

Extracting Advertisements in Spotify Podcast Transcripts using Transformer Models in a Task Oriented Chatbot

Olivia Martin^{1,†}

¹ Northwestern University School of Professional Studies
Master of Science in Data Science Program
633 Clark St, Evanston IL 60208

[†]Address to which correspondence should be addressed:
olivia.martin.473@gmail.com

Abstract

Using transformer models to read podcast transcripts makes it possible to answer a variety of user queries including identifying advertisements. The research proposes an efficient solution to improve interactions between hosts, advertisers, and listeners. The task chatbot can identify advertisements in specific shows by using a transformer model to perform a semantic search of the episode transcript. Spotify podcast transcripts offer expansive text data, but efficiently searching long transcripts is difficult beyond keyword searches and baseline natural language processing (NLP) techniques. Utilizing transformer models based on large language models (LLMs) can help listeners find advertisements among the variety of podcast and advertisement styles.

Introduction

As many podcasts are free to listen to, hosts can earn money by attracting paid episode advertisers. A listener may skip past these ads, but later, may want to find a coupon code and show support for a podcast. Instead of manually searching through the entire episode, a chatbot or conversational agent could assist in finding advertisers and coupon codes. Creating a chatbot to find advertisers will make searching podcast episode transcripts easier. LLMs like BERT, are powerful tools to evaluate text documents and answer user queries. Skipping advertisements is easy when listening to a podcast, so having the option for the user to search for sponsored segments could help the podcast host earn money, and the listener save money.

Podcasts are a growing form of media, attracting millions of listeners who engage with shows in various formats and themes (Marshall 2023). The landscape of podcasts can include the following podcast types: conversational, interviews, sponsored content, narrative, non-fiction, and many more. The podcast topics can also vary widely, including dating, education, crime, history, sports and many more. Spotify, a digital audio service, provides music and more than 5

million podcast titles to 515 million users across 184 markets (Spotify A.B. n.d.). In 2021, podcast advertising totaled 1.34 billion U.S. dollars and is projected to grow to 2.56 billion dollars by 2024 (Insider Intelligence 2022). Of those advertisements, host-read ads generated almost half of all podcast revenue between 2019 to 2021, (IAB (U.S.) 2022). Adult podcast listeners are likely driving this trend in advertisements as 48% of survey respondents said they preferred an ad voiced by the show's host (Cumulus Media 2022). These host-read ads are scripted but could be less distinguishable from regular podcast content and more difficult to locate. The increased listener interest and money spent on podcast advertising indicates that increasing access to podcasts advertisements is valuable work.

Literature Review

TREC Conference - Podcast Track

The TREC, Text Retrieval Conference, hosted the first Podcast track in 2020 to encourage research into podcasts in the information retrieval and NLP research communities (Jones et al. 2021, 1). There was great interest in bringing this track to TREC as podcast data is more varied in style than previous tracks like TREC Spoken Document Retrieval, which focused on broadcast news transcripts generated from automatic speech recognition systems (Jones et al. 2021, 2). Another conference, CLEF, explored retrieval from a large archive of spoken testimonies (Jones et al. 2021, 2). These conferences found challenges in identifying topic starting points for less structured text, accurately querying when entities are not mentioned frequently, and handling content diversity (Jones et al. 2021). There are some other spoken content datasets with variety like podcasts including the Blip10000, a social video dataset and BBC television dataset (Jones et al. 2021).

The Podcast tracks for both TREC 2020 and TREC 2021 conferences included two tasks: summarization and segment retrieval. The summarization task focused on generating an episode

description summary, and the segment retrieval task concentrated on identifying a relevant segment to a query. In both years, episode audio was available, but most participants did not utilize the audio, even though approaching queries with a multimodal approach has been effective (Jones et al. 2021). In 2021, the podcast segment retrieval changed to include reranking criteria as entertaining, subjective, or discussion-based (Jones et al. 2021). The baseline runs for each year include traditional information retrieval methods, BM25, query likelihood (Q.L.) with the Pyserini package, and neural reranking BERT model pre-trained on MS MARCO dataset (Yu et al., 2020). Judging the effectiveness of the research methods must be done against a quality baseline; otherwise, claims of novel results may not be as efficacious as claimed (Lin, Rodrigo, and Yates 2020, 42). BM25 has proven to be a very successful term weighting method for vector space models and is based on exact term matching and uses term frequency and document frequency (Lin, Rodrigo, and Yates 2020, 12)

Best Performing TREC Podcast Runs

Of the evaluation queries, Galuscáková et al. (2020) used a combination of the query description and title fields when developing their retrieval method for relevant transcript segments. Galuscáková et al. (2020) created seven separate retrieval systems that used different combinations of data processing techniques, including stemming, word2vec, and TF-IDF using the Indri query language. Each retrieval system produces possible query matches using different evidence, so combining the rankings should exceed one system. The systems are combined using a reciprocal rank combination, then the combination is reranked using the BERT-Large model and T-5 model, and then all three reranked outputs are combined again (Galuscáková et al. 2020). NDCG scores, a metric used to judge ranking models, are compared across the systems.

