Podcast Summarization: Bridging Audio and Text

Ameya Ranade
University of Massachusetts Amherst
aranade@umass.edu

Riya Adsul
University of Massachusetts Amherst
radsul@umass.edu

1. Introduction

Podcasts have witnessed rapid growth as a versatile medium encompassing news, commentary, entertainment, and educational content. While some podcast shows adhere to a predictable release schedule, others follow an irregular pattern. Episode duration also vary significantly, ranging from brief 5-minute segments on specific topics to lengthy, in-depth discussions that can extend beyond 3 hours. The nature of podcasts varies, encompassing news delivery, conversational exchanges, and storytelling. For a faster access to the content of a podcast, it can be useful to produce a summary of it before the user decides to play the podcast. But efficiently searching the actual content of podcast episodes remains a challenge. Current methods consists of producing a summary which is a subset of sentences of the original transcript (extractive summarization). However, these descriptions often fall short in capturing the core content of each episode.

The objective of this project is to provide concise textual summaries using semantic self segmentation and abstractive summarization of transcript data that enables users to efficiently grasp the podcast's content and understand its key topics. These summaries should be well-structured, self-contained statements, considerably shorter than the original episode descriptions, and contextually relevant.

2. Problem Statement

The project aims to generate high-quality abstractive summaries of podcast transcripts from the Spotify Podcast Dataset, which contains over 100,000 transcribed podcast episodes. The key challenges include automatic transcription errors, disfluencies, redundancies, and the variable nature of podcast content, making traditional summarization methods less effective. The documents are also longer than typical summarization data.

2.1. Existing methodologies

Existing summarization methods, such as recursive summarization, have limitations. They split the text into chunks without considering the logical and structural flow of the content, and they cannot account for the varying importance of different sections. This also leads to a loss of context. Another method involves passing each chunk, along with the summary from previous chunks, through a language model. The summary is progressively refined as more text is processed. However, since it is sequential, it cannot be parallelized, takes linear time, and may over-represent the initial parts of the podcast in the final summary.

To address these challenges, a more effective approach is proposed through this project. This approach aims to capture the hierarchy of meaning within the text, ensuring that the resulting summary accurately conveys the author's intended meaning.

2.2. Length of transcripts

Another issue we observed while analysing the dataset was that the spotify podcast dataset is way longer than the regular summarization texts. Due to this, the density of meaningful information in the dataset is less. Since the spotify transcripts are so long, the normal chunking approach gives us around 90/100 segments and aproximately 6000 tokens. However, the abstractive summarization model implemented (BART), is limited to only 1024 tokens. This issue is addressed in the paper. We have compared various ways of chunking the dataset like normal chunking, logical segmentation and semantic self segmentation and concluded on which of these can give us a better result.

3. Related Work

The PodSumm approach predominantly relies on linear extractive summarization [9] based on sentence-level segmentation and selection. We examine the utilization of Automatic Speech Recognition (ASR) that generates speech-to-text transcripts. For text summary generation, the research builds summaries from transcript by selecting appropriate sentences. The research also goes on further to generate audio representing the selected sentences, to obtain a final audio summary [6].

To address the limitation of choosing 1024 tokens for chunking, Zheng C. et al. [8] employed a two-phase process, selecting crucial sentences from the transcript. There are two main approaches in this process, a sliding-window ROGUE-based approach and Topic-enhanced approach. The paper then employed an encoder-decoder network for abstractive summarization. Their selection process is based on ROUGE scores, emphasizing sentences with higher ROUGE scores as significant summaries encapsulating essential information from the input.

In contrast, Song K. et al. [5] explored a segment-based approach rather than sentence-based extraction, dividing transcripts into 30-second audio segments—33 from the beginning and 7 from the end. They trained a classifier to discern whether a segment is salient or not. However, this method might separate closely related sentences into different segments, posing a challenge.

4. Data

Dataset: For our project, we will be using a subset of the Spotify Podcast data set. The data set contains over 200,000 podcasts with 100,000 hours of audio and over 1 billion transcribed words. The episodes span a variety of lengths, topics, styles, and qualities. The podcast has both English and Portuguese summaries; we will only be using the EN folder.

