NORTHEASTERN UNIVERSITY

Automated Podcast Summarization and Highlight Generation

CS 6220 - Natural Language Processing

Aakash Singhal(singhal.aak@northeastern.edu)

Marin Witherspoon(witherspoon.m@northeastern.edu)

Meagan Dyer(dyer.mea@northeastern.edu)

Parth Malik(malik.p@northeastern.edu)

Abstract

This report outlines the development and evaluation of an innovative podcast summarization system designed to convert audio content into concise, informative summaries with key highlights. The project began with thorough preprocessing of audio data, followed by advanced audio-to-text conversion techniques, ensuring accurate transcription. Utilizing state-of-the-art models, the system efficiently processes the transcribed text to produce summaries and extract significant highlights, addressing the growing need for quick content digestion in the digital age. An extensive evaluation, incorporating rigorous metrics, demonstrates the system's effectiveness in generating coherent and relevant summaries. This work not only advances podcast summarization technology but also opens avenues for future research and development in multimedia content analysis and summarization.

1 Introduction

In the rapidly expanding world of digital content, podcasts have emerged as a significant medium for entertainment, education, and information dissemination. With millions of episodes available across various genres, listeners are often overwhelmed by the sheer volume of content. This challenge has created a pressing need for technologies that can efficiently distill these audio resources into more accessible formats. Our project, focusing on the development of an advanced podcast summarization system, addresses this need by providing a means to convert lengthy audio episodes into concise, informative summaries accompanied by key highlights.

The utility of such a system extends beyond mere convenience. For educators and students, it offers a quick way to glean insights from educational podcasts without dedicating hours to listening. Business professionals can stay updated with the latest industry trends and insights without disrupting their busy schedules. Casual listeners, on the other hand, can easily decide which episodes to invest their time in, based on the summaries and highlights provided.

Our approach leverages cutting-edge audio processing and natural language processing (NLP) techniques to ensure high accuracy in transcription and relevance in summaries. By automating the conversion of audio to text and then distilling the essence of the content, we aim to make podcast content more accessible and digestible for a wider audience. This project not only contributes to the field of multimedia content analysis but also offers a practical solution to the information overload problem in the digital age. Through rigorous development and evaluation, we have demonstrated the potential of our system to transform the way we interact with podcast content, making it a valuable tool for content creators and consumers alike.

2 Problem Statement and Data Acquisition

In the burgeoning landscape of digital audio content, podcasts such as those produced by Joe Rogan, Andrew Huberman, This American Life (TAL), and Vox have emerged as cornerstones of cultural, educational, and informational exchange. Each of these platforms brings its unique style, audience, and content diversity, ranging from deep-dive interviews and storytelling to analytical discussions on current events. While the rich variety of these podcasts represents a treasure trove of knowledge and insights, it also poses significant challenges for listeners attempting to navigate this wealth of information efficiently.

The primary problem lies in the accessibility and manageability of content given the extensive duration of episodes, which often exceed several hours. This situation is further compounded by the diverse nature of the content, which varies not only in subject matter but also in complexity and presentation style. As a result, potential listeners, especially those with limited time, may find it daunting to engage with these podcasts, leading to a significant gap between available content and consumed knowledge.

2.1 Types Of Datasets Used

The primary dataset used in the project comes from podcast transcripts that have been compiled from different podcasts such as; Joe Rogan, Ben Shapiro, This American Life, and Andrew Huberman. This dataset, named *'final_df_raw.csv',* contains textual representations of podcast episodes. Each entry in the dataset includes metadata such as episode title, description, and the episode's transcript. This dataset serves as the foundation for text analysis tasks, including sentiment analysis, topic modeling, and natural language processing.

With the incorporation of audio processing functionalities, the project involves the utilization of audio data extracted from podcast episodes. The *'download_and_preprocess_audio'* function retrieves audio content from RSS Feed's of podcasts, converts MP3 files to WAV format, and preprocesses the audio data for transcription. This audio data complements the textual transcripts, enabling multimodal analysis and exploration of podcast content.

The project encompasses both textual and audio datasets, combining podcast transcripts with corresponding audio recordings. By leveraging these diverse data modalities, the project aims to achieve a comprehensive understanding of podcast content, enabling advanced analyses and insights generation. The integration of audio processing capabilities enriches the analytical scope, facilitating tasks such as speech-to-text transcription and audio-based feature extraction alongside traditional text-based analyses.

2.2 Data Acquisition and Creation of Datasets

This section delves into the methodologies employed for pulling and preprocessing the podcast datasets, shedding light on the specific problems encountered during the process.

The podcast dataset, sourced from rss feeds and transcripts of multiple podcast episodes, was acquired to facilitate comprehensive analysis of podcast content. The dataset, named 'final_df_raw.csv', constituted the primary resource for generation.