Almost all teams that submitted runs in the 2021 TREC Podcast Track used transfer learning. Hofstätter, Sertkan and Hanbury (2021) had several well ranked podcast runs because

they used models that they developed for the Deep Learning track. The best run combined the ranking from the BM25 from Pyserini and the DistilBERT Topic Aware Sampling-Balanced ranking model and then re-ranked the results with their BERTcat reranking model (Hofstätter, Sertkan and Hanbury 2021).

Another approach a TREC Podcast team used to improve their results combined Elasticsearch (BM25) and 1,000 two-minute podcast segments manually annotated according to the reranking criteria (Bondarenko et al. 2021). The labeled data set was used to train a SVM classifier model to predict podcast reranking criteria segments (Bondarenko et al., 2021, 4). The overall score of a podcast segment became a combination of the BM25 score multiplied by the SVM's confidence probability. However, this run did not beat the baseline run using just the top 1,000 BM25 results (Bondarenko et al. 2021, 4).

Additional Podcast Research

Although the research into transcribed podcast text is relatively new, recent developments include a sentence transformer model, pinecone distiluse-podcast-nq, that is fine-tuned with generated queries on podcast data (Briggs, 2022). Compared to keyword searches, semantic search allows users to search for concepts by the meaning of words versus exact word matches (Briggs 2022). The sentence transformer model receives the sentence tokenized transcript, encodes the sentences, and then the cosine similarity of each sentence is compared to the set question. The basis for this Pinecone model is BERT-base-uncased, which is trained on general purpose English language (Briggs 2022). Pinecone's motivation to create this model was to recreate Spotify's search querying process. In order to replicate the query and episode pairs, Pinecone used a fine-tuned query generation BART model, doc2query/all-t5-base-v1, that used the MS MARCO dataset and then used the generated 100k queries to fine-tune the DistilUSE-base-multilingual-case-v2 model (Briggs 2022).

Another study, Reddy et. al. (2021), worked on producing better summarizations of podcasts based on episode transcripts by identifying and removing advertisement-related sentences. Reddy et al. (2021) had two approaches to advertisement identification: creating a ground truth by annotating transcript sentences and analyzing where users skip or fast-forward content as a possible advertisement indicator. As many podcast advertisements are read by the podcast host, identifying where a transcript shifts from content to advertisement is difficult. The results from Reddy et al. (2021) indicated promising document and sentence identification with the sentence-level BERT model on podcast show descriptions, 88.7% identification for gold transcript sentences, and 96.6% identification of silver transcript sentences.

Some studies show evidence that using a multimodal approach that includes spoken content text and audio can help improve search on podcast transcripts. Vaiani et al. (2022) utilized audio encodings to create summaries via extraction using an ad hoc, deep feature fusion network, MATeR (Multimodal Audio-Text Regressor). Overlapping speech and background effects makes podcast data complex, but using acoustic and textual features can improve extracting segments to correspond to queries (Vaiani et al., 2022). Similar to Reddy et al. (2021), Vaiani et al. (2022) wanted to improve summarization by identifying and removing advertisements from descriptions prior to training the retrieval models. The golden summaries, human-generated descriptions, are compared to the automatically generated summaries using the Rouge scores, measuring syntactic unit overlap and measuring similarity using cosine distance between Sentence-BERT encodings (Vaiani et al. 2022). Leveraging the combination of text and audio using MATeR saw performance gains across Rouge and SBERT scores compared to the other current models and the text-only MATeR model (Vaiani et al. 2022).

Long document Transformers

Many transformers cannot assess the entirety of a long document. Beltagy, Peters, and Cohan (2020) have developed a new transformer to handle long documents in a variety of tasks like text classification and question-answering and outperforms RoBERTa on many standard long document datasets. Many BERT-style pretrained models have a maximum ability to analyze 512 tokens at a time, and the Longformer has multiple-layer self-attention components that capture contextual information from the entire document (Beltagy, Peters, and Cohan 2020). Previous attempts to work with long documents with transformer models like BERT were done manipulating the document in various ways like chunking, which lead to information loss (Beltagy, Peters, and Cohan 2020). Beltagy, Peters, and Cohan (2020, 5) utilize multiple phases of training that start with a short sequence length and window size with a high learning rate and then increase the sequence length and window size while decreasing the learning rate.

