Controlled Hierarchical Summarization for Longform Spoken Dialog

ANONYMOUS AUTHOR(S)*

Every day we are surrounded by various forms of spoken dialog: interviews, debates, meetings, and conversations with friends and mentors. They are rich sources of information, yet it is difficult to access that information. They all invariably contain the shared aspect of auditorily delivering a rich stream of information. Despite such daily pervasiveness, automated speech content understanding and information extraction quality remains markedly poor, especially when compared to written prose. Furthermore, compared to text understanding, auditory communication poses many additional challenges such as speaker disfluencies, informal prose styles, and lack of structure. These concerns all demonstrate the need for a distinctly speech tailored interactive tool to help users understand and navigate the spoken language domain. While individual ASR and text summarization methods already exist, they are imperfect technologies; neither consider user purpose and intent nor address spoken language induced complications. Appropriately, we design a two stage automatic speech recognition (ASR) and text summarization pipeline and propose a set of semantic segmentation and merging algorithms to resolve speech modeling challenges. Our system corrects for errors in the underlying technologies and the accompanying interactive features allow users to easily browse and navigate records, as well as semantically extract relevant information. The system is evaluated in both formal and informal dialogue settings on a diverse range of topics and speech styles.

CCS Concepts: • **Human-centered computing → Human computer interaction (HCI)**.

Additional Key Words and Phrases: summarization, natural language interaction, automatic speech recognition, information retrieval, machine learning applications

ACM Reference Format:

Anonymous Author(s). 2018. Controlled Hierarchical Summarization for Longform Spoken Dialog. *ACM Trans. Graph.* 37, 4, Article 111 (August 2018), 21 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Every day we are surrounded by various forms of spoken dialog: interviews, debates, meetings, and conversations with friends and mentors. They are rich sources of information, but that information is difficult to access quickly. For example, YouTube has panel discussions on important topics such as COVID-19, diversity, and voting rights. Although people may be interested in a given topic, they may be unwilling to commit the required time necessary to consume long form media given uncertainty as to whether such media will live up to their expectations. There exists a clear need to provide access to spoken dialog information in a manner through which individuals can quickly and intuitively access areas of interest in these forms of media without spending large amounts of time.

An ideal solution would be to automatically summarize the content and distill it to the most interesting points, but this is problematic for three reasons. First, despite many advances in machine learning, Automatic Speech Recognition (ASR) and summarization are not yet mature enough to do this. Second, there is a question as to whether the ASR and summaries can be trusted to be accurate, especially in the presence of informal language, minimal structure and

Manuscript submitted to ACM

speech disfluencies. Third, what each user wants from a summary will differ based on their previous knowledge and experience on the subject matter – summaries are not one-size-fits all. This makes it difficult to provide training data for summaries that would be acceptable to a wide range of users, even if machine learning algorithms were perfectly accurate. We want to explore solutions that can leverage the strengths of machine learning, while overcoming many of its weaknesses.

We present a system that produces hierarchical summaries of spoken dialog that allow the user to browse and navigate the content to find things that are interesting to them. Hierarchical summarization allows users to first see a high level summary of the content and then drill into progressively longer and more detailed summaries - or listen to the raw audio itself. This approach addresses two key issues:

(1) It allows users to be in control of what information they read at a high level and what information they consume in greater detail.
(2) When machine learning (ML) models makes mistakes in ASR and summarization, users can quickly recover the ground truth.

Although the typical approach to creating automated summarization systems require training data that is difficult to obtain, our approach allows us to use previously trained ML models recursively to generate shorter and shorter summaries. However, reusing models that were trained on different data requires careful selection and use of existing trained ML models as well as novel algorithms to semantically segment the input text and thus output coherent summaries.

This paper makes the following contributions:

(1) A hierarchical summarization interface for browsing and navigating speech dialog to find interesting information.
(2) A novel segmentation algorithm that creates semantically similar input blocks to maintain conceptual cohesiveness
(3) A semantic hierarchical clustering algorithm that joins conceptually similar ideas for further abstract summarization
(4) A user study showing the semantic segmentation algorithm is 72% accurate in producing the shortest summaries. Using hierarchical features allowed users to correct 98% of summaries. Average time that users spent to understand the file was 27% of the time of the original audio.
(5) Qualitative findings about the how people use the system to to find interesting information in audio data.

2 RELATED WORK

We discuss the three main areas that our work builds on top of: (1) natural language processing (NLP) analysis of audio and video to generate key points and (2) summarization of multi-party audio such as meetings and podcasts and (3) state of the art ASR and summarization. We leverage several of the techniques used in both the user studies and the summarization work to create our system.

2.1 Using NLP to Generate Multimodal Interactions

Researchers have developed models and systems to easily navigate through videos and movies by navigating to the video clip and allowing users to interpret content [Barnes et al. 2010; Goldman et al. 2006; Jackson et al. 2013]. However, these videos require users to search visual information in a video they may know little about and is inapplicable to pure audio files. To solve these issues, some researchers have employed summarizing key content in text as a means

of helping users easily digest long-form content [Pavel et al. 2014], [Pavel et al. 2015]. More recent work has adapted the use of hierarchical information to provide users with multiple levels of summarization and information [Truong et al. 2021]. We build atop these systems targeting multi-party audio transcripts which pose novel challenges because these transcripts necessitate proper semantic segmentation to preserve meaning across speakers while simultaneously leveraging the usefulness of hierarchical information.

Still other work utilizes NLP to generate multimodal interactions such as images for video editing or even adding visuals to existing audio files [Xia 2020; Xia et al. 2020]. However, they rely on human-created transcripts, hurting the ability for the system to scale without automatic processes. Furthermore, visual representations only represent higher level abstract topics, not the summarizations needed to represent the speaker.

2.2 Summarization of Multi-Party Audio

Creating meaningful summarization from multi-party audio has been a difficult problem for researchers, often requiring hierarchical transformers and speaker segmentations to effectively retain information. Many of these papers, however, require full end-to-end training on transformers and even custom datasets [Karlbom and Clifton [n.d.]; Li et al. 2019; Vartakavi and Garg 2020; Zheng et al. 2020; Zhu et al. 2020]. Still others also employ graph-based summarization and coreferences to better summarize discourse [Xu et al. 2019]. Meanwhile, current unsupervised abstractive summarizations do not utilize deep learning summarization modules and require the use of word graphs and ranking algorithms [Shang et al. 2018]. These works focus on learning end-to-end summarization which is not practical across multiple domains. Instead, we focus on utilizing these summarization systems as part of a larger unsupervised abstractive system to generalize and reduce the overhead needed to deploy and scale such a system.

2.3 Automatic Speech Recognition and Abstractive Summarization

Automatic Speech Recognition systems (ASR) are used to transcribe audio (word recognition) into a source language transcript and have made relatively recent significant strides in terms of practical performance. Additionally, state of the art ASR [Google 2021] is no longer constrained by vocabulary and remains relatively robust, encouragingly extending word recognition to topical domains and noisy audio.