We will be primarily dealing with the transcript files provided for each podcast. The transcript files are in JSON format. The transcripts were created using Google Cloud Platform's Speech-to-Text API and may contain some expected errors in representing the audio content. On average, these transcripts consist of approximately 6,000 words, with variations ranging from a few exceptionally brief episodes to as long as 45,000 words. The majority of the transcripts fall within the range of 1,000 to 10,000 words, making up about two-thirds of the dataset. Only a small fraction, approximately 1%, comprises very short trailers designed for promoting other content. [1].

5. Approach

Our project draws inspiration from the referenced research paper, adopting a similar podcast summarization approach while implementing distinct algorithms. We also deviate from the paper by utilizing an alternative method to summarize text transcripts [7].

5.1. Phase 1

In the first phase of our approach, we prioritize the creation of robust training labels. We commence by conducting in-depth pre-processing of the creator-provided descriptions, encompassing activities like description cleaning, episode selection, and integration with the gold dataset which consists of summaries for a select few episodes. This process culminates in the extraction of the best episode summaries, which subsequently serve as the target description set, essentially forming the foundation for our model's training labels. These refined descriptions are then integrated with the corresponding podcast transcripts, resulting in the formation of a comprehensive training dataset. This dataset provides the essential groundwork for our subsequent steps in the summarization process.

5.2. Phase 2

In the second phase, we focus on filtering out the long podcast transcripts by converting them into semantic chunks and conducting abstractive summarization using BART to yield contextually enriched summaries for the diverse range of podcast transcripts present in the Spotify Podcast Dataset.

6. Implementation

In our pursuit of generating high-quality abstractive summaries for podcast transcripts from the Spotify Podcast Dataset, we have followed a systematic methodology and have so far achieved the following milestones:

6.1. Exploratory Data Analysis

We initiated our process by performing an exploratory data analysis on the metadata file containing information about the podcasts. This included details like show and episode names, episode descriptions, and other relevant data.

6.2. Gold Dataset Integration

We incorporated a gold dataset present in the data provided by Spotify consisting of multiple sets of summaries for selected episode, graded on a scale of Bad/Fair/Good/Excellent. We iterate over the podcast metadata and substitute the episode descriptions with the best summaries from the gold set for each corresponding episode that exists in both datasets. These summaries were merged with the metadata set (replacing a few episode descriptions), and the best summary for each episode was selected.

6.3. Episode Description Quality Enhancement

To train a supervised model, we considered the creator-generated descriptions as one of our training labels. However, these descriptions vary in quality and do not always serve as ideal summaries. Therefore, we took steps to clean and refine the episode descriptions to filter out useless sentences and symbols, ensuring that the subset chosen for reference summaries was more suitable. To improve the quality of creator descriptions, we applied heuristics to identify and remove sentences containing improper content. This pre-processing step involved removing content following ”—,” sentences with URLs, @mentions, email addresses, and emojis.

We used a metric based on word IDF scores to eliminate sentences with low relevance. To achieve this, we calculate a salience score for each sentence in the description by adding up the word IDF scores. We then filter out sentences with salience scores below a specified threshold. After experimentation, we determined that a threshold of 3.6 works effectively. In this context, a low IDF score indicates that the word appears frequently in other episode descriptions and is therefore considered less significant.

6.4. Data Preparation

Given the large number of episodes, we filtered out episodes based on heuristics mentioned in a

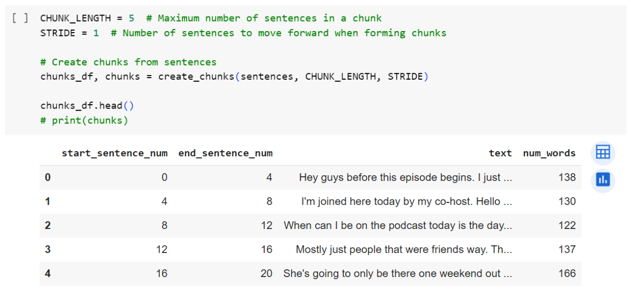
	Episode Description
Original	Danielle and Jessi could talk your ears off when it comes to this topic. Episode 004 is all about their skincare routines, products they love, and tips and tricks for feeling radiant and confident in your own skin. Follow them basically-organicpodcast (and jessimechler itsdaniellebridges) for tags of all the brands they're currently loving! Rate and subscribe!! – Support this podcast: https://anchor.fm/basically
After cleaning	Danielle and Jessi could talk your ears off when it comes to this topic. Episode 004 is all about their skincare routines, products they love, and tips and tricks for feeling radiant and confident in your own skin.