Chatbots

In recent years, the human-like performance of large language models like ChatGPT and Bard has made user interactions closer to human interactions. However, there are limitations in the passive approach in responding to queries (Liao, Yang and Shah, 2023). Artificial intelligence, information retrieval, and natural language practitioners should consider incorporating proactive approaches to conversational agents (Liao, Yang and Shah, 2023).

Chiu et al. (2022) offers a framework based on BlenderBot, similar to a salesperson, to allow a conversational agent to engage in chit chat dialogue and move into task-oriented dialogue when appropriate. Requiring manual rules for dialogue can make scaling a chatbot's capabilities difficult, and Chiu et al. (2022) address this problem by creating the first large-scale dataset of transitioning chit-chat to task-oriented dialogues. Refining the task-oriented intent detection is an essential aspect of a more natural conversation. The BlenderBot framework could

replace a direct ask of the conversational agent with more of a thought, and the conversational agent interprets each user response for a possible task intent (Chiu et al. 2022). The next step is to respond to the task intent either with a template-based response or a generative-based response that allows for a more natural transition.

Amir (2017) identifies that one of the challenges to creating a chatbot is trying to emulate fundamental human interactions. Carefully considering conversational patterns is essential to creating a helpful bot. For task-oriented chatbots, Amir (2017) indicates that the goal is to design a chatbot to communicate with the user to complete a task efficiently. There is a more regimented flow for task chatbots, but like real-life conversations, there is the potential to go off-topic, and course correction is necessary (Amir 2017). Identifying key entities to extract can assist in better course correction versus analyzing all unstructured user inputs (Amir 2017). Another strategy that Amir (2017) recommends to establish a successful chatbot is to onboard users and clearly explain the chatbot's purpose and possible tasks. The onboarding of users may be necessary if a chatbot is not utilizing the detection of task transitions and relies on template-based responses versus generative responses.

Data

Podcast audio and transcript data have not been extensively investigated compared to other spoken content like commercial broadcasting (Clifton et al. 2020). Spotify podcast titles and descriptions have been available to study, but only recently has Spotify made transcripts and audio available. In 2020, Spotify created the first version of the Podcast Transcript and Audio dataset for the Text Retrieval Conference's (TREC) Podcasts Track (Clifton et al. 2020). The English-language version includes over 100,000 transcribed podcast episodes and metadata (Clifton et al. 2020). Google's Cloud Speech-to-Text API transcribed the podcast episodes and provides many features: word-level alignments, speaker diarization, casing, and punctuation

(Clifton et al. 2020). The entire dataset has a sample word error rate of 18% and named entity recognition of 81.8% (Clifton et al. 2020). The spotify-podcasts-2020-summarization-testset was used for this study and consists of 1027 episodes. Within a series of nested directories, the test set of transcripts is indexed, and each show has one directory and one to three possible episodes. The podcast metadata is in a CSV file. The entire test set of episodes was loaded into elastic search, but only 30 podcasts were processed with the models. The limited sample size is due to time constraints around manually evaluating model responses to use the NDCG scores.