Text summarization techniques can be classified into two categories: abstractive and extractive. Abstractive summarization generates a new unique summary of text given a context whereas the extractive summarization "quotes" and concatenates relevant portions to compose into a summary. Because of spoken language noise effects in ASR transcripts, extracting transcript segments verbatim often leads to poor summaries. Therefore, we opt for the current state of the art abstractive summarization model, PEGASUS [Zhang et al. 2019], which is able to achieve much higher human-quality summaries. This is achieved by innovatively changing the pre-training process from standard word level masked language modeling, where models learn language conventions and syntax by predicting individually removed words within sentences, to sentence level masked language modeling, where entire sentences are removed and then recovered. This training process gives PEGASUS a high level of document understanding and helps to distill important information. Though promising, like most language models, it is important to note that PEGASUS is tailored towards specific benchmark datasets such as news or social media and that performance does not translate across different data domains, especially when applied to speech specific noise and disfluencies.

3 FORMATIVE STUDY

There has been much progress on machine learning models for natural language processing, including ASR and summarization. If possible, we want to use existing pretrained models as a component of our system to avoid the costly process of collecting longform summarized speech training data, as none readily exist or are available. This is particularly difficult for summarization because every user may want a slightly different summary. Moreover, there are two key problems:

(1) ASR and summarization models are far from perfect and have inherent pre-existing challenges.
(2) Summarization models are almost always trained on text rather than speech data. If a text trained summarization model is deployed on speech data, there would be a data domain mismatch, leading to considerable degraded model performance.

Compared to text, speech is far less structured - there are no topic sentences to rely on, speakers can stop mid sentence and backtrack their thought or never complete it, and coherency is challenging when multiple speakers making different points at the same time. Additionally, speech contains informal language and disfluencies such as hesitation and vocal filler words. These reasons heavily indicate that existing text-trained summarization models will perform very poorly on speech dialog.

To evaluate the practical performance of existing ASR and summarization models and determine which models to use as the basis of our system, we investigate the following criteria:

(1) *Coherency*, are the final output summaries coherent? If this constraint is not met, the model is not usable. Aside from re-training and adapting a model towards speech data, we have no tractable strategies for compelling model coherence.
(2) *Information retention*, because output summaries are shorter and lossy, we check if they still retain salient information from the original passage. If a shortened summary does not contain useful or relevant information, it has no value.

3.1 Evaluation Data

We evaluate seven recordings of longform spoken dialog that span different topics, domains, and speech styles (Table 1). Average length of the recordings is 32.5 minutes and the average word count output from ASR is 5622. Of these seven recordings, four are edited interviews from the NPR podcast "How I Built This", and 3 are unedited recordings from live events. Two are Bloomberg interviews regarding finance and one is a conversation about "How to foster true diversity and inclusion at work (and in your community)." In edited recordings, dialog has the potential to be more coherent and structured than in an unedited live session.

3.2 Models

For word recognition, we use a state of the art ASR model, publicly available with the Google Speech-to-Text API. This system is already robust to a variety of domains and speech noise, while providing features such as diarization (speaker detection) and punctuation prediction. While we suspect the ASR component will not be a large contributing factor to poor summarization, we conduct a brief investigation on word recognition errors (word errors, i.e. homonyms such as `weather` compared to `whether`) as they could non-trivially impact downstream summarization performance.

For summarization, we investigate the current abstractive state of the art language model PEGASUS. While PEGASUS is noticeably improved over other summarization methods in terms of producing human level quality summaries, it

Table 1. Dataset metadata used in formative study and final evaluation

Transcript Name	Length	Word Count	Source	Edited?
NPR: M. Night Shyamalan	48 minutes	9184 words	How I Built This podcast	Yes
NPR: Chipotle	48 minutes	7847 words	How I Built This podcast	Yes
NPR: Health	29 minutes	5102 words	How I Built This podcast	Yes
NPR: Teach for America	22 minutes	3909 words	How I Built This podcast	Yes
Diversity and Inclusion	23 minutes	4201 words	Recorded Ted Talk Interview	No
Bill Ackman on Economy	29 minutes	5140 word	Recorded Bloomberg TV Interview	No
Ray Dalio on Economy	29 minutes	3971 words	Recorded Bloomberg TV Interview	No

Table 2. Model nomenclature, training data descriptions, and model maximum input and typical output sizes.

Model Name	Domain and Fine-Tuning Data	Max Input	Output Size
M1, Model 1	XSUM, BBC News	64 words	1 sentence
M2, Model 2	News, CNN, DailyMail	128 words	3-5 sentences
M3, Model 3	Paraphrasing, Quora and Google PAWS	60 words	1 sentence

requires fine-tuning onto domain specific summarization data. It is also important to note that a pre-trained only instance of PEGASUS is not normally used without modification; the pre-training procedure is different from summarizing and the authors focus solely on fine-tuned downstream summarization datasets. Appropriately, we select fine-tuned instances from `huggingface.co` [Wolf et al. 2019] that generate complete and grammatically correct passages (i.e. not a few keywords) and are still in considerably general domains (i.e. not a medical field model instance) to assess PEGASUS coherence and information retention. Model details are given in Table 2.

We begin by processing audio files to obtain raw ASR transcripts. However, because of the nature of longform dialog, the number of words per transcript greatly exceeds the maximum input length that **`M1, M2, M3`** can accept. Transcripts must be processed and split into manageable lengths. We naively segment the transcript in fixed 60 word length segments, set to **`M3`**'s maximum input length[1]. For example, if an input transcript segment had a total of 154 words, it would be broken into a list of 3 individual segments, each containing $[60, 60, 34]$ words. To maintain evaluation consistency across all models, any evaluation involving naive fixed segmentation is set to 60 words. These are then summarized by **`M1`**, **`M2`**, and **`M3`**, which are set to output summaries containing at most half of the original passage's words.

3.3 Metrics

We evaluate coherency and information retention with state of the art automated metrics in natural language processing. For coherence evaluation, we use a `BERTScore` [Zhang et al. 2020] (given as an F-score) between a reference ASR segment and a model generated summary (candidate input). This method correlates well with human evaluation and uses word level contextualized embeddings to capture dependencies and word ordering. For retained information, we use the cosine similarity between `Sentence Transformer` [Reimers and Gurevych 2019] embeddings of a reference ASR segment and a model generated output summary. A higher cosine similarity between the reference ASR segment and output summary suggests the summary captures the reference ASR segment's semantic content. The final heuristic

[1]We also experimented with increasing the input size to 128 for **`M2`**, but still observed poor results (in fact, noise artifacts and incorrect model behaviors were more exaggerated than using 60 word length segments)

is the simple average of the two and has a range of $[-1, 1]$. In practice, cosine distance based metrics used to determine similarities between word embeddings are positive, with a general range of $0 - 0.5$ for a weak correlation, $0.5 - 0.8$ for a moderate correlation, $0.8 - 1$ for a strong correlation, and 1 for a perfect correlation [Jatnika et al. 2019]. As a sanity check, we observe a correlation of 1.0 when we set the reference and candidate text inputs to be the same. Intuitively, as **M1**, **M2**, and **M3** outputs are still summaries, they will contain at least some semantic similarity to the reference ASR segment; therefore we expect to observe a somewhat moderate correlation $(0.5 - 0.6)$ with our heuristic.

3.4 Results and Discussion

We discuss Table 3 throughout this section. It contains an example of the ASR transcript segment of one speaker in the "Diversity and Inclusion" recording and the corresponding summaries generated by the three models. Text is color coded to indicate shared regions between the ASR transcript and the summary.

Table 3. Example summary generations on the "Diversity and Inclusion" dialog showing an instance of the ASR transcript for one speaker turn and errors frequently made by the three summarization models. Text is color coded to indicate shared regions between the input and output.

