be readable by HuggingFace's Dataset API. We created a Python function to combine the meeting id, dialogue, and reference summary together and wrote them to a single JSON file. In other words, each JSON object is an individual meeting.

4.2 Experiment Setup

4.2.1 System setup. The following are some of the experiments that we setup:

- Changing the baseline model: We attempted to use a baseline model without the pretraining on SAMSUM. However, the summary that the model produced was unsatisfactory for our purposes.
- Different number of training epochs: We attempted to increase the number of epochs to ten for the second model. However, after the first epoch, the validation loss continues to increase and our HuggingFace model automatically saved the model from the first epoch.
- Code is available in a public Github repository: https://github.com/vmarklynn/parrot. To setup the project environment please refer to *project_setup.md* file on Github which has instructions to setup all the dependencies for the proposed pipeline. Links to Models on Hugging Face are below:
  - https://huggingface.co/vmarklynn/bart-large-cnn-samsum-icsi-ami
  - https://huggingface.co/vmarklynn/bart-large-cnn-samsum-icsi-ami-v3

4.2.2 Performance Metrics. Our evaluations on the performance of models are based on two types of evaluation methods: (1) Gnerated Length (GEN Len) and (2) Recall-Oriented Understudy for Gisting Evaluations (ROUGE).

Gen Len measures the average length of the model-generated sequences. It is used to evaluate how well a language generation model can produce output of the desired length.

ROUGE compares automatically produced 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 doen by humans [reference]. ROUGE calculates three following metrics:

- Precision: how much of the model summary was relevant? Precision = Number of overlapping N-grams / Number of tokens in the generated Text
- Recall: how much of the reference summary is recovered from the reference? Recall = Number of Overlapping N-grams / Number of tokens in the reference Text
- F_1-score: it symmetrically represents both precision and recall in one metric. F_1-score = (2 * Precision * Recall) / (Precision + Recall)

In our analysis, we used three ROUGE methods:

(1) Rouge-N: This measures the overlap between the generated text and the reference text in terms of n-grams. To compute the Rouge-N score,
   i. Tokenize the generated text and the reference text into n-grams of size N.
   ii. Compute the number of overlapping n-grams between the generated text and the reference text.
   iii. Compute the precision, recall, and F1-score using the formulas above.

(2) Rouge-L: This measures the overlap between the generated text and the reference text in terms of longest common subsequences. To compute the Rouge-L score,
   i. Tokenize the generated text and the reference text into individual words.
   ii. Compute the longest common subsequence between the generated text and the reference text.
   iii. Compute the precision, recall, and F1-score using the following formulas:

(3) Rouge-Lsum: It computes the Rouge-L score for multiple sentences, and then averages the scores. In other words, 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.

A good Rouge-1 score typically falls between 40 and 50, while a Rouge-L score of around 30 or higher is generally considered good [ref]. However, 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.

4.3 Quantitative Analysis

!!! This section needs more work to analyze the results. Any other results for quantitative analysis?

In this section, we compare two BART models: (1) BART-A: It was trained on the CNN Daily Mail and SAMSUM dataset and retrained the model with our datasets using HuggingFace's Trainer API. This model is suitable for summarizing articles and short chat messages as it was trained. However, it may not produce good summaries for long dialogues [reference?]. Our quantitative results throughout the epochs are shown in the table below:

What is the difference between Modal-A and Model-B? The second model must have different training, right?

Need to explain the model performance results

Table 2 summarizes hyperparameters we used for the two models.

4.4 Qualitative Analysis

!!! This section needs more work to analyze the results.

This section presents qualitative results of our proposed pipeline using the BART-B model

We compared to the human reference summary. The output from our model is more comprehensive, accurately identifying the specific locations visited by the individuals. Summaries from BART-B and their corresponding summaries by humans are shown in Table 3 .

Our model covers about 60 percent 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.

We tested the limits of our model by giving input of the entire Lord of the Rings screenplay to summarize. It has been able to provide a reasonable summary but as one can see it makes a wrong inference that people should be more like the KINGS of Men when they shouldn't because the KINGS of Men become corrupted by power. We can also see that the model has learned typical workplace lingo such as "The meeting began ..." and "They discussed ..." due to the transfer learning and finetuning we have carried out.

5 DISCUSSION

We introduced a pipeline that transcribes, diarizes, and summarizes conversation meetings using Whisper, Pyannote, and a retrained BART model, resulting in a 148

However, our project currently faces limitations in terms of 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 data sets. This would enable the model to learn and adapt to various meeting scenarios, ultimately improving its accuracy and generalizability.