An integral aspect of preprocessing involved the removal of timestamps embedded within the podcast transcripts. However, this task proved to be notably challenging due to several factors:

- Variability in Formats: The time stamps exhibited multiple formats, ranging from "00:00:00" to "00:00", causing adaptable parsing mechanisms to accommodate the variability.
- Inconsistent Placement: Time stamps were positioned irregularly within the transcripts, appearing at varying intervals and locations due to the timing of the podcast's discussions, which complicated their identification and extraction.
- Overlap with Textual Content: Time stamps occasionally overlap with meaningful textual content, causing greater validation to prevent inadvertent deletion of relevant information.

To address the challenges encountered, the following mitigation strategies were employed; Custom Parsing Logic in which tailored parsing algorithms were developed to accurately identify and remove timestamps, accommodating the diverse formats and placements observed in the transcripts.

Also, manual inspection and validation using human oversight was leveraged to manually inspect and validate the timestamp removal process, and to ensure the integrity of the preprocessed data.

The code also utilized stemming to reduce words to their root forms, which aided in text normalization. However, stemming can have some limitations including; Over-Stemming and Under-Stemming since stemming algorithms like Porter Stemmer may sometimes over-stem (e.g., 'universe' becoming 'univers') or under-stem (e.g., 'meeting' remaining unchanged), leading to inaccuracies in word representations. Also, Loss of the Semantic Meaning of the words can occur since stemming operates purely on linguistic rules without considering semantic context. As a result, these stemmed words may lose their original semantic meaning, potentially affecting downstream analyses that rely on precise word representations. The code saves the updated dataset to a CSV file after preprocessing. This helps to ensure that the processed data is preserved although there were considerations for data storage and management including; File Format Considerations which depending on the size and nature of the dataset, alternative storage formats such as Parquet or HDF5 may offer better performance and space efficiency compared to CSV. Also, Versioning and Backup because implementing versioning and regular backup procedures for datasets is crucial for data integrity, especially in collaborative or production environments. Addressing these considerations helped to enhance the reliability, and usability of the data acquisition and preprocessing work for the project, contributing to more effective downstream analyses and insights.

3 Transcribing Podcast Audio to Text

The process of converting audio content from podcasts into textual transcriptions is a critical component of this project. The primary challenge addressed in this section is the development and implementation of an audio transcription algorithm capable of producing accurate transcriptions, which are essential for subsequent natural language processing tasks. The algorithm's effectiveness is evaluated based on its ability to handle various issues, such as inaccuracies in timestamping and the limitations imposed by computational resources.

3.1 Model Overview

The transcription process leverages a pre-trained model, specifically the facebook/wav2vec2-base-960h from the Wav2Vec2 series developed by Facebook AI. This model is renowned for its performance in speech recognition tasks and is designed to convert audio signals into textual representations. By utilizing a pre-trained model, the need for extensive training and fine-tuning is circumvented, allowing for direct application to the audio files. This approach not only saves time but also ensures that the transcription process benefits from the advanced capabilities of a model trained on a diverse and extensive dataset.

3.2 Methodology

The methodology employed in transcribing audio files involves several steps, beginning with the preparation of data. The process starts with the loading of a CSV file, ‘final_df_with_cleaned_transcripts.csv’, which contains metadata about the podcast episodes, including the URLs of the audio files. These URLs are then processed to remove any extraneous characters or parameters that might hinder the transcription process.

Once the URLs are cleaned, the audio files are accessed and converted into a format suitable for transcription. This conversion is crucial as it ensures compatibility with the transcription model and maintains the integrity of the audio data. The transcription process itself is facilitated by the Wav2Vec2Processor and Wav2Vec2ForCTC from the transformers library, which work in tandem to convert the audio signals into textual transcriptions. To manage computational resources effectively, the audio files are processed in batches. This approach optimizes memory and processing power and allows for parallel processing of multiple audio files, thereby accelerating the transcription process.

The transcribed text is subsequently added to the DataFrame as a new column, GENERATED_COLUMN_TRANSCRIPT, which serves as the basis for further analysis and processing. The batch processing of audio files is a critical aspect of the methodology, as it enables the handling of large datasets without overwhelming the computational resources. By processing audio files in segments or batches of a fixed duration, such as 3 minutes, the algorithm can maintain a balance between efficiency and accuracy.

3.3 Challenges

The development of the transcription algorithm was not without challenges. One of the primary issues encountered was the intense computational demands of the initial model, which necessitated a revision of the approach to ensure feasibility. This challenge was addressed by implementing batch processing and reducing the size of the dataset to three podcasts, thereby optimizing the use of computational resources.

Another significant challenge was the large size of the audio files, which posed difficulties in downloading and processing. To overcome this, the audio files were converted to a more manageable format and processed in smaller segments, allowing for efficient handling without compromising the quality of the transcription.

In conclusion, the transcription of audio content from podcasts into text is a complex process that requires careful consideration of various factors, including the choice of model, the format of the audio files, and the management of computational resources. The methodology outlined in this section provides a comprehensive approach to addressing these challenges, resulting in accurate and reliable transcriptions that form the foundation for subsequent analysis and processing.