Table 1. Preprocessing epsiode description text

relevant paper [2]. We excluded descriptions that were too long or too short, those with high lexical overlap with show descriptions, and those with high lexical overlap with other episode descriptions in the same show. The intermediate dataset resulting from the above steps was combined with the corresponding episode transcripts to create our training dataset. This dataset is now ready for further processing.

6.5. Logical chunking

Instead of creating large chunks, we adopted a logical segmentation approach. Our chunk size corresponds to the number of sentences required to express a discrete idea, typically set at 5 sentences with a 1-sentence overlap to ensure continuity and contextual information. But this process gave an output of approximately 6000 tokens where as the abstractive summarizer model BART that we are going to use in the next step required a maximum of 1024 tokens. So we need to find another way to filter these transcripts.



```
[ ] CHUNK_LENGTH = 5  # Maximum number of sentences in a chunk
    STRIDE = 1  # Number of sentences to move forward when forming chunks

    # Create chunks from sentences
    chunks_df, chunks = create_chunks(sentences, CHUNK_LENGTH, STRIDE)

    chunks_df.head()
    # print(chunks)

       start_sentence_num  end_sentence_num                                              text  num_words
    0                   0                 4    Hey guys before this episode begins. I just ...        138
    1                   4                 8     I'm joined here today by my co-host. Hello ...        130
    2                   8                12  When can I be on the podcast today is the day...        122
    3                  12                16       Mostly just people that were friends way. Th...        137
    4                  16                20  She's going to only be there one weekend out ...        166
```

Figure 1. Chunks created for the show_filename_prefix 'show_600u26rsWE63UsQl6oCZDW'

6.6. Semantic Segmentation and Filtered transcript generation

The process begins by loading the training dataset and the spotify transcripts dataset. since the transcripts are long documents, they are split into semantically related chunks using a method referred to as "semantic self-segmentation (Se3) [4]." The semantic_segmentation function implements this segmentation, breaking down the transcript into chunks of sentences based on semantic relationships. The function uses a sentence transformer model for encoding sentences and a specified lower and upper chunk size. Another function, extract_features, is defined to extract features from each chunk. It uses a pretrained Sentence Transformer model to encode each sentence in a chunk and computes the mean of these embeddings to obtain a dense encoding for the entire chunk.

6.6.1 Chunk Classification

We then define a function isChunkUseful to determine if a chunk is useful for summarization. It uses a metric called ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation - Longest

Common Subsequence) to evaluate the summary quality. The dataset is processed to create a chunk classification dataset. Chunks are labeled as useful (1) or not useful (0) based on the ROUGE-L score compared to the reference summary.

6.6.2 Neural Network for Chunk Classification

A neural network model is defined using Keras for automating the classification of chunks as useful or not. The model architecture consists of several dense layers with activation functions like ReLU and Sigmoid.A balancing technique is applied to ensure an equal number of positive and negative samples in the training set.