ASR Transcript Segment	Summary	Description
opportunity is inclusion because you know, I have heard this stories so many times about there's you know, no Black Talent out there. No Latin next talent for these particular roles. The talent is out there. I will tell you that is underdeveloped because I think we have spent more time trying to reach numbers. Then we have changing our `[60 words]`	**M1**: *In our series of letters from African-American journalists, journalist and columnist Richard Lawson* looks at the importance of inclusion in the workplace.	**Hallucinations**, **M1** produces words and concepts (*underlined red*) in the summary are not present in the given input.
	M2: opportunity is inclusion because you know i have heard the stories so many times about theres no black talent out there no latin next talent. i think we have spent more time trying to	**Speech noise sensitivity**, **M2** loses abstractive summarization ability and outputs unimportant sections.
	M3: I think we have spent more time trying to reach numbers than we should because there is no black or latin talent out there.	**Semantic Inaccuracy**, **M3** combines two different sentences into a semantically incorrect output.

Here we briefly discuss the ASR quality. We quickly and quantitatively evaluate the word error rate (WER) of the ASR system. Because Table 1 only consists of audio data and perfect transcripts (human transcribed) are non-existent, we benchmark ASR performance on a random subset of Ted Talks as they are most similar in terms of speech and data structure to Table 1 and thus likely be indicative of ASR performance. We find an average WER of 10%[2], slightly above the reported 6.7% WER [Kim et al. 2019], and far below a usability constraint of 30% [Gaur et al. 2016].

As seen in the provided ASR Transcript example in Table 3, the ASR Speech-to-Text makes very few errors. However, rare words, unfamiliar phrases, or new words not yet encountered still degrade performance. For example, in the NPR "Chipotle" dialog, *"mise-en-place"* was mistakenly transcribed as *"knees in place"*. In the "Diversity and Inclusion" dialog, *"rectangle. Opening"* was mistakenly transcribed from *"reckoning"*.

Table 4 gives the automatic evaluation heuristic scores[3] for **M1**, **M2**, and **M3** ranging from $0.61 - 0.70$. Despite generating summaries on out of domain speech data, we can conclude that the baseline language models can still reasonably

[2]This number should be treated as an upper bound as the human transcribed transcripts contain artifacts such as `"(Applause)"` or `"(Laughter)"`.
[3]A later comparison against Table 4 with our full system using this heuristic is given in Section 5.2

Table 4. Automatic evaluation heuristic scores for various segmentation strategies.

Model Name	Segmentation Strategy	Heuristic Score
M1	Naive Fixed Length Segmentation	0.61
M2	Naive Fixed Length Segmentation	0.70
M3	Naive Fixed Length Segmentation	0.68

function and retain a moderate amount of information with a summary containing at most half the words as the input ASR segment. This observation merits further investigation into the re-usability of **M1**, **M2**, and **M3**. While our heuristic is telling, it still provides only a limited perspective into performance and is not a replacement for human evaluation. To get a sense of what types of errors the automatic summarization models are making and whether they could potentially be addressed, we studied various segments by hand.

It is important to note that although this style of evaluation was not formal; the errors were pronounced, ubiquitous, and immediately apparent. Such poor performance severely impeded practical usability and therefore did not necessitate a formal evaluation. Unfortunately, we observe that all three summarization models make frequent and substantial errors.

M1 is trained on news data and clearly hallucinates semi-related entities related to the text. For example, in the table M1 generates "African-American journalists" and "Richard Lawson." Neither of these entities are ever mentioned in the input (or entire audio file). This is a typical problem seen in language models [Maynez et al. 2020] when deployed on new data that is not encountered in training. Only recently, an attempt at fixing hallucinations has resulted in improved ROUGE precision and increased human preference [Zhao et al. 2020]. These errors are in almost every summary produced by **M1**.

M2 is also trained on news data, but trained to produce longer summaries. Unfortunately, it introduces many grammatical errors (i.e. sentences trail off without finishing making it difficult to read). Moreover, it fails to produce an abstractive summary and defaults to an extractive behavior; it has mostly picked a section of the input rather than trying to summarize the entire input. Reiterating Section 2.3, it is essential for a speech summarization model to be abstractive. These errors are frequently in summaries produced by **M2**.

M3 has more fluent text with no hallucinations. However, it makes an egregious error of misrepresenting the content. The transcript clearly states that ' *"The [Black and Latin] talent is out there,"* but the summary introduces a negation to say that the talent is not there. The root of this problem is that **M3** coerces two different segments into a semantically incorrect summary. These errors are egregious and occur when multiple non-sequitur or different topics are provided as a single input. Because abstractive summarization generates words that are not necessarily present in the source input text, they require a high degree of content understanding of the underlying semantic information in the passage [Gliwa et al. 2019] to successfully generate a semantically faithful summary; a poorly segmented input containing multiple different concepts would be exceedingly detrimental towards a model's semantic comprehension. Thus, **M3**'s resulting coherent and abstract summaries (albeit contextual misrepresentation errors) signal that:

(1) A successful semantically accurate segmentation that can group similar topics and ideas together, while splitting dissimilar sentences into a separate chunk can improve a model's semantic comprehension, and transitively improve summary generation accuracy. Additionally, the summary context accuracy issue is now rephrased as a tractable processing challenge that does not require training data.

(2) **M3** is able to maintain its abstractive nature, which is essential to summarizing dialog due to speech disfluencies and other noise artifacts.

Based on this exploration of three models, we hypothesize that **M3** is the best one to build on top of. **M1** and **M2** make too many errors to correct for without massive amounts of speech specialized training data. Although this is rarely the case in written language where ideas are groups in semantically coherent paragraphs, it is likely the norm in spoken language where topics shift over time as speakers react to the last thing that was said. If we can segment the transcript into semantically cohesive segments and summarize those, **M3** may be an effective summarization tool. When errors do remain, there is the fallback of the user using the hierarchical browsing features to investigate surprising or suspicious claims to see if the summary is consistent with the text.

4 SYSTEM

We present a system that produces hierarchical summaries of spoken dialog that allow the user to browse and navigate the content to find things that are interesting to them. Hierarchical summarizing allows users to first see a high level summary of the content and to then drill into progressively longer and more detailed summaries - or listen to the raw audio itself. As shown by our formative study, pre-existing technology performance drastically suffers when applied to speech and is still considerably below the requirement for practical usage. Therefore, in addition to the borrowed pre-existing ASR system (Google Speech-to-Text API) and language summarization model (**M3**, paraphrasing adapted PEGASUS), we develop a method to identify semantically related segments of text that can be input into the summarization model, then merged back together to maximize coherent summaries. This process can be done recursively to get increasingly shorter and abstractive summaries.

The core technical novelty and contributions within our speech summarization framework is as follows:

(1) A novel segmentation algorithm that creates semantically similar input blocks from an input ASR segment in order to maintain conceptual cohesiveness
(2) A semantic hierarchical clustering algorithm that joins conceptually similar ideas for logical subsequent (recurrent) abstract summarization

The inclusion of these procedures to the two-stage framework enables not only grammatical and semantic cohesiveness but also facilitates various levels of summarization detail:

(1) *Long Summary*: Cleaned ASR transcript. At this stage, a transcript's disfluencies and noises are cleaned and presented according to conversational order or speaker turns.
(2) *Medium Summary*: Moderately Detailed Summarization. Similar *Long Summaries* are merged and further paraphrased, providing key concepts along with essential details.
(3) *Short Summary*: High level Summarization. Similar *Medium Summaries* are further merged and provide the transcript's salient ideas.