4. Summary Generation of the Transcribed Audio

The transcriptions generated from the podcasts's audios, contained grammatical and spelling errors, as imposed by the speech to audio transformer. In addition, the absence of punctuation in all scripts posed new challenges for generating summaries.

Sample of Transcribed Text:
"WELL IT'S THE END OF THE ROAD FOR NICKEY HALEY THE FORMER YOU AN AMBASSADOR UNDER A DONALD TRUMP FORMER GOVERNOR OF SOUTH CAROLINA SHE IS DROPING OUT OF THE RACE TO DAY BECAUSE DONALD TRUM DOMINATED SUPERTUESDAY THIS OF COURSE WAS NOT A SHOK….."

4.1 Preprocessing

The base of preprocessing consisted of the following steps:

- 1. Case Normalization: All text was converted to lowercase initially to facilitate uniform processing, especially for expanding contractions and correcting spellings (spell check in another version).

- 2. Noise Reduction by Noisy Words Removals: Commonly occurring but irrelevant words (e.g., 'um', 'uh', 'you know') that do not contribute to summary quality were identified and removed using regular expressions.

- 3. Contraction Mapping and Expansion: Contractions were expanded to their full forms using a predefined dictionary mapping to improve the text's clarity and formal tone. This dictionary is applied to the text with case preservation for the initial characters, ensuring proper nouns and sentence starts were correctly formatted.

- 4. Whitespace Normalization: Excess white spaces were reduced to a single space to maintain text uniformity.

4.1.1 Preprocessing with Named Entity Recognition (NER):
Named Entity Recognition (NER) was employed as a critical component of the text preprocessing to enhance the clarity and accuracy of the summarization process. NER refers to the technique used in natural language processing to identify and classify named entities mentioned in text into predefined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc.

NER was utilized to perform the following functions:

- Entity Identification and Placeholder Substitution: As the text was processed using Spacy's "en_core_web_sm" model, each identified entity was temporarily replaced with a unique placeholder. This placeholder was constructed in a specific format: `<ENTITY_LABEL_STARTPOS>`, where `ENTITY_LABEL` is the type of the entity (e.g., `PERSON`, `ORG`), and `STARTPOS` is the starting position of the entity in the text. This substitution aimed to preserve the entity information through subsequent preprocessing steps, ensuring that these critical data points are neither altered nor lost.

- Case Insensitivity and Placeholder Management: The placeholders were inserted into the text in a lowercase format to maintain consistency during case normalization and other preprocessing steps like contraction expansion. Lowercasing the entire text, including these placeholders, standardized the text processing and made it insensitive to case variations which might affect matching and processing logic.

- Post-Processing Entity Restoration: After all preprocessing steps were completed (e.g., contraction expansion, noise reduction, and whitespace normalization), these placeholders were replaced back with the original text of the entities. The replacement was done in uppercase to make entities stand out in the processed text.This step was crucial as it reintroduced the precise and original names of the entities into the text, ensuring that the summary generated reflects accurate and specific information.

Sample of Application of preprocessing function with NER on a Test Text:
Text: "Dr. John Smith's groundbreaking research on CRISPR gene editing at Harvard University wasn't that interesting"

Sample of Application of preprocessing function with NER on a Test Text:
Text: "Dr. John Smith's groundbreaking research on CRISPR gene editing at Harvard University wasn't that interesting"

Output: “Dr. JOHN SMITH's groundbreaking research on CRISPR gene editing at HARVARD UNIVERSITY was not that interesting”

Sample of Application of Preprocessing Function with NER on a Audio Transcribed Text:
Text: “SHE'S NOT GOING TO ANNOUNCE AN INDORSEMENT ON WEDNESDAY HOWEVER SHE'S GOING TO ENCOURAGE DONALD TRUMP TO EARN THE SUPPORT OF REPUBLICANS”

Output: “she is not going to announce an INDORSEMENT on WEDNESDAY however she is going to encourage donald trump to earn the support of republicans”

4.1.2 Preprocessing with Automated Spell Check
Automated spell checking was implemented using the `SpellChecker` library, which is a powerful tool for identifying and correcting spelling errors in text data.

Spell Checking Process
1. Initialization: A caching mechanism (`spell_check_cache`) was used to store previously checked and corrected words. This sped up the run time, by avoiding repeated checks for the same word.

2. Word Splitting: The input text was split into individual words. This tokenization was necessary because the `SpellChecker` works at the word level, checking each word independently for misspellings.

3. Spell Correction:
   - For each word in the text:
      - If the word has not been checked before (i.e., it is not in `spell_check_cache`), the script uses `spell.correction(word)` to find the most likely correct form of the word.
      - If the corrected word is found, it is added to the cache. If not, the original word is used.
      - If the word is already in the cache, the corrected word is retrieved from the cache, bypassing the correction process.
   - This resulted in a list of corrected words, which are then joined back into a single string.