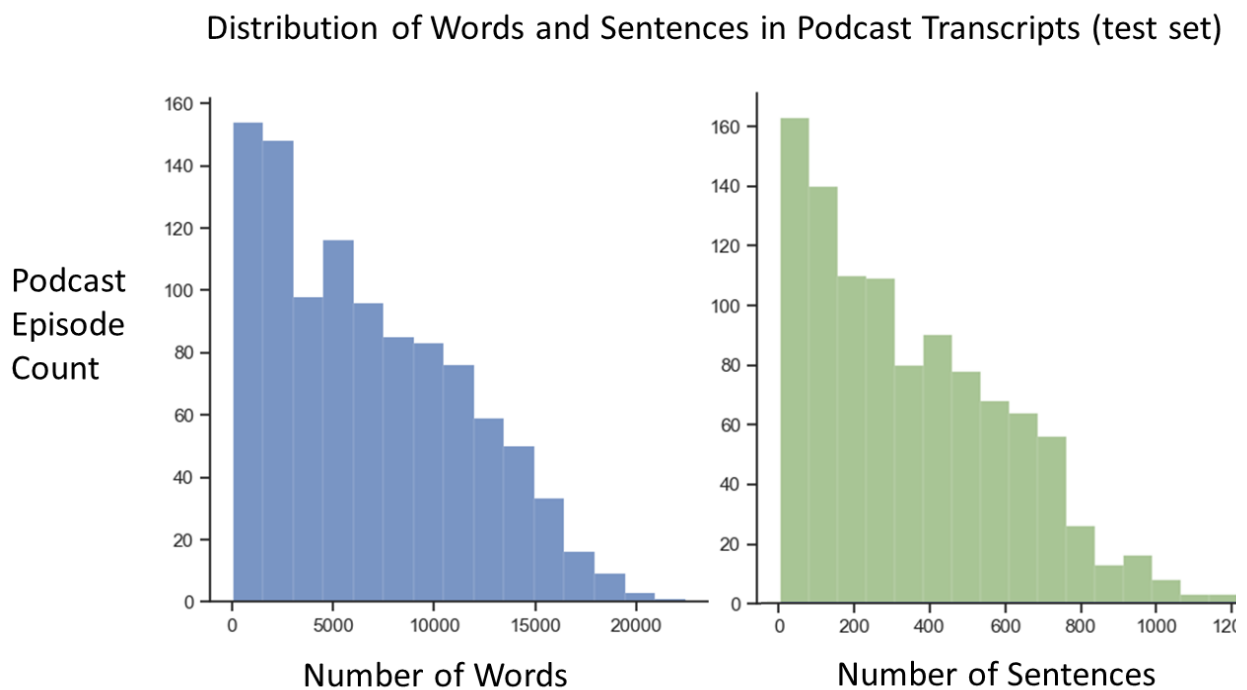


Figure 1. Distribution of words and sentences across 1027 podcast episode transcripts. Median number of words is 5988 and the median number of sentences is 300.

Methods

Chatbot Setup

The transcripts are read into Elasticsearch, version 7.11.2, using a Python script to extract only the transcript data. Appendix Figure 1 shows the document structure.

Once the podcast data populates Elasticsearch, the chatbot can begin interacting with the user to identify the show and the episode they want to know the advertisement for (Figure 2).

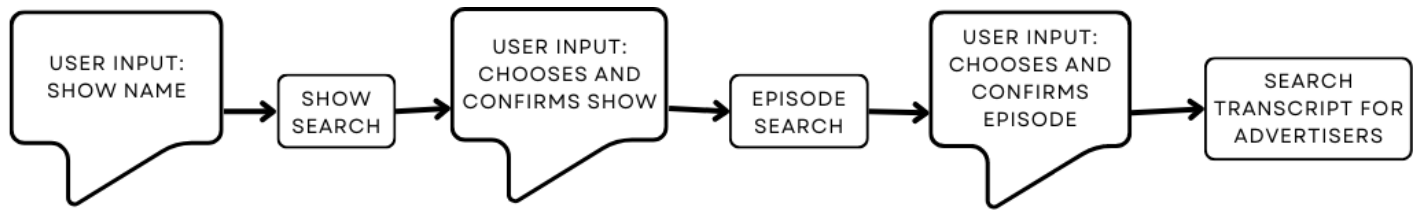


Figure 2. Flow diagram of chatbot. Elasticsearch is used to identify the list of shows and episodes. The sentence transformer, pinecone/distiluse-podcast-nq, searches the transcript for sentences that indicate advertisement or sponsorship.

The show search uses a fuzzy query, returning documents with terms in all possible variations via the *show_name* and *show_description* values in the *_source* of the episode document. The chatbot receives the user confirmation, retrieves the selected show's list of episodes via Elasticsearch and then the user confirms the chosen episode.

Testing Transformer Models

Three transformer models were selected, two query based models, pinecone/distiluse-podcast-nq and sebastian-hofstaetter/distilbert-dot-tas_b-b256-msmarco and one classification model, morenolq/spotify-podcast-advertising-classification and downloaded from the huggingface (Wolf et. al. 2019). When the transcripts were loaded into Elasticsearch, the nltk library, Natural Language Toolkit, tokenized the transcript by sentences. Each model was tested using the 30 sample transcripts at the sentence level. The query-based models both used one query to compare transcript sentences to. The advertisement query was, "Who are the advertisers, sponsors, advertisement, or ads and/or businesses, people, teams thanked?". For the pinecone/distiluse-podcast-nq, the process to compare query to input sentence is based on the calculated cosine similarity score. The sebastian-hofstaetter/distilbert-dot-tas_b-b256-msmarco model used a dotproduct scoring metric. The top 30 results were identified for each query model. For morenolq/spotify-podcast-advertising-classification, the probability of predicting a sentence to contain advertisement was used as the scoring metric. The top results for the classification model were identified to be used for developing the truth score.