4.1 Interface

The interface (Figure 1) consists of three main sections: *high level summary* column on the left, the *segment data view* in the middle and the *timeline of segments* at the top of the interface. Users explore the content by first browsing the *Short Summary* column to get a high level overview of the content. Users may click a *Short Summary* to see summaries of different length and additional levels of abstraction (*Medium and Long Summaries*) as well as the ASR transcript, or elect to listen to the corresponding audio. Yellow highlighting shows a key phrase in the short summary and it's

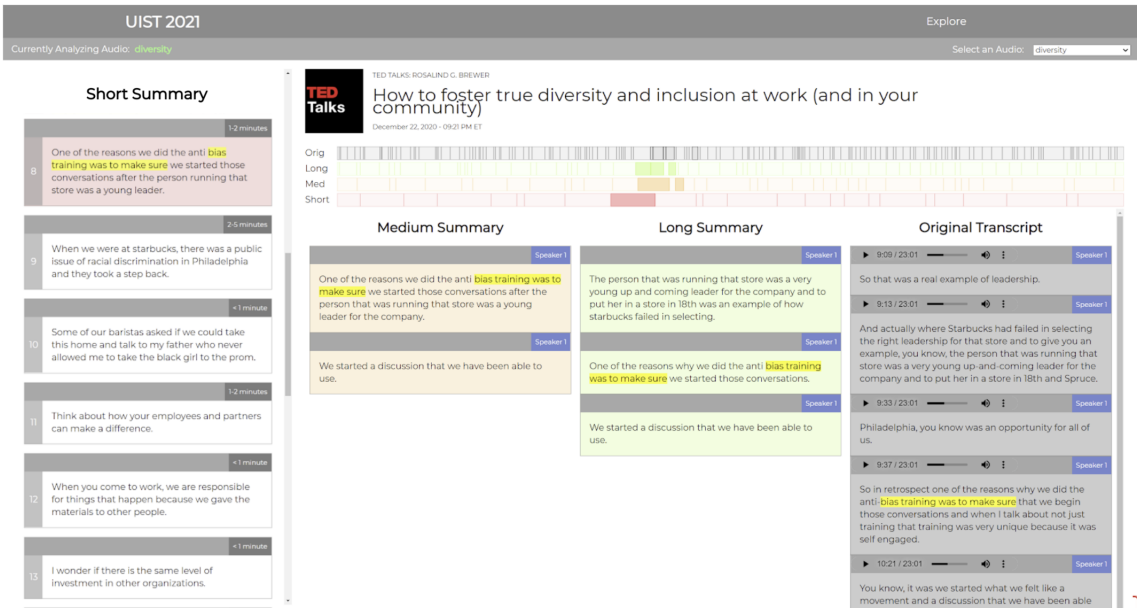

Fig. 1. **System User Interface.** Example of the system's summarization display for each unique audio file. The left part of the interface contains several short summaries which, when clicked, displays the medium and long summaries along with the corresponding original transcripts and audio clips. The top part of the interface shows what part of the transcript that each summary encapsulates information about. When the top part is clicked, users can navigate to any part of the summaries or transcripts.

corresponding phrase in the other summaries and ASR to help orient readers as they move from reading the short summary to the other summaries of the same underlying text. The timeline of segments shows how all the summaries are aligned. The user can see some *Short Summaries* that cover longer portions of the original transcript than others. Clicking on the timeline will take the user to the summaries of that section.

The interface was designed with two goals in mind:

4.1.1 Design Goal 1: Enable users to quickly identify useful information to them. Presenting high level summaries to the reader allows them to quickly grasp a general idea of what is being said. However, simply reading short summaries probably would not entirely satisfy the reader. By nature of being summaries, they may omit details that may be of interest. Additionally, the automatic summary algorithms are imperfect and sometimes present summaries that are more vague than a user would like them to be. However, the purpose of the *Short Summaries* are not necessarily to fully summarize, but to allow the user enough information scent [Pirolli 2007] to decide if they want more detail. If they want more detail, they can use the hierarchy of summaries to read *Medium* or *Long Summaries*, read the ASR transcript or listen to the underlying audio.

4.1.2 Design Goal 2: Support Error Correction and Recovery. As both automatic speech recognition and summarization may produce errors at various stages of the system, the interface provides multiple tiered layers of information for users to fall back on in order to recover comprehension of any given set of summarization data in the event that either a portion of the transcribed audio or summaries has erroneous text.

Listening to the raw audio will provide the full information a user should need to recover from an error. However, listening to audio takes longer for most people than reading text. If users want near-full fidelity information in a form they can read (or scan), they refer to the ASR transcript. Many people find transcripts of dialog difficult to read because of the informal language and speech disfluencies. Thus, users may find the *Long Summaries* easier to read - they retain almost all the information of the ASR transcript, but the text is cleaned up to remove these artifacts of speech. Users who want more actual summarization can refer to the *Medium Summary*.

By presenting users with these options for recovering from errors, users can decide how much time and effort they want to put into recovering from the error. However, there is a possibility that offering users multiple options could provide a negative experience by overwhelming them with options. Over time, we expect users will become familiar with the nature of each level of detail and get a sense of which option to select. This is an issue we address in the evaluation section.

4.2 Summarization Algorithm and Implementation

Figure 2 shows the steps through which an audio file is recurrently processed to obtain different levels of summarization (*Short, Medium, Long*). At a high level, the system segments an ASR transcript and iteratively summarizes previously combined conceptually similar segments to obtain increasingly abstract summaries while preserving semantic meaning.

Stage 1 (Fig. 2) of the system employs ASR to create a speaker diarized transcript of the input audio file. These speaker turns are further processed by a semantic segmentation algorithm which divides a given speaker turn into chunks of semantically related sentences. The now refined speaker turns are iteratively given as inputs into stage 2 (Fig. 2) of the system where processing and hierarchical automatic summarization occurs. After each speaker turn is individually summarized, its summaries are embedded which are then used to cluster sentences of the summary. Clusters are concatenated (merged) and shorter summaries, which generally contain little salient information, are stemmed. The first resulting summary of this iteration through the pipeline represents a *Long Summary*. This *Long Summary* is fed back into the the automatic summarization model and follows the same embedding, hierarchical clustering, and stemming steps once more to generate a *Medium Summary*. One further cycle using the *Medium Summary* yields a *Short Summary*.

Details of each of the system components are as follows:

Automatic Speech Recognition. We begin by using the Speech-To-Text Google API to obtain transcripts with speaker diarization and predicted punctuation for initial sentence boundaries. Speaker turns are alternating blocks of text separated by changes in speaker; they provide a very coarse starting point for transcript segmentation. Speaker turns frequently discuss multiple different ideas and may result in a long monologue before another speaker interjects.