4. Integration with Other Preprocessing Steps: Spell checking was integrated as one of the final steps in the text preprocessing pipeline. It followed case normalization, filler removal, contraction expansion, and whitespace and punctuation normalization. This ordering ensured that the text is in a suitable form for spell checking, potentially reducing the number of errors and improving the accuracy of corrections.

Sample of Application of Preprocessing Function with Spell Check on the Transcribed Text:
Text: "WELL IT'S THE END OF THE ROAD FOR NICKEY HALEY THE FORMER YOU AN AMBASSADOR UNDER A DONALD TRUMP FORMER GOVERNOR OF SOUTH CAROLINA"

Output: "well it is the end of the road for nickel hale the former you an ambassador under a donate trump former governor of south caroling"

The spell checker made inappropriate changes:
- "NICKEY" to "nickel"
- "HALEY" to "hale"
- "DONALD" to "donate"
- "CAROLINA" to "caroling"

These errors typically arose from the context-insensitive nature of basic spell checkers, which rely purely on statistical likelihoods from a generic corpus without understanding context or proper nouns common in specific domains (e.g., names in political texts).

4.1.3 Preprocessing with Named Entity Recognition & Spell Check Together

After recognising the errors made by the spell check library we combined traditional spell checking with Named Entity Recognition to distinguish between common words and proper nouns. The spell correction was ideally placed after the removal of noisy words and before the conversion to sentence case because of two main reasons.

- Post Noisy Word Removal: Performing spell correction after removing fillers (noisy words) such as 'um', 'uh', etc., ensures that we don't waste computational resources trying to correct words that are going to be removed anyway.
- Pre-Sentence Case Conversion: Correcting spellings before converting the text to sentence case helps in correcting any case-related errors that might affect the spell checker's accuracy.

The order of the updated preprocessing function was as follows:

1. Case Normalization
2. Contraction Expansion
3. Removal of Noisy Words
4. Whitespace and Punctuation Normalization
5. Spell Correction
6. Conversion to Sentence Case
7. Restoration of Named Entities

Sample of Application of Preprocessing Function with NER and Spell Check on Audio Transcribed Text:
Text: “THE REPUBLICAN PARTY IS THEN GOING TO SWIVEL AND TURNE TO NICKEY HALY”

Output: “THE REPUBLICAN PARTY is then going to swivel and turn to nickel half”

It can be seen that combining NER with Spell Check did slightly better compared to doing Spell Check alone, as ‘TURNE’ was corrected to ‘turn’. Due to the punctuational, grammatical and spelling related inconsistencies in the text generated from audio to text, this approach was still not stable as it changed some of the names (eg. ‘NICKEY HALEY’ to ‘nickel half’).

4.2 Summarization – Model Info and Setup

The notebook ‘SUMMARIZATION.ipynb’ contains all the code for summarization, it utilizes the BART (Bidirectional and Auto-Regressive Transformers) model, specifically the "facebook/bart-large-cnn" variant, for summarization tasks on transcribed text. BART is a transformer model optimized for sequence-to-sequence tasks, and its CNN variant was fine-tuned specifically on a summarization dataset from CNN, enabling it to generate concise summaries effectively.

1. Model and Tokenizer Initialization: The `AutoTokenizer` and `AutoModelForSeq2SeqLM` from the `transformers` library were employed to load the pre-trained 'facebook/bart-large-cnn' model and tokenizer.

2. Computing Device Configuration: The model was configured to use the `torch.device("mps")`, indicating the use of Metal Performance Shaders (MPS) backend for computation on macOS devices equipped with Apple Silicon, which supports hardware acceleration.

3. Text Tokenization: The input text was tokenized using the BART tokenizer to convert it into a sequence of tokens, which is the required input format for the model. The tokenizer was configured with a `max_length` of 1024 tokens to truncate texts longer than this limit and `truncation=True` to enable this feature.

4. Model Input and Summarization Parameters: Tokenized inputs were transferred to the specified computing device and the `model.generate` function was called with parameters designed to optimize the summarization quality:

- `max_length=300` and `min_length=100` defined the boundaries for summary length.
- `length_penalty=2.0` was set to encourage slightly longer, more informative summaries.
- `num_beams=4` employed beam search to consider multiple summarization paths, thereby enhancing quality.
- `early_stopping=True` was used to halt the search once all beams concluded, improving process efficiency.

*Note: * For The Andrew Huberman Podcast some parameters (max_length = 1000, min_length = 100, and num_beams = 6) were changed as the speech to text component produced the most noise in this data, in addition to the technical rigor of the text **

5. Output Decoding: The model's output tokens (`summary_ids`) were decoded back into text, omitting any special tokens used during model operations. The resulting text constituted the summarized content.

6. Iterative Summarization of Transcripts: A directory named ‘SUMMARIES’ was established to systematically organize each generated summary into a text file. Using batch processing (predefined start_index and end_index based on a whole podcast), the script handled specific entries from a DataFrame (df) that included cleaned and preprocessed transcripts. Each transcript was summarized with the summarize_text function, and the corresponding summary was saved in a uniquely indexed text file.