In order to use the NDCG score, the top results for each model were manually reviewed and judged against a framework to qualitatively measure advertisement existence.

<i>Value</i>	<i>Grade</i>	<i>Reasoning</i>
2	<i>Excellent</i>	The segment conveys highly relevant advertisement information and is fully on topic. An example would be a segment that includes some reference to an advertisement, sponsor or coupon code
1	<i>Good</i>	The segment conveys highly-to-somewhat relevant advertisement information and is fully to mostly on topic. An example would be a segment that is an advertisement for another podcast, the podcast itself, or a hard-to-understand advertisement.
0	<i>Bad</i>	The segment is not relevant, and no advertisement is mentioned

Table 1. True value scoring framework for model responses

The NDCG score was calculated using scikit-learn (n.d.) by using the manually labeled ‘truth score’ and the score generated by the model. To calculate the mean NDCG score, mNDCG, is calculated by averaging NDCG score calculated for the top model responses for each sample episode.

Knowledge Graph

In order to better understand what kind of advertisements exist within podcasts, a small sample of 30 episodes were read and advertisement sentences were identified, if any advertisements existed at all. Knowledge graphs were created to identify any trends among the sample transcripts and built with code from Master’s in Data Science course 453 - Natural Language Processing (Martin 2023). The tokenized sentences were used and the sentences were also tokenized by words. Variations of text processing were used including removing tags, special characters, digits, punctuation, and stop word removal to create different sets of vocabulary to test. Entities were extracted from the sentences and assigned as sources, targets, and edges. The entities determined if a word was a modifier based considering its role as a subject or object. The extracted entities are then paired and represent the source and target.

Using the Matcher function from the spacy package, the connection or edge is assigned between the source and target. The entity pairs were further filtered to find important entities by selecting certain words and phrases like ‘*make*’, ‘*spotify*’, ‘*love*’, and ‘*anchor*’ as either a source, target, or edge. The knowledge graphs were visualized using the NetworkX python library.

Results

The initial runs of the chatbot used the pinecone/distiluse-podcast-nq model to test out the flow. Of the five runs completed, the top scoring sentence did not always have a related advertisement. Somewhere within the top five cosine similarity scoring sentences there was sometimes an advertisement found. Many of the top sentences included ‘Thank you’ as the query also used the word ‘Thank you’, but this did not lead to indicating any advertisement (Appendix Table 1).

After reworking how the episodes would be processed, each query model was run and the top 30 responses and scores were recorded. The classification model mNDCG score was calculated only on the sentences predicted to be advertisements. Any advertisement predicted sentence for the classification model was recorded along with the probability score (Table 2). Final scores are listed in Table 3.

y true	top scores	top_responses
0	89.7	Thank you listeners for supporting these incredible Catholic businesses.
2	87.0	Today's episode is sponsored by glory and shine offering lotions soap bars lip balms beard care and much much more all of their products are natural wholesome and prayerfully designed and crafted to inspire the Catholic faithful during everyday acts of life and to promote prayer meditation on scripture and Marian devotion for ten percent off your entire order use promotional code Rua at checkout.
2	86.0	Today's episode is all Also sponsored by pray more novenas this Lent we can't recommend enough the pray more Retreat dot-org this online Retreat will help you meditate on the Lord's passion death and Resurrection.
0	85.2	She is passionate about her faith and has contributed to a number of Publications including catholicmatch and radiant magazine.
1	84.7	Please visit pray more Retreat dot-org.

Table 2. Example podcast episode sentence ranking by sebastian-hofstaetter/distilbert-dot-tas_b-b256-msmarco

	Model	mNDCG score
1	pinecone/distiluse-podcast-nq	0.28
2	morenolq/spotify-podcast-advertising-classification	0.17
3	sebastian-hofstaetter/distilbert-dot-tas_b-b256-msmarco	0.33

Table 3. Mean NDCG score rankings for all three models. Top 30 sentences for model 1 and model 2. Average 5 - 6 sentences marked as advertisement from model 2.