Coreferenced Semantic Segmentation. To understand how coreference resolution [Soon et al. 2001] and referenced entities can be used to semantically link sentences together and correct poor segmentations, we first need to recognize a few spoken language tendencies. Unlike written prose, conversation can be far more vague; nouns and objects, herein referred to as entities, are only initially mentioned and then sporadically referenced, while all other mentions are pronouns (it, s/he, they, etc.). Identifying all of the expressions that refer to the same entity is known as the language task coreference resolution. We use the publicly available Allen NLP API [Gardner et al. 2018] for coreference resolution. To model this observation into a heuristic, our algorithm seeks to group sentences spanned by an entity, which is defined as the sentences contained by the start and end of the entity. We directly refer to variables in the pseudocode for Algorithm 1 in the following walk through of the process.

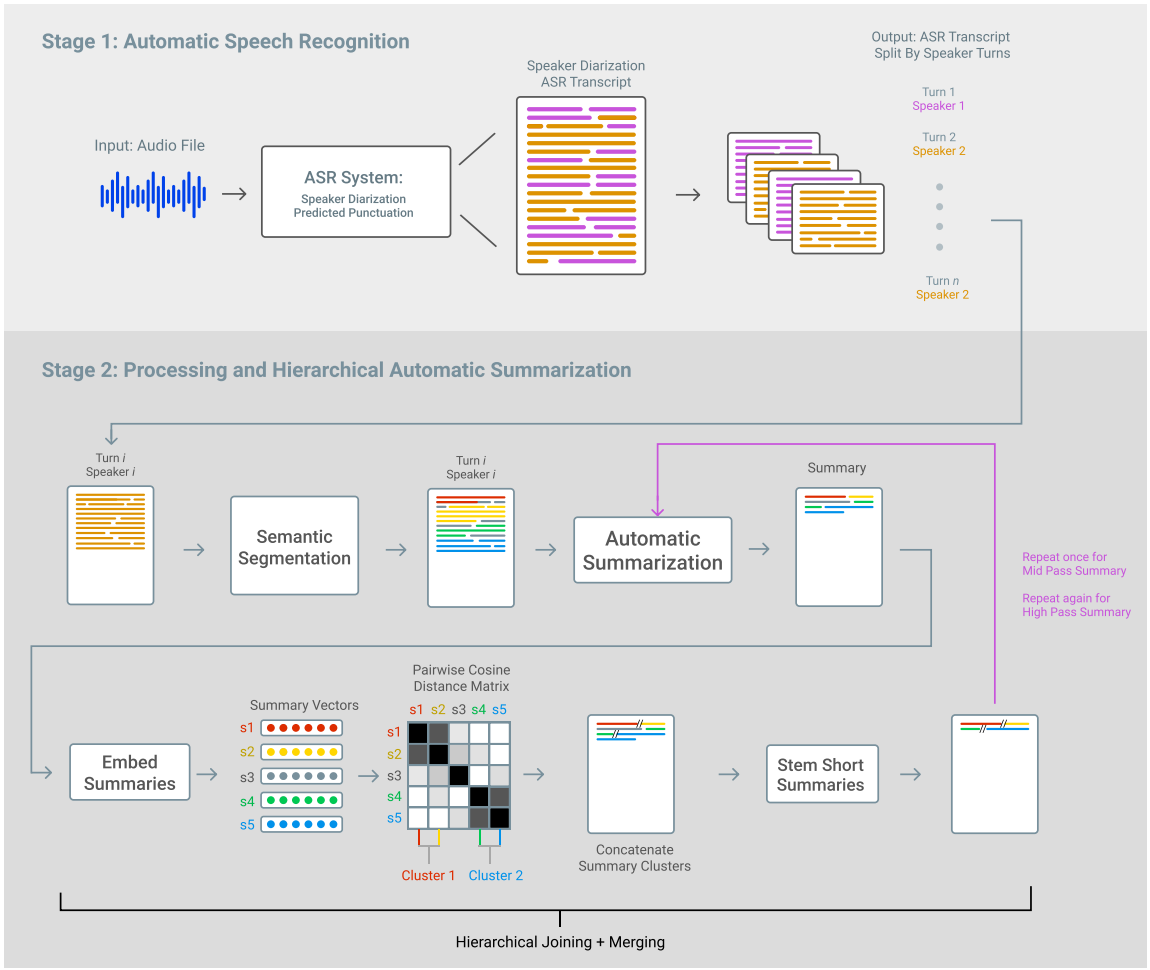

Fig. 2. **Summarization Generation Pipeline.** Our system enables the conversion of audio files to multiple tiers of summarization. In the first stage, we convert the audio file into a speaker-segmented and punctuated transcript and process the transcript, split by speaker turns. In the second stage, we take each speaker turn and cluster conceptually similar summaries via semantic segmentation. Each cluster's summaries are joined (concatenated) based off of semantic similarity. We remove small summarizations and then repeat the summarization and merging process to obtain the *Medium* and *Short* summaries.

We segment our speech transcripts using the linguistic hypotheses that a new speaker usually begins a relatively different concept and that repeated references to the same entity indicate the same concept is still being discussed. All iterative instances of the algorithm on each speaker turn segment S_i are independent of each other.

Here we walk through a single speaker turn pass of Algorithm 1. Speaker turn S_i contains sentences s that were previously divided by ASR predicted punctuation. For each sentence s, we first obtain the coreferenced entity c_{min} (`line 13`) that maximally spans the current sentence s and future consecutive sentences $s \in S_i$ (`lines 4:10`), subject to constraints. This denotes the start and end pointers (e_0, e_f) of the current semantically similar chunk, $\mathbf{P}$ (herein referred to as cluster, `line 13`). Sentences contained by $\mathbf{P}$'s span e_0, e_f, are assigned to $\mathbf{P}$ (`lines 19:20`). If sentence s contains

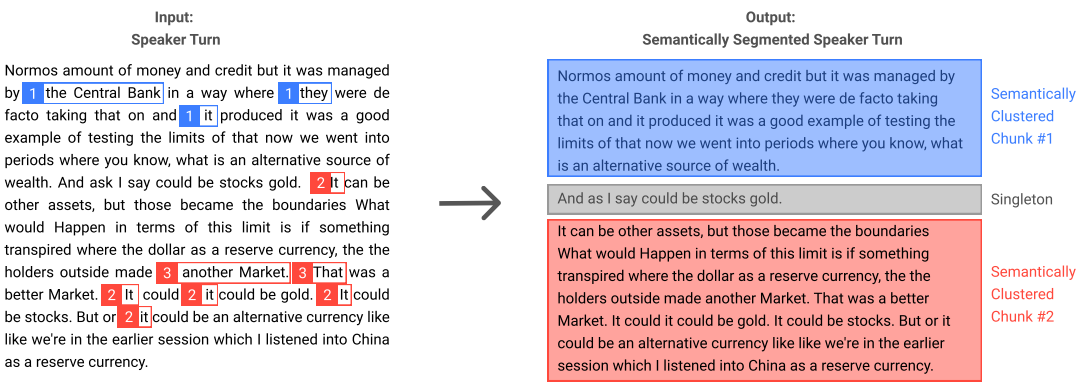

Fig. 3. **Alg 1. Semantic Segmentation** Example of one speaker turn input into our coreference resolution algorithm. On the left, the coreference tags generated from the AllenNLP coreference resolution module are shown with three different references highlighted. Our algorithm groups sentences with overlapping references into semantic chunks, shown in cluster 2, and then isolates sentences without any references into singletons.

another entity that spans further than **P**'s current end pointer, e_f is updated to the sequentially higher indexed sentence (`lines 21:23`). When **P**'s span is exhausted and cannot be further extended, the algorithm begins a new cluster **P**. Sentences that contain no entities are singleton clusters.