6.6.3 Transcript Filtering

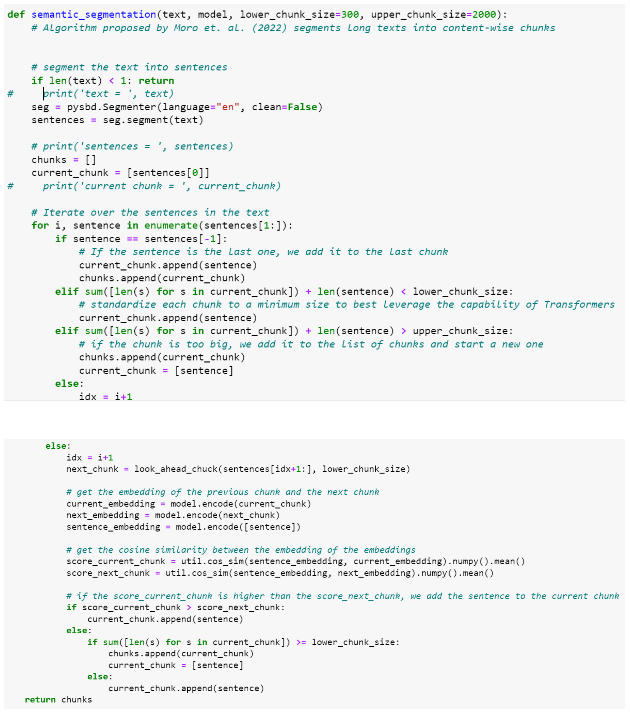
We then define a function called transcript_filtering that takes an episode, the chunk classifier model, a sentence encoder, and a BART tokenizer as input.The function uses semantic segmentation to obtain chunks from the transcript. It then uses the chunk classifier to predict the usefulness of each chunk.Chunks are sorted based on their relevance scores, and a subset of chunks is selected such that the total token count does not exceed 1024.The selected chunks are concatenated to form a filtered transcript.

6.6.4 Creating the new filtered set

The chunk classifier model is loaded along with the BART tokenizer and the Sentence Transformer model. The transcript_filtering function is applied to each episode in the training dataset, resulting in a new column containing the filtered transcripts used for abstractive summarization.

6.7. Abstractive Summarization

Under the domain of abstractive summarization BART, or Bidirectional and Auto-Regressive Transformers, stands out as a state-of-the-art



```
def semantic_segmentation(text, model, lower_chunk_size=300, upper_chunk_size=2000):
    # Algorithm proposed by Moro et. al. (2022) segments long texts into content-wise chunks


    # segment the text into sentences
    if len(text) < 1: return
#     print('text = ', text)
    seg = pysbd.Segmenter(language="en", clean=False)
    sentences = seg.segment(text)

    # print('sentences = ', sentences)
    chunks = []
    current_chunk = [sentences[0]]
#     print('current chunk = ', current_chunk)

    # Iterate over the sentences in the text
    for i, sentence in enumerate(sentences[1:]):
        if sentence == sentences[-1]:
            # If the sentence is the last one, we add it to the last chunk
            current_chunk.append(sentence)
            chunks.append(current_chunk)
        elif sum([len(s) for s in current_chunk]) + len(sentence) < lower_chunk_size:
            # standardize each chunk to a minimum size to best leverage the capability of Transformers
            current_chunk.append(sentence)
        elif sum([len(s) for s in current_chunk]) + len(sentence) > upper_chunk_size:
            # if the chunk is too big, we add it to the list of chunks and start a new one
            chunks.append(current_chunk)
            current_chunk = [sentence]
        else:
            idx = i+1
```

```
        else:
            idx = i+1
            next_chunk = look_ahead_chuck(sentences[idx+1:], lower_chunk_size)

            # get the embedding of the previous chunk and the next chunk
            current_embedding = model.encode(current_chunk)
            next_embedding = model.encode(next_chunk)
            sentence_embedding = model.encode([sentence])

            # get the cosine similarity between the embedding of the embeddings
            score_current_chunk = util.cos_sim(sentence_embedding, current_embedding).numpy().mean()
            score_next_chunk = util.cos_sim(sentence_embedding, next_embedding).numpy().mean()

            # if the score_current_chunk is higher than the score_next_chunk, we add the sentence to the current chunk
            if score_current_chunk > score_next_chunk:
                current_chunk.append(sentence)
            else:
                if sum([len(s) for s in current_chunk]) >= lower_chunk_size:
                    chunks.append(current_chunk)
                    current_chunk = [sentence]
                else:
                    current_chunk.append(sentence)
    return chunks
```