4.2.1 Qualitative Evaluation of Strategies Used

All three preprocessing strategies (NER, NER+Spell Check, Base Text Processing), were used to generate sample summaries. After the evaluation of sample summaries a final approach (Base Preprocessing) was used to summarize all the generated transcripts.

The following are the outputs of the different 4 different approaches for summaries:

- Sample Summary of NER based preprocessing:
  "nickey haley is dropping out of the race to day because donald trum dominated supertuesday. The former south carolina governor and you an ambassador is expected to makan appearance to deliver brief o mar remarks around TEN AM eastern her sision rive the day after super tuesdaysio one only vermont among the FIFTEENSDAYS on VERMONT is basically candida. According to the walstry trurnal hailey plans to ouspen her republican presiden trall primary bid in a speech on wdnestay."

- Sample Summary of Spell Check based preprocessing:
  "well it is the end of the road for nickel hale the former you an ambassador under a donate trump former governor of south caroling she is dropping out of the race to day because donate true dominated supertuesday. She really had no shot at the nomination after swan know one had a shot at nomination after i as i pointed out which you bi rondo mantis dropped out early. She is going to encourage donate trump to earn the support of republicans and independent voters who have backed her which presumably could theoretically be if i i vice presidential play may be."

- Sample Summary of NER + Spell Check based preprocessing:
  "nickel hale the former you an ambassador under a donate trump former governor of south caroling she is dropping out of the race to day because donate true dominated supertuesday. She is going to encourage donate trump to earn the support of republicans and independent voters who have backed her. If donate trump loses in NOVEMBER in the general election she jobidon it is very unlikely the THE REPUBLICAN PARTY is then going to swivel and turn to nickel half a specially."

- Sample Summary of Base Preprocessing:
  "well it is the end of the road for nickey haley the former you an ambassador under a donald trump former governor of south carolina she is droping out of the race to day because donald trum dominated supertuesday. She is not going to announce an indorsement on wednesday however she is going to encourage donal trump to earn the support of republicans and independent voters who have backed her. she is hoping that he picks her as sort of a unity ticket bot given his dominant performente in the primaryes very unlikely."

After manual evaluation on multiple samples, the Base Preprocessing approach proved to be the most effective for generating summaries. It struck a reasonable balance between readability and factual accuracy without the introduction of significant errors seen in the other preprocessing strategies. While it does allow for some minor errors, these do not substantially detract from the overall quality and integrity of the summaries, making it the superior choice for summarizing the transcribed audio data in this context. This strategy underscored the importance of a nuanced approach to text preprocessing, where the goal is to enhance clarity without compromising the text's original factual content.

4.2.3 Quantitative Evaluation using Rouge Scores
*Note: * Testing / Pre Written summaries of the transcripts were only available for the Ben-Shapiro Podcast. They were parsed from XML trees of the RSS FEED of the podcasts, preprocessed, and used for evaluation in the "SUMMARIZATION_EVALAUATION.ipynb" notebook **

ROUGE Metric: The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric was used in assessing the quality of summaries. ROUGE-1 measures the overlap of unigrams between the generated summary and a set of reference summaries. ROUGE-2 assesses the overlap of bigrams, providing insights into the sequential word agreement which indicates more coherent summaries. ROUGE-L (Most Important in Our Case) focuses on the longest common subsequence, useful for evaluating sentence-level structure similarity which helps in understanding the fluency and order of content.

These metrics collectively offer a multifaceted view of summarization performance, with ROUGE-1 and ROUGE-2 gauging content accuracy and ROUGE-L assessing form and readability.

Challenges of Audio Transcriptions: Evaluating summarization models on audio-transcribed text introduces distinct challenges. These transcriptions often contain irregularities such as misrecognitions, colloquial expressions, and filler words, deviating significantly from standard written language. These issues can detrimentally affect precision, a metric indicating the proportion of correctly identified relevant outputs in summaries. Miss-Recognitions are prevalent, especially with varied speech accents or in noisy conditions, leading to erroneous textual representations. Additionally, filler words and informal phrases, commonly omitted in refined written summaries, remain in transcriptions, inflating word counts and complicating accurate summarization.

Limitations of Precision in our Situation: Precision measures the proportion of relevant outputs (in this case, words or phrases from summaries) that are correct. Precision is particularly sensitive to noise in the input data, such as that commonly found in speech-to-text outputs because of the following:

- Misrecognized Words: Common in transcriptions, especially with accented speech or in noisy environments, leading to summaries that include incorrect or nonsensical terms eg. spelling and grammatical errors.

- Filler Words and Repetitions: Often preserved in transcriptions but typically omitted in written summaries, artificially inflating the word count in machine-generated summaries and reducing precision.
- Colloquial and Spoken Phrases: Transcriptions may include conversational phrases that are not often reflected in the more formal reference summaries used for evaluation.