We restrict the entity span at most $m = 100$ words and require each valid span to contain at least $p = 3$ mentions (coreferences).[4] We also do not consider "`I`" and "`me`" entity references since these references do not indicate a semantic change. Figure 3 demonstrates an instance of the sentence entity spans procedure.

Summarization Model. We opt to reuse the paraphrasing **`M3`** instance of PEGASUS. The implementation of **`M3`** was taken from `huggingface.co`, using checkpoint `Tuner/007`. We also make the key observation that multiple recurrent forward passes of **`M3`** (independent of `Short, Medium, Long` heirarchical summarizations) removes speaker disfluencies and other speech artifacts of low importance.

The tradeoff for increased robustness towards speech noise and artifacts is also inherently found in **`M3`**'s paraphrasing nature; **`M3`** struggles to reason out semantically different ideas and suffers substantially from contextual errors (Table 3, **`M3`**). However, when ASR transcripts are preprocessed with Algorithm 1, our framework is able to generate not only cohesive and semantically logical summaries, but also achieve practical accuracy.

Hierarchical Concept Clustering and Merging. The next challenge is to determine which of the previous level's summaries to concatenate for further abstract summarization. Recall that semantically similar summaries must be joined or the model output can be factually incorrect (Table 3, **`M3`**). As a means to compare summary content similarity, we use `Sentence Transformer` to individually embed summaries $s \in \mathbf{S}$, into vectors in a semantic space. We then use cosine similarity to compute a pairwise distance matrix for hierarchical (agglomerative) clustering. Identified summary clusters are then concatenated. Pseudocode is given in Algorithm 2.

5 EVALUATION

We conducted user studies on our system's complete pipeline and quantitatively evaluate the following:

[4]We empirically observed that entities below these requirements had low relevance to the underlying concept.

(1) The language model's summarization performance for high level *Short Summaries*.
(2) The interface's affordances for recovering and correcting errors. Since the AI produced summaries that may still be imperfect, we must evaluate if the system's hierarchical nature and tiered presentation of information enables users to recover a particular audio segment's true meaning.
(3) The amount of time users saved by using our system when used in an unconstrained setting with their own browsing styles and comprehension goals.

Additionally, we present qualitative findings on how and when people would the system to find interesting information in spoken dialog.

5.1 Methodology

We recruited 10 recent graduates (denoted as $P_1, ..., P_{10}$ in a randomized order) from diverse professions (5 women, 5 men, average age = 26 with a range of 23 to 28) for our study. Each study lasted 1.2 to 2 hours and averaged 1.5 hours; subjects were paid $20 per hour for their time. To begin, participants were provided with a scenario where a dialog summarization tool would be potentially useful: *imagine you get an email from a friend or colleague about an exciting interview on YouTube about "Diversity and Inclusion in the Workplace." It's 23 minutes long, you're not sure if you want to commit to watching the whole thing, but you want to know if there's anything new or interesting in it. We're trying to help people explore audio clips to find key take-aways.* They were also informed that the summaries were generated by an AI and might be imperfect.

Participants were then given a link to the interface with the audio for "Diversity and Inclusion" loaded in. We explained the different UI components, *Short, Medium,* and *Long* levels of summarization, the original transcript (ASR transcription process and how it was obtained through Speech-to-Text), and the media button to play and scan the corresponding original audio section. Figure 1 is an example of what the user would see. We instructed the participant to read the first *Short Summary*, its corresponding *Medium Summary*, *Long Summary*, and ASR transcript segment, as well as to play the audio segment.

Participants were asked to perform two tasks:

(1) `Task 1`: evaluate *Short Summary* accuracy while diagnosing and correcting any errors for two audio files ("Diversity and Inclusion" and one of their choice). For `Task 1`, participants were asked to think aloud so we could understand how participants built an intuition and what their interpretation of the system was like. We asked participants to score each *Short Summary* for grammatical and syntactical accuracy as well as semantic comprehensibility (i.e. if they were able to understand the *Short Summary* and if it matched the corresponding audio segment's content) in a binary fashion (`"Y/N"`). Users were able to check for semantic meaning by comparing against *Medium, Long Summaries*, original ASR transcripts, and audio. If participants were confused on a *Short Summary*'s meaning, they were asked first, if they could recover the audio's meaning and second, if appropriate, what system features they used to recover the meaning. Participants were told to spend as much time as they deemed fit to comprehend the audio segment.
(2) `Task 2`: use the system as they would in an every day situation for one audio file of their choice. For `Task 2`, participants were asked to use the system without any restrictions and without thinking aloud. We timed participants' usage and observed their browsing strategies.

Finally, we concluded with an exit interview consisting of free form thoughts and subsequent pointed questions about their experience using the system.

5.2 Results

5.2.1 Short Summary accuracy. Individuals were asked to rank all the *Short Summaries* for the two recordings they saw in `Task 1` ($N = 556$ individually user scored individual short summary evaluations). Overall, these users rated accuracy of the *Short Summaries* to be 71.4% (see Table 5). The overall accuracy includes both grammar accuracy (84.9%) and semantic accuracy (75.9%). Of these, semantic accuracy is a better indicator for usability [Kaschak and Glenberg 2000].

Table 5. Participant scored *Short Summary* accuracies and standard deviations (SD). Total accuracy is a subset (intersection) of "Y" the grammar and semantic accuracy.

Criteria	Accuracy
Grammars	84.9 ± 5.1%
Semantics	75.9 ± 4.8%
Overall	**71.4 ± 4.9%**

As expected, the summaries are not perfect, however, they are definitely an improvement over the naive **`M1, M2, M3`** instances from the formative study. In running the same analysis for coherency and information retention, we see our model with semantic segmentation gets a heuristic score of 0.83 - a 18.8% improvement over the best naive model.

Table 6. Automatic evaluation heuristic scores for various segmentation strategies.

Model Name	Segmentation Strategy	Heuristic Score	% Improvement
`M1`	Naive Fixed Length Segmentation	0.61	36.7%
`M2`	Naive Fixed Length Segmentation	0.70	18.8%
`M3`	Naive Fixed Length Segmentation	0.68	21.4%
`M3`	**Semantic Segmentation**	**0.83**	-

The next evaluations will determine whether the the system can still help users achieve their goals despite these errors.

5.2.2 User Error Recovery. When a participant encounters a confusing *Short Summary*, we evaluate whether the interface allows them to recover from that error and regain comprehension. Out of all errors that required further investigation, participants were able to recover and understand the audio segment's content **92.9**% of the time. In other words, errors were unrecoverable **7.1**% of the time. Thus, the overall error rate recovery (given that users recover from 92.9%) percent of errors is 98.2%. Some participants noted that some of these errors may not be possible to recover from because the underlying audio content was difficult to understand or incoherent.

Table 7. Distribution of the heirarchical levels users explored in order to recover from an inaccurate *Short Summary*.