Figure 2. Screenshots of semantic segmntation implemented in the code

model that has proven its prowess in various Natural Language Processing (NLP) tasks, with a particular emphasis on text summarization and adopts a denoising autoencoder strategy during pretraining for sequence-to-sequence tasks. Denoising autoencoders serve the purpose of preventing the network from simply memorizing training data, a common pitfall in standard autoencoders where the output mirrors the input without substantial learning. BART addresses this by intentionally corrupting data, introducing noise, and training the model to reconstruct the original text. Its architecture follows a standard transformer-based seq2seq approach, incorporating a bidirectional encoder and a left-to-right decoder. Notably, the encoder's attention mask is fully visible, while the decoder's attention mask is causal, ensuring a progressive generation of output. The pre-training task consists of two stages: deliberate text corruption followed by training the seq2seq model to reconstruct the initial text. The objective during this task is to optimize the negative log-likelihood of the original document. BART's uniqueness lies in its flexibility, extending its applicability to various tasks beyond pretraining. While the original paper cites applications in sequence classification, token classification, sequence generation, and machine translation, this experiment explores its efficacy in abstractive summarization. In our case, the BART baseline underwent fine-tuning for third epochs using training dataset as input, deviating from the conventional approach of considering only the first 1024 tokens of the transcript. The training parameters included a batch size of 2, an Adam optimizer with a learning rate of 2e-6, and a weight decay rate of 0.01. The optimization focused on BART's internal loss computation.

7. Evaluation and Results

Expected Results: High-quality abstractive text summary that the user might read when deciding whether to listen to a podcast. Thus the summary should accurately convey the content of the podcast, and be short enough to quickly read on a smartphone screen. It should also be human-readable.

7.1. Qualitative Evaluation

- Readability and Coherence: See how the summary flows and whether it captures the main points and maintains the context.
- Content Coverage: We can check if the summary covers all the main themes and points and doesn't miss important details of the podcast.

7.2. Quantitative Evaluation

For evaluating the model's performance on the test set consisting of 1,027 podcast episodes from the Spotify Podcast Dataset, two key metrics have been selected: ROUGE[3] and BERT score. ROUGE is chosen as a metric to assess the generated summaries against the reference summaries provided by the creators. BERT score is chosen as a semantic metric for evaluating the results. It assesses the similarity between each token in the candidate summary and the reference summary using contextual word embedding.The combination of ROUGE and BERT score allows for a comprehensive evaluation of the model's performance. ROUGE assesses syntactic similarity, which is essential for summarization tasks, but may unfairly penalize abstractive models. BERT score, with its semantic focus and IDF weighting, complements ROUGE by capturing the nuances of meaning and rewarding summaries that convey the same message using different phrasing. This evaluation strategy ensures that the summarization model is assessed from both a syntactic and semantic perspective, providing a well-rounded assessment of its performance on podcast transcript summarization.

7.3. Result and Analysis

After training the loss for the train and validation is shown in the diagram below.

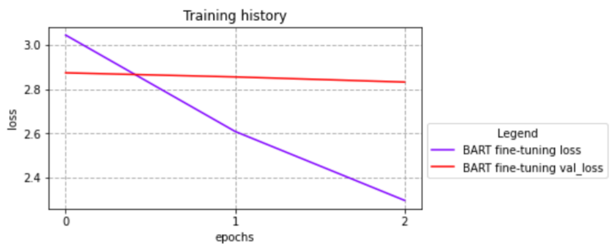
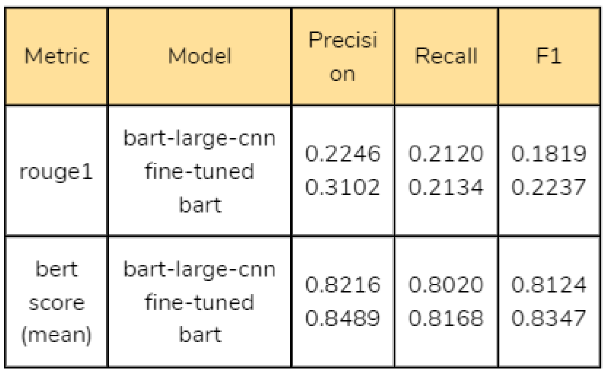


Figure 3. loss

After training the loss for the train and validation is shown in the diagram below.