Given these factors, optimizing for precision might result in overly conservative summaries that omit substantial content to avoid the inclusion of potentially incorrect elements. This approach might sacrifice the richness and informative value provided by a more comprehensive capture of the transcription's content.

The Case for Emphasizing Recall: Recall measures the proportion of relevant data points from the reference captured in the summaries, serving as a crucial metric for audio transcriptions. High recall ensures that the summary captures a comprehensive snapshot of the transcribed text, which is crucial in applications like medical or legal document processing where missing information can be critical..Recall-focused models are less penalized for including extra content, which can be preferable in noisy transcription scenarios where ensuring the inclusion of all pertinent information is more important than conciseness. This approach aligns with user needs where complete information is paramount, often outweighing the need for conciseness.

METRICS	PRECISION	RECALL
rogue1	.4516	.8889
rogue2	.3696	.8
rogueL	.4516	.8889

High Recall: The notably high recall values across ROUGE-1, ROUGE-2, and ROUGE-L metrics suggest that the summaries are extensively capturing the content present in the reference summaries. This is particularly crucial in our case for audio transcriptions:

- Comprehensive Coverage: Ensuring that all potentially relevant information from the often noisy and error-prone transcriptions is captured is vital, especially in domains requiring detailed information (e.g., medical, legal etc).
- Tolerance to Noise: Audio data often include non-standard language use, background noises, and speech errors, which can lead to the inclusion of extraneous text in transcriptions and, subsequently, in summaries.

Low Precision: The lower precision indicates that while the summaries extensively cover the reference content, they also include a significant amount of non-relevant material. This stems from the fact the length of the generated summaries and original summaries, which were sourced from the internet, were significantly different with generated summaries being on average of approximately 500 words and original summaries being on average of 220 words.
ROUGE-1 (Unigram Overlap): The high recall and moderate precision suggest that the summaries effectively include most of the keywords and basic content from the transcriptions. This ensures that essential information is not omitted.

ROUGE-2 (Bigram Overlap): The lower precision here reflects challenges in capturing two-word phrases accurately, indicative of possible issues in maintaining narrative coherence or in dealing with the syntactic structure introduced by transcription errors.

ROUGE-L (Longest Common Subsequence): The high recall in ROUGE-L indicates good preservation of longer text segments from the references in the summaries, suggesting that significant parts of the sentence structures or data sequences are maintained. This is most beneficial in our case, specially for understanding contexts or sequences within the transcribed content, critical for detailed analytical tasks.

4.3 Summarization Conclusion

In contexts like summarizing audio-transcribed content, the observed high recall is beneficial despite the low precision. It ensures that summaries are comprehensive and include all potentially crucial information, aligning well with use cases where the complete capture of content is more important than the exclusion of non-relevant details. The evaluation based on ROUGE metrics supports this approach, with ROUGE-L (.89%) being particularly indicative of the model's ability to maintain important content sequences effectively.

5 Implementing Topic Modeling for Enhanced Content Analysis:

In our ongoing efforts to enhance content analysis capabilities, we have implemented a topic modeling system using advanced machine learning techniques. This report outlines the process of developing this system, focusing on the utilization of Latent Dirichlet Allocation (LDA) and Term Frequency-Inverse Document Frequency (TF-IDF). These methods are critical in uncovering and understanding the underlying themes within large volumes of textual data. They are used at a very high scale in a bunch of different usages. Some of them are listed below:

5.1 Overview

5.1.1 Information Retrieval

Topic modeling significantly improves information retrieval systems by organizing and indexing documents according to identified topics. By associating documents with specific topics, search engines and database systems can more efficiently retrieve documents relevant to a user's query.

5.1.2. Content Organization and Management

In content management systems, topic modeling helps categorize and tag large sets of documents automatically, which simplifies content navigation and improves user experience. Applications include:

5.2 Data Preparation

The first step in our topic modeling process involves preparing the text data. This includes cleaning the text to remove noise and formatting issues, followed by tokenization—breaking the text into words or phrases. We also apply linguistic preprocessing like stopword removal (eliminating common words like 'and', 'the', etc.), stemming, and lemmatization to reduce words to their base or root form.

5. Term Frequency Inverse Document Frequency (TF-IDF)

Before applying LDA, we transform our text data using the TF-IDF technique. TF-IDF measures the importance of a term within a particular document in the corpus relative to its frequency across the entire document set. This statistic reflects how important a word is to a document in a collection and is crucial for reducing the impact of frequently appearing words that are less informative about the document's content.

5.3 Latent Dirichlet Allocation (LDA)

LDA is a probabilistic model that assumes documents are mixtures of topics, where each topic is a collection of terms with certain probability scores. Here’s how we implement LDA:

- Model Training: We use the prepared and vectorized text data (via TF-IDF) to train the LDA model. The training involves adjusting the model parameters to best capture the topic distributions within the documents.
- Hyperparameter Tuning: Key parameters like the number of topics, alpha (document-topic density), and beta (topic-word density) are tuned based on model performance metrics such as coherence score, which measures the semantic similarity between high scoring words in the topic.
- Topic Inference: Once trained, the LDA model can be used to infer the topic distribution of unseen documents, providing an insight into the prevalent themes or discussions contained within the text.