Required Hierarchical Traversal Level	Fraction (%) Used
Medium Summary	11.5%
Long Summary	19.2%
ASR Transcript Segment	40.8%
Audio Segment	20.8%
Querying Neighboring Segments	6.9%

Across evaluations, participants relied on five sources (levels) of information available from the interface to recover from comprehension errors (Table 7): *Medium Summary*, *Long Summary*, Original (ASR) Transcript Segment, Audio Segment, Neighboring Segments Across Summarization Levels (using the full suite of features available in addition to surrounding contextual information). Participants either hierarchically traversed down the tiered information flow or skipped to their preferred source of information. The hierarchical traversal levels indicate incrementally increased time costs to a user; *Medium Summaries* would be the cheapest due to its shorter text and closer proximity to the *Short Summary*, and playing an audio segment or querying local neighboring segments and using the full set of features would be the most expensive.

5.2.3 Time Savings. In `Task 2`, when participants used the system at their own pace (without thinking aloud or rating tasks) and on a recording of their own choice, we found that users were able to gain valuable information out of the dialog in much less time that it would have taken to listen to it. The average time participants spent was **27.1**% of the original audio time to reach a level of understanding that they were content with. All participants spent less time using the system compared to the audio time. A breakdown of users along with their self-declared style of browsing (i.e. how they would use the system themselves in an unmonitored setting) on the system is given Table 8.

Table 8. Individual participant times on using the system. Further details including intended usage,

Participant	Self-Declared Style Browsing	Percent (%) of Audio Time
P_1	Detailed	75.9%
P_2	Cursory	12.1%
P_3	Cursory	14.9%
P_4	Cursory	20.7%
P_5	Cursory	20.8%
P_6	Cursory	6.9%
P_7	Cursory	25.9%
P_8	Moderately Detailed	24.7%
P_9	Cursory	24.3%
P_{10}	Moderately Detailed	45.5%
$P_{average}$	-	27.2%

Interestingly, P_1 noted a use case that we had not anticipated; the system was conducive towards learning new concepts and reinforcing the new material. P_1 noted the ability to play audio and follow along with an ASR transcript which was particularly conducive towards his audio-visual learning style. He spent a comparatively long time with the system (75.9% of audio time), but he chose to do so to get information out of it. More on users learning and interaction styles and it's effect on their usage of the system is in the qualitative findings section.

5.2.4 Qualitative Evaluation. The participants were generally enthusiastic about the system. P_1 liked that *"You can scan a 20 minute interview in a minute and find the parts you find interesting"* P_5 characterized it as *"Automatic Sparknotes for boring talks."* And P_2 said *"I would get this as a youtube plugin"* Users did note the flaws in the automated tools. P_5 said *"Something ... to polish it a bit more would be great"* However, they still found it usable and many user liked how the information was broken up and presented. P_3 said *"[It] structures information well by length and complexity and lent itself to how I read, I read in multiple passes"*

As expected, users treated the short summaries less as summaries and more as navigational tools. P_1 said *"At first I tried reading it linearly, but I found that I could use the short summaries as subheadings and individual bullet points"*. That user had a strong finance background and has selected a finance interview to browse. There were two key points where he sought more information than the short summary contained. first, "I found this as a thesis statement and read more into it" and second, *"[Mentioning he played tennis] was sort of an interesting factoid so I went into the audio and listened to it"*

It was common for users to rely on their own knowledge to guide them towards exploring more in detail. P_5 works in the medical field and was intrigued a short summary in the Health dialog: *"There was a spike in demand for specific types of nurses."*. She wanted to know what those specific types were. The medium summary did not contain that information but the long summary did - ICU nurses and ED (emergency department) nurses were the two specialties named.

The hierarchical features were more useful for some dialogs than for others. 8 of 10 users of the Diversity dialog did not mention reading medium and long summaries, and always went to the transcript. However, 5 of 5 readers of Ray Dalio dialog used the medium summaries. This is likely do to the nature of the underlying audio and the quality of the summaries. Ray Dalio tends to speak in paragraphs and the ASR transcript for each segment was long, but the medium summaries were a more attractive length. The medium and long summaries of the Diversity dialog did not add enough information, so users read the transcript to get the additional details they wanted. Since these factors are difficult to control for, the solution of presenting summaries at multiple granularities was successful.

6 DISCUSSION & CONCLUSION

Longform spoken dialog is a rich source of information that people encounter every day. However, for people who lack the time or inclination to listen to that audio, the information is not easily accessible. Progress in machine learning models for ASR and summarization give hope that such data accessibility can be made easier to access, however fully automated approaches are still out of reach. Current state of the art ASR and summarization models still present several sources of error ranging from word recognition errors and failures in handling speech disfluencies during speech recognition, to the introduction of hallucinations, grammatical errors, and misrepresented contexts during summarization. As a result, we present an end-to-end pipeline and user interface which provides real world practical mitigation of these sources of inaccuracy through an intuitive data exploration interface which enables hierarchical information presentation.

During our user study, evaluation errors from both ASR and summarization steps were detected resulting in the need for users to regain comprehension. Unexpectedly, users did in fact experience comprehension issues due to ASR errors, but were able to successfully recover understanding by utilizing the original audio present on the interface. In other cases, users were able to regain comprehension and recover from summarization errors due to the hierarchical presentation of summaries by the interface. For example, when a *Short Summary* was generated from an erroneous recurrent summarization due to the merging of semantically dissimilar *Medium Summary* segments, users quickly recovered by simply reading the individual *Medium Summary* segments. These forms of error recovery which users intuitively discovered, while using our interface, highlight the value of our system as well as the hierarchical summarization approach as a whole.

A limitation of this system occurs when summaries lack any indication that they contain errors as users may not realize a correction is needed. Often errors are obvious due to to the surrounding context or the reader's prior knowledge of the topic. However, there is certainly potential for non obvious errors to mislead readers. This evaluation did not address this limitation. An important next step is to study whether or not our summarization algorithm system contains

these non-obvious errors, and if so, how frequent they are. Depending on the nature and frequency of these errors, future research could be done to correct them automatically or to help readers detect and correct them.

REFERENCES

Connelly Barnes, Dan B Goldman, Eli Shechtman, and Adam Finkelstein. 2010. Video tapestries with continuous temporal zoom. In *ACM SIGGRAPH 2010 papers*. 1–9.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640* (2018).

Yashesh Gaur, Walter S Lasecki, Florian Metze, and Jeffrey P Bigham. 2016. The effects of automatic speech recognition quality on human transcription latency. In *Proceedings of the 13th Web for All Conference*. 1–8.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. SAMSum Corpus: A Human-annotated Dialogue Dataset for Abstractive Summarization. *Proceedings of the 2nd Workshop on New Frontiers in Summarization* (2019). https://doi.org/10.18653/v1/d19-5409

Dan B Goldman, Brian Curless, David Salesin, and Steven M Seitz. 2006. Schematic storyboarding for video visualization and editing. *Acm transactions on graphics (tog)* 25, 3 (2006), 862–871.

Google. 2021. Speech-to-Text. https://cloud.google.com/speech-to-text/

Dan Jackson, James Nicholson, Gerrit Stoeckigt, Rebecca Wrobel, Anja Thieme, and Patrick Olivier. 2013. Panopticon: A parallel video overview system. In *proceedings of the 26th annual ACM symposium on User interface software and technology*. 123–130.

Derry Jatnika, Moch Arif Bijaksana, and Arie Ardiyanti Suryani. 2019. Word2vec model analysis for semantic similarities in english words. *Procedia Computer Science* 157 (2019), 160–167.