The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores evaluate the quality of text summaries generated by models compared to human-written references. Higher ROUGE scores indicate better performance in terms of precision, recall, and F1 scores.

In summary, the fine-tuned model exhibits enhancements in various ROUGE metrics compared to the base pretrained model, particularly in unigram and sentence-level overlaps. However, there is still scope for improvement in capturing bigram overlaps and achieving higher scores



Metric	Model	Precision	Recall	F1
rouge1	bart-large-cnn fine-tuned bart	0.2246 0.3102	0.2120 0.2134	0.1819 0.2237
bert score (mean)	bart-large-cnn fine-tuned bart	0.8216 0.8489	0.8020 0.8168	0.8124 0.8347

Figure 4. scores

overall. These findings suggest the effectiveness of fine-tuning in improving summary generation quality, but further refinements might be necessary for achieving more substantial improvements in certain aspects of summarization.

The BERTScore evaluation measures the similarity between model-generated summaries and human references, providing insights into the quality and closeness of generated text to the ground truth references.

In summary, the BERTScore evaluation without IDF weighting demonstrates that the fine-tuned model outperforms the base pretrained model, showcasing higher similarity and closeness to the human references. These findings suggest the efficacy of fine-tuning in enhancing the summarization quality of the model by capturing more nuanced details and context from the input texts, resulting in improved summary generation.

Below we have shown the summary generated by our model on a spotify podcast. Three outputs are shown here- creator provided descriptions, pre-trained model and the fine tuned model.

Link to the video submission:

Video submission

	Summaries
Creator provided description	In our debut episode for our Traveller rpg actual play campaign our heroes meet and (after a jump drive failure) crash land. Tune in and join our adventure! Our Imperium actual play games use the Traveller RPG (Mongoose 2nd Edition). Background music and ambient sounds provided by Tabletopaudio.com. Intro music is Stock Media provided by Pond5.
Fine-tuned model prediction	In this episode, the crew set off on their first mission to Zemus, a system in the Concordian sub-sector of space. They find themselves in the midst of a fight with a gas giant, and the crew must figure out what to do about it.
Pre-trained model prediction	I play Reuben Glaser a harpoon, which is a dog race are and get used to that. So there's no recap to do because this is our first episode ever, but we're going to start off actually at Zemus. So yeah, I'm the type of guy that has like, you know, 50 windows open. I'm like moving in between them.

Table 2. Summaries obtained from our model

References

[1] Ann Clifton, Sravana Reddy, Yongze Yu, Aasish Pappu, Rezvaneh Rezapour, Hamed Bonab, Maria Eskevich, Gareth Jones, Jussi Karlgren, Ben Carterette, and Rosie Jones. 100,000 podcasts: A spoken English document corpus. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5903–5917, Barcelona, Spain (Online), 2020. International Committee on Computational Linguistics.

[2] Rosie Jones, Ben Carterette, Ann Clifton, Maria Eskevich, Gareth JF Jones, Jussi Karlgren, Aasish Pappu, Sravana Reddy, and Yongze Yu. Trec 2020 podcasts track overview. *arXiv preprint arXiv:2103.15953*, 2021.

[3] Chin-Yew Lin. ROUGE: A package for auto-

matic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, 2004. Association for Computational Linguistics.

[4] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[5] Kaiqiang Song, Chen Li, Xiaoyang Wang, Dong Yu, and Fei Liu. Automatic summarization of open-domain podcast episodes, 2020.

[6] Damiano Spina, Johanne R. Trippas, Lawrence Cavedon, and Mark Sanderson. Extracting audio summaries to support effective spoken document search. *J. Assoc. Inf. Sci. Technol.*, 68(9): 2101–2115, 2017.

[7] Isaac Tham. Summarize podcast transcripts and long texts better with nlp and ai, 2023. Accessed on October 14, 2023.

[8] Chujie Zheng, Kunpeng Zhang, Harry Jiannan Wang, and Ling Fan. A two-phase approach for abstractive podcast summarization, 2020.

[9] Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. Extractive summarization as text matching, 2020.