Future work should focus on making our application more marketable to companies by improving the summarization capabilities and providing accurate 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.

REFERENCES

[1] 2021. Pyannote Whisper. https://github.com/yinruiqing/pyannote-whisper..
[2] 2021. Whisper. https://github.com/openai/whisper..
[3] accessed on April 13, 2023. Django. https://www.djangoproject.com..
[4] accessed on April 13, 2023. FFmpeg. https://ffmpeg.org..
[5] Hervé Bredin, Ruiqing Yin, Juan Manuel Coria, Gregory Gelly, Pavel Korshunov, Marvin Lavechin, Diego Fustes, Hadrien Titeux, Wassim Bouaziz, and Marie-Philippe Gill. 2020. Pyannote. audio: neural building blocks for speaker diarization.

Table 1: Model performance comparison

Model	Epoch	Step	Training loss	Validation loss	Gen Len	Rouge-1	Rouge-2	Rouge-L	Rouge-Lsum
BART-A	1	20	No log	3.2095	108.503	39.817	11.556	24.030	36.305
	2	40	No log	3.1361	108.152	39.756	11.129	23.263	36.566
	3	60	No log	3.1599	122.951	41.790	12.097	23.534	37.686
	4	80	No log	3.2878	122.704	42.316	12.280	23.935	38.239
	5	100	No log	3.3671	129.233	40.797	10.734	22.943	36.438
BART-B	1	135	No log	3.2308	185.794	38.427	13.646	22.366	35.235
	2	270	No log	3.3026	146.853	40.175	11.794	23.266	36.472
	3	405	No log	3.5199	141.765	39.821	12.162	22.777	36.497
	4	540	2.2131	4.0508	131.441	40.433	11.655	22.996	36.878
	5	675	2.2131	4.6988	145.971	38.410	9.831	20.389	34.197
	6	810	2.2131	4.9590	169.245	38.576	9.634	20.865	35.032
	7	945	2.2131	5.4264	148.039	38.281	9.576	21.141	34.599
	8	1080	0.4010	5.4887	139.353	38.301	9.688	21.240	34.158
	9	1215	0.4010	5.8044	145.235	39.960	10.433	22.690	36.241
	10	1350	0.4010	5.8987	138.794	39.093	10.841	21.914	35.507

Table 2: Hyperparameters in model retraining

Model	learning rate	train batch size	eval batch size	seed	optimizer	LR scheduler type	# epochs
BART-A	5e-05	8	8	42	Adam, $\beta = (0.9, 0.999), \epsilon = 1e-8$	linear	5
BART-B	5e-05	1	1	42	Adam, $\beta = (0.9, 0.999), \epsilon = 1e-8$	linear	10

Table 3: Qualitive analysis

Type	Machine Summary	Human Reference Summary
Short prose/dialogue [ref]	The whole vacation was spent in Paris with my parents. **They saw the Eiffel Tower, Nortre-Dame Cathedral, the Seine River, and the famous art museums.** They also went a a boat tour around the city, and they had **a lot of fun sitting in cafes and watching people.**	#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 [ref]	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**.	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.
A meeting of Load of the Rings [ref]	Machine Summary (no human reference summary) **The meeting began with the forging of the Great RINGS, each set with a single GEN, which were given to the Dwarf Loards, great miners, and craftsmen of the mountain halls.** Next, the meeting moved onto a discussion of the design of the New Line Cinema's next project, which was due to be presented in the next meeting. Then, the group moved onto the discussion of how they would govern each other and their own lives. **They discusssed how they could make the best use of the power they had, and decided that they would have to be more like the KINGS of Men.** Finally, they talked about how to make the project a success. The meeting ended with a brief discussion of their next meeting	

[6] European Commission. accessed on April 13, 2023. AMI Corpus. https://groups.inf.ed.ac.uk/ami/corpus/..
[7] UC Berkley International Computer Science Institute Speech Group. accessed on April 13, 2023. ICSI Corpus. https://groups.inf.ed.ac.uk/ami/icsi/..
[8] Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, et al. 2003. The ICSI meeting corpus. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, Vol. 1. IEEE, I–I.
[9] Wessel Kraaij, Thomas Hain, Mike Lincoln, and Wilfried Post. 2005. The AMI meeting corpus. (2005).
[10] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. 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*. 7871–7880.
[11] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356* (2022).
[12] Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. 2021. QMSum: A new benchmark for query-based multi-domain meeting summarization. *arXiv preprint arXiv:2104.05938* (2021).