Hannes Karlbom and Ann Clifton. [n.d.]. Abstractive Podcast Summarization using BART with Longformer attention. ([n. d.]).

Michael P Kaschak and Arthur M Glenberg. 2000. Constructing meaning: The role of affordances and grammatical constructions in sentence comprehension. *Journal of memory and language* 43, 3 (2000), 508–529.

Joshua Y. Kim, Chunfeng Liu, Rafael A. Calvo, Kathryn McCabe, Silas C. R. Taylor, Björn W. Schuller, and Kaihang Wu. 2019. A Comparison of Online Automatic Speech Recognition Systems and the Nonverbal Responses to Unintelligible Speech. arXiv:1904.12403 [cs.SD]

Manling Li, Lingyu Zhang, Heng Ji, and Richard J Radke. 2019. Keep meeting summaries on topic: Abstractive multi-modal meeting summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2190–2196.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On Faithfulness and Factuality in Abstractive Summarization. arXiv:2005.00661 [cs.CL]

Amy Pavel, Dan B Goldman, Björn Hartmann, and Maneesh Agrawala. 2015. Sceneskim: Searching and browsing movies using synchronized captions, scripts and plot summaries. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 181–190.

Amy Pavel, Colorado Reed, Björn Hartmann, and Maneesh Agrawala. 2014. Video digests: a browsable, skimmable format for informational lecture videos.. In *UIST*, Vol. 10. Citeseer, 2642918–2647400.

Peter. Pirolli. 2007. *Information foraging theory : adaptive interaction with information / Peter Pirolli.* Oxford University Press New York. 204 p. : pages. http://www.loc.gov/catdir/toc/ecip0617/2006021795.html

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *CoRR* abs/1908.10084 (2019). arXiv:1908.10084 http://arxiv.org/abs/1908.10084

Guokan Shang, Wensi Ding, Zekun Zhang, Antoine Jean-Pierre Tixier, Polykarpos Meladianos, Michalis Vazirgiannis, and Jean-Pierre Lorré. 2018. Unsupervised abstractive meeting summarization with multi-sentence compression and budgeted submodular maximization. *arXiv preprint arXiv:1805.05271* (2018).

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics* 27, 4 (2001), 521–544.

Anh Truong, Peggy Chi, David Salesin, Irfan Essa, and Maneesh Agrawala. 2021. Automatic Generation of Two-Level Hierarchical Tutorials from Instructional Makeup Videos. (2021).

Aneesh Vartakavi and Amanmeet Garg. 2020. PodSumm–Podcast Audio Summarization. *arXiv preprint arXiv:2009.10315* (2020).

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *CoRR* abs/1910.03771 (2019). arXiv:1910.03771 http://arxiv.org/abs/1910.03771

Haijun Xia. 2020. Crosspower: Bridging Graphics and Linguistics. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 722–734.

Haijun Xia, Jennifer Jacobs, and Maneesh Agrawala. 2020. Crosscast: Adding Visuals to Audio Travel Podcasts. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 735–746.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019. Discourse-aware neural extractive text summarization. *arXiv preprint arXiv:1910.14142* (2019).

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. *CoRR* abs/1912.08777 (2019). arXiv:1912.08777 http://arxiv.org/abs/1912.08777

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. arXiv:1904.09675 [cs.CL]

Zheng Zhao, Shay B Cohen, and Bonnie Webber. 2020. Reducing Quantity Hallucinations in Abstractive Summarization. *arXiv preprint arXiv:2009.13312* (2020).

Chujie Zheng, Kunpeng Zhang, Harry Jiannan Wang, and Ling Fan. 2020. A Two-Phase Approach for Abstractive Podcast Summarization. arXiv:2011.08291 [cs.CL]

Chenguang Zhu, Ruochen Xu, Michael Zeng, and Xuedong Huang. 2020. A hierarchical network for abstractive meeting summarization with cross-domain pretraining. *arXiv preprint arXiv:2004.02016* (2020).

A ALGORITHM PSEUDOCODE

Algorithm 1: Semantic Co-reference Segmentation

Input: List $\mathbf{T}_{in}$ of n speaker turn segments $\{S\}_{i=0}^{n}$ where word $w \in$ sentence $s \in S$, hyperparameters $m = 100$ for maximum coreference word span and $p = 3$ for minimum number of coreference mentions.

$\mathbf{T}_{out} \leftarrow$ list() *# all semantically segmented speaker turns*;
Initialize e_0, e_f *# coreference span start and end pointers*;
Initialize $\mathbf{B} \leftarrow$ list(*"I", "me"*) *# stop tokens*;
for $i = 0, 1, ..., n$ **do**
    $\mathbf{C} \leftarrow$ Coreference(S_i) *# Allen NLP API*;
    for *coreference entity* $c \in \mathbf{C}$ **do**
        if $c.span > m$ *and* $|c| < p$ **then**
            delete c from $\mathbf{C}$;
        if $c \subseteq \mathbf{B}$ **then**
            delete c from $\mathbf{C}$;
    $\mathbf{S}_{new} \leftarrow$ list() *# instantiate new speaker turn block*;
    $\mathbf{P} \leftarrow$ list() *# semantic topic cluster within speaker turn* $\mathbf{S}_{new}$;
    $e_0, e_f \leftarrow c_{min}$ with $min(w_0 \in c \in \mathbf{C})$ *# entity with earliest word index*;
    $\mathbf{C}_{used} \leftarrow$ list(c_{min});
    for $s \in S_i$ **do**
        $s_0, s_f \leftarrow w_0, w_f \in s$ *# start and end word indices of* s;
        for *coreference entity* $c \in \mathbf{C}$ *and* $c \notin \mathbf{C}_{used}$ **do**
            $c_0, c_f \leftarrow w_0, w_f \in c$ *# start and end word indices entity* c;
            if $s_0 \leq e_0$ *and* $e_f > s_f$ **then**
                P.append(s) *# s within span, add to semantic block*;
                if $c_f > e_f$ *and* $s_0 \leq c_0 \leq s_f$ **then**
                    $e_0, e_f \leftarrow c_0, c_f$ *# update maximal entity span*;
                    $\mathbf{C}_{used}$.append(c);
                break;
            else
                $\mathbf{S}_{new}$.append(**P**) *# add topic cluster to speaker block*;
                $\mathbf{P} \leftarrow$ list(s) *# begin new topic cluster*;
                $e_0, e_f \leftarrow c$ with $min(w_0 \in c \in \mathbf{C})$ and $c \notin \mathbf{C}_{used}$;
                $\mathbf{C}_{used}$.append(c);
                break ;
    $\mathbf{T}_{out}$.append($\mathbf{S}_{new}$) *# add semantically segmented speaker*;
return $\mathbf{T}_{out}$ *# all semantically segmented speaker turns*

Algorithm 2: Hierarchical Concept Clustering and Merging

Input: List S of summaries (sentences) $\{s\}_{i=0}^{n}$, Embedding Sentence Transformer Model M

$\mathbf{E} \leftarrow M.embed(\mathbf{S})$ *# embed all summaries*;

$\mathbf{D} \leftarrow$ pairwise_cosine_distance$(\mathbf{E}, \mathbf{E})$;

$labels \leftarrow$ agglomerative_clustering$(\mathbf{D})$;

$L_{out} \leftarrow s \in \mathbf{S}$ grouped by $labels$;

return L_{out} containing sentences concatenated by cluster label