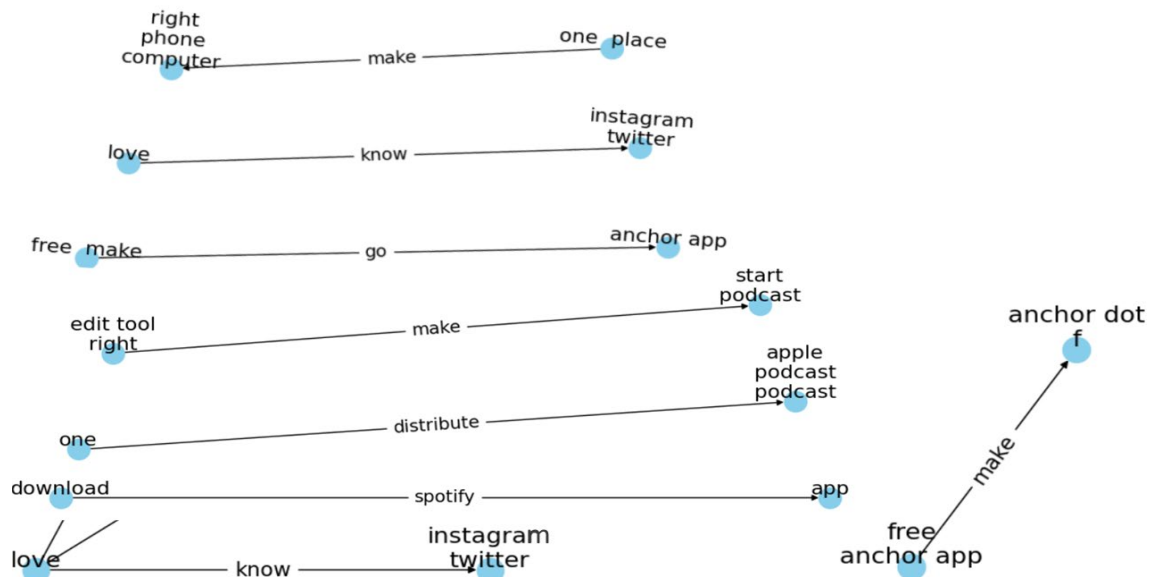


Figure 3. Example podcast episode sentence ranking by sebastian-hofstaetter/distilbert-dot-tas_b-b256-msmarco

Analysis and Interpretation

Part of looking for advertisements in podcasts transcripts assumes advertisements exist in a transcript. Looking through the small sample of podcast transcripts, not many advertisements were apparent. Although many podcasts did advertise themselves within an episode, upon first glance. Since the data is also from Spotify, there were many mentions of Spotify or products from Spotify advertised in the episodes; including, Anchor a podcast recording software, as seen in the knowledge graph (Figure 3). The knowledge graph was prepared after testing the models and did not inform the formation of the query used. Across the 30 episodes sampled, the top responses often included ‘Thank you’ which was backed by the true score, but if the sentence only had ‘Thank you’ it would be given a score of 0.

The metadata provided does not indicate the listenership to the podcast show, so identifying the maturity of the podcast is unknown. More established podcasts may have more advertisements versus podcasts that have just started to publish episodes. Additional analysis of what queries could help target certain types of advertisements would be important to look into.

For the classification model, `morenolq/spotify-podcast-advertising-classification`, the advertisements that the model was trained on were from podcast episode descriptions. The sentence advertisement training data is available from Vaiani et al. (2022) and identifies many sentences with links as advertisement. There were some advertisements found in the transcripts that were for websites, so making sure that the entire website is not sectioned by the periods when sentence tokenizing is important. The difference in how advertisement is written in episode descriptions seems to be different than how it is spoken in transcripts. Another issue with applying the classification model to the transcript data set was the model run time took hours compared to the DistilBERT based models.

The two models that were fine-tuned on Spotify title and descriptions did not perform as well as the sebastian-hofstaetter/distilbert-dot-tas_b-b256-msmarco that was not fine-tuned on podcast data (Table 3). Both query models did perform better than the classification model. The main benefit to the top performing model is the architecture that allows for dense passage retrieval. sebastian-hofstaetter/distilbert-dot-tas_b-b256-msmarco still has a limit of 512 tokens but was able to find more semantic meaning from the transcripts in relation to the query.

Conclusions

Choosing the right transfer learning model for the application may not always depend on what data the model was fine-tuned on. However, the architecture related to the task is critical. Both models that were trained on Spotify podcast titles and descriptions are not able to derive as much semantic similarities between different types of advertisements compared to the best performing model. sebastian-hofstaetter/distilbert-dot-tas_b-b256-msmarco model emphasized how using dense passage retrieval compared models that may have sparse matrices and was able to achieve the best mNDCG score of the three models. In order to judge any real novelty using the transformer models for this task, the models should be compared to an Elasticsearch baseline. Creating the baseline search to use Elasticsearch and find relevant sentences within the transcript was not completed due to issues indexing the transcript sentences as individual documents.