5.3.1 Key Libraries Used in the Implementation of Topic Modelling

Gensim: Gensim is an open-source library designed specifically for unsupervised topic modeling and natural language processing. It's known for its efficiency and ability to handle large text collections with ease.

Key Features Used:

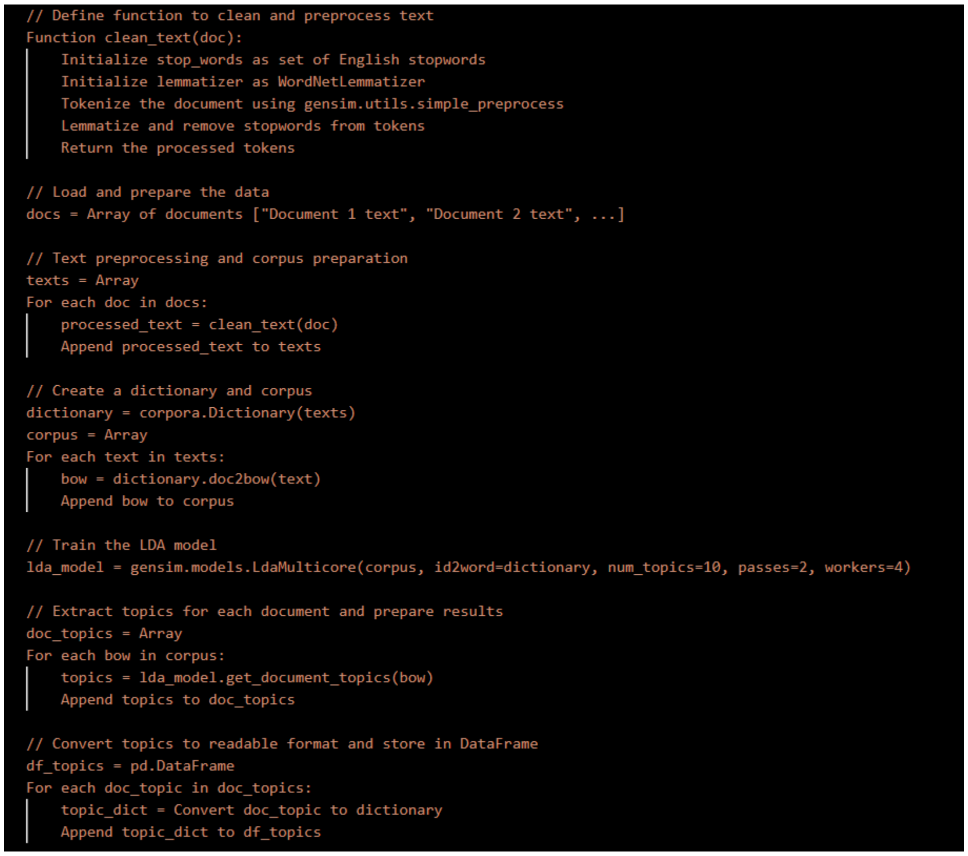
Corpora and Models: Gensim provides functionalities for building and working with text corpora, and it's particularly well-known for its efficient implementations of various topic models, including LDA.

LDA Implementation: Gensim's LdaMulticore is a variant of LDA that is optimized for multicore CPUs, enhancing performance by processing multiple documents in parallel during training.

Natural Language Toolkit (NLTK): NLTK is a comprehensive library providing easy-to-use interfaces to over 50 corpora and lexical resources, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, and parsing.

Key Features Used: Text Preprocessing: Functions for tokenizing text into sentences and words, removing stopwords (commonly used words such as 'and', 'the', etc. that do not contribute much to the meaning of text), and other linguistic processing tools which are crucial for cleaning and preparing text data for modeling.

5.3.2 Following below is the pseudocode for the Topic Modelling case:



```
// Define function to clean and preprocess text
Function clean_text(doc):
    Initialize stop_words as set of English stopwords
    Initialize lemmatizer as WordNetLemmatizer
    Tokenize the document using gensim.utils.simple_preprocess
    Lemmatize and remove stopwords from tokens
    Return the processed tokens

// Load and prepare the data
docs = Array of documents ["Document 1 text", "Document 2 text", ...]

// Text preprocessing and corpus preparation
texts = Array
For each doc in docs:
    processed_text = clean_text(doc)
    Append processed_text to texts

// Create a dictionary and corpus
dictionary = corpora.Dictionary(texts)
corpus = Array
For each text in texts:
    bow = dictionary.doc2bow(text)
    Append bow to corpus

// Train the LDA model
lda_model = gensim.models.LdaMulticore(corpus, id2word=dictionary, num_topics=10, passes=2, workers=4)

// Extract topics for each document and prepare results
doc_topics = Array
For each bow in corpus:
    topics = lda_model.get_document_topics(bow)
    Append topics to doc_topics

// Convert topics to readable format and store in DataFrame
df_topics = pd.DataFrame
For each doc_topic in doc_topics:
    topic_dict = Convert doc_topic to dictionary
    Append topic_dict to df_topics
```

5.3.3 Explanation of the pseudocode as how the topic is generated:

- Text Preprocessing: The clean_text function is defined to handle tokenization, lemmatization, and the removal of stopwords, preparing the text for further processing.

- Loading Data: Documents are assumed to be available in a simple list format.

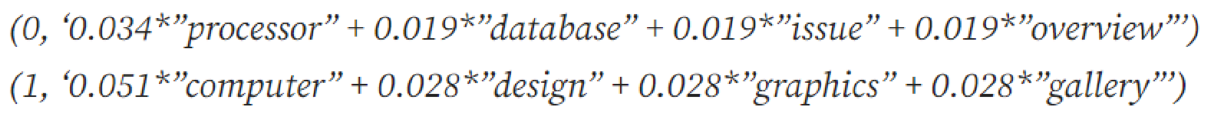
- Dictionary and Corpus Creation: The texts are processed into a 'bag of words' model using a Gensim dictionary, which maps each unique word to an integer and a corpus, which represents each document as a collection of word tokens and their corresponding frequencies.

- Model Training: The Ld Multicore method from Gensim is used to train the LDA model on the corpus, specifying parameters like the number of topics, the number of passes over the corpus, and the number of worker threads for parallel processing.

- Extracting and Storing Topics: Topics for each document are extracted and converted into a more manageable format (dictionary) for easy interpretation and storage in a Pandas Dataframe, which can be output or saved as needed.

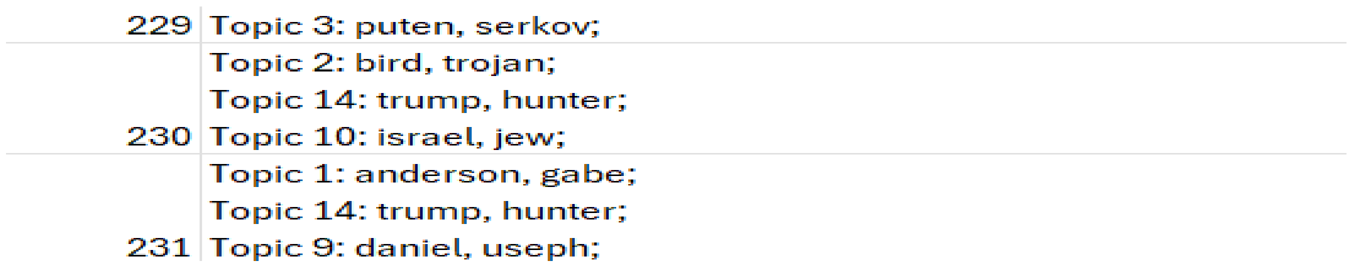
In this part, a total model was trained, and 20 topics were selected for the whole corpus. After that process each document has been checked and depending upon the frequency of the words each document has been assigned the specific topics by their respective probability. We have filtered out the main three top topics which were assigned the maximum probability. After that a loop has been run in every text present in the directory to generate the topic to get the user a brief idea as to what topics would be discussed in the podcast.

Following below is the screenshot as how the models are selected based on the assigned probabilities topic wise.Different types of text files will be generated from different documents. As mentioned below as a sample the document 0 contains topics such as Processor, database which gives us the hint about the topic discussing about the Databases.



```
(0, '0.034*"processor" + 0.019*"database" + 0.019*"issue" + 0.019*"overview"')
(1, '0.051*"computer" + 0.028*"design" + 0.028*"graphics" + 0.028*"gallery"')
```

These are how the topic is generated for the different types of txt files. For the hyperparameters tuning a total of 20 topics are selected and the number of passes will be 15 to get most of the data. and hence the top three topics have been selected. The csv files the topics are saved with the index number along with these are the results of the topic modeling generated for each individual podcast text.



229	Topic 3: puten, serkov;

230	Topic 2: bird, trojan;
	Topic 14: trump, hunter;
	Topic 10: israel, jew;

231	Topic 1: anderson, gabe;
	Topic 14: trump, hunter;
	Topic 9: daniel, useph;

5.4 Conclusion:

In the context of Topic modeling we have taken and filtered out the top three topics of individual podcast text. The entire corpus was taken and important are categorized based on the topics. Total 20 topics were selected.However based on frequency of individual words the topic modeling could not do well in all the documents , also because of challenges related to spelling errors. I have observed that in some of the podcasts the same topics are generated which doesn't give us a good result.It may be due to the spelling errors which are getting repeated as some words are not properly transcribed leading to some changes in desired output result.

In the future tasks we need more training data and will run more iterations to the more desired result. We would like to get the better transcribed generated data as per the expectation which will also help in generating the desired output result.

Github Repo Link:

https://github.com/Pmalik24/Automated_podcast_sumarization__and_highlight_generation