Another topic to consider is the level of analysis of the podcast transcript text. All of the models run were at the sentence level. Continuing to manually identify advertisements would help to better understand the composition of various advertisements using techniques like n-gram of phrases. When developing the ground truth scores from the sample podcast transcripts, many advertisements could span four to five sentences. Trying to extract one sentence at a time may remove the context of a paragraph that advertises a product.

Directions for Future Work

Testing the Lonformer model on the podcast transcripts would be a key next step to see what insight could be retrieved from a unified long document without splitting the document into many chunks (Beltagy, Peters, and Cohan, 2020). Another aspect to consider is how to incorporate combining the newest systems with traditional NLP techniques and reranking the model outputs as tested by Galuscáková et al. (2020). Other future work includes testing additional queries to account for types of advertisements and exploring generative transformer models to improve the chatbot user experience. Further down the line, learning how to work with the available podcast audio would be important in order to incorporate audio content along with the transcripts. This multimodal approach could improve the identification of advertisements and, potentially, the style of ad (Vaiani et al., 2022).

Acknowledgements

Thank you to Dr. Alianna J. Maren for thoughtful feedback for my first draft. Additionally thank you to The Writing Place at Northwestern University tutor, Laurisa Pabon, fellow MSDS 498 student Reaves Wimbish, and partner, Jose Alcala for looking over my final paper. I found Chat-GPT (3.5) helpful to assist in establishing the indexing of the transcripts from the files into Elasticsearch (interaction in the appendix)

Code Availability

Jupyter notebooks can be found here: https://github.com/oimartin/podcast_ad_sponsor_chatbot

References

- A.B., Spotify. n.d. "About Spotify." Spotify. Accessed May 15, 2023. <https://www.spotify.com/us/about-us/contact/>.
- Beltagy, Iz, Matthew E. Peters, and Arman Cohan. 2020. "Longformer: The Long-Document Transformer." *Allen Institute for Artificial Intelligence*: 1-17. <https://doi.org/10.48550/ARXIV.2004.05150>.
- Briggs, James. 2022. "Spotify's podcast Search explained." Pinecone. Accessed May 15, 2023. <https://www.pinecone.io/learn/series/wild/spotify-podcast-search/>
- Bondarenko, Alexander, Maik Fröbe, Marcel Gohsen, Sebastian Günther, Johannes Kiesel, Jakob Schwerter, Shahbaz Syed, et al. 2021. "Webis at TREC 2021: Deep learning, health misinformation, and podcasts tracks." *Thirtieth REtrieval Conference Proceedings (TREC 2021)*. National Institute of Standards and Technology (NIST), Special Publication: 1–6. <https://trec.nist.gov/pubs/trec30/papers/Webis-DL-HM-Pod.pdf>
- Clifton, Ann, Sravana Reddy, Yongze Yu, Aasish Pappu, Rezvaneh Rezapour, Hamed Bonab, Maria Eskevich, et al. 2020. "100,000 podcasts: A spoken English document corpus." *Proceedings of the 28th International Conference on Computational Linguistics*, 5903–5917. <https://aclanthology.org/2020.coling-main.519.pdf>
- Cumulus Media. 2022. "Preferred podcast ad types in the United States as of October 2022." Chart. *Statista*. Accessed August 09, 2023. <https://www-statista-com.turing.library.northwestern.edu/statistics/1238793/podcast-ad-types-usa/>
- Galuscáková, Petra, Suraj Nair and Douglas W. Oard. 2020. "Combine and Rerank: The University of Maryland at the TREC 2020 Podcasts Track." *Text Retrieval Conference (2020)*: 1 - 9. https://trec.nist.gov/pubs/trec29/papers/UMD_IR.P.pdf
- Hofstätter, Sebastian, Mete Sertkan, and Allan Hanbury. 2021. "TU Wien at TREC DL and Podcast 2021: Simple Compression for Dense Retrieval." In *Proceedings of Text REtrieval Conference (TREC)*: 1 - 6. https://trec.nist.gov/pubs/trec30/papers/TU_Vienna-DL-Pod.pdf
- IAB (U.S.). 2022. "Distribution of podcast advertising revenue in the United States from 2019 to 2021, by ad type." Chart. *Statista*. Accessed August 09, 2023. <https://www-statista-com.turing.library.northwestern.edu/statistics/1238595/us-podcast-advertising-revenue-type/>
- Insider Intelligence. 2022. "Podcast advertising spending in the United States from 2019 to 2024 (in billion U.S. dollars)." Chart. *Statista*. Accessed August 09, 2023. <https://www-statista-com.turing.library.northwestern.edu/statistics/610071/podcast-ad-spending-us/>
- Jones, Rosie, Ben Carterette, Ann Clifton, Maria Eskevich, Gareth J. F. Jones, Jussi Karlgren, Aasish Pappu, Sravana Reddy, and Yongze Yu. 2021. "TREC 2020 Podcasts Track Overview". *The Proceedings of the Twenty-Ninth Text REtrieval Conference Proceedings (TREC 2020)*: 1-17. <https://doi.org/10.48550/ARXIV.2103.15953>.

- Liao, Lizi, Grace Hui Yang, and Chirag Shah. 2023. "Proactive Conversational Agents in the Post-ChatGPT World." *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*: 3452–55.
<https://doi.org/10.1145/3539618.3594250>.
- Lin, Jimmy, Rodrigo Nogueira, and Andrew Yates. 2020. *Pretrained Transformers for Text Ranking: BERT and Beyond*. arXIV. <https://arxiv.org/abs/2010.06467>.
- Marshall, Nancy. 2023. "Council Post: Podcasting Is Growing for a Reason-Don't Miss out on That Growth." *Forbes*, January 18, 2023.
<https://www.forbes.com/sites/forbesagencycouncil/2023/01/17/podcasting-is-growing-for-a-reason-dont-miss-out-on-that-growth/?sh=6e91847b42a7>.
- Martin, Olivia. 2023. "Evaluating NLP Techniques to Create Podcast Chatbot." Masters in Data Science 453 Assignment 3, Northwestern University.
- scikit-learn. n.d. "Sklearn.Metrics.Ndcg_score." Accessed August 20, 2023. https://scikit-learn/stable/modules/generated/sklearn.metrics.ndcg_score.html.
- Shevat, Amir. 2017. "Chapter 8. The Conversation." In *Designing Bots: Creating Conversational Experiences*. Sebastopol: O'Reilly Media.
<https://learning.oreilly.com/library/view/designing-bots/9781491974810/>.
- Reddy, Sravana, Yongze Yu, Aasish Pappu, Aswin Sivaraman, Rezvaneh Rezapour, and Rosie Jones. 2021. "Detecting Extraneous Content in Podcasts." *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*: 1166 - 1173 . <http://arxiv.org/abs/2103.02585>.
- Vaiani, Lorenzo, Moreno La Quatra, Luca Cagliero, and Paolo Garza. 2022. "Leveraging Multimodal Content for Podcast Summarization." *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*: 863 - 870.
<https://doi.org/10.1145/3477314.3507106>.
- Chiu, Ssu, Maolin Li, Yen-Ting Lin, and Yun-Nung Chen. 2022. "SalesBot: Transitioning from Chit-Chat to Task-Oriented Dialogues,". *ACL 2022*: 1 - 16.
<https://doi.org/10.48550/ARXIV.2204.10591>.
- Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac et al. 2022. "Huggingface's transformers: State-of-the-art natural language processing." *arXiv*: 1- 8. <https://doi.org/10.48550/arXiv.1910.03771>
- Yu, Yongze, Jussi Karlgren, Ann Clifton, Md Iftekhar Tanveer, Rosie Jones, and Hamed R. Bonab. 2020. "Spotify at the TREC 2020 Podcasts Track: Segment Retrieval." *The Proceedings of the Twenty-Ninth Text REtrieval Conference Proceedings (TREC 2020)*: 1-9. <https://trec.nist.gov/pubs/trec29/papers/Spotify.P.pdf>

Appendix

Runs	Word Phrase Show Search	Num. Shows	Show Name	Num. Episodes	Episode Name	Response	Cosine Similarity
1	bachelor	5	The Bachelor After Show Podcast	3	The Women Yell All - S24 E10	Thank you.	0.317
						And today Spotify is one of our sponsors on Spotify.	0.330
						We just want to say thanks to our sponsor	0.539
2	therapy podcast	9	Tell Me What You're Proud of with Dr. Maggie Perry	2	#2 Finding Freedom From Your Anxious Mind w/Dr. Martin Seif founder of Anxiety and Depression Foundation of America	For exposure to work.	0.240
						It's a component of successful exposure.	0.241
						Some of them are very famous now, some of them are not Exam is but they were in Innovative and there's also a rise of something called the consumer movement where people the consumers the old patients were saying wait a second	0.263
3	sports	4	Wildchat Sports	1	1. Interview w/ The Professor	But thank God	0.284
						Thank you	0.317
						Thank goodness.	0.329
4	coffee	8	Crime Over Coffee	2	Dorothy Jane Scott - Murdered	If you would like you can support us by going to Anchor dot f m / crime over coffee donations are greatly appreciated and in making the podcast possible other ways to support us include recommending us to friends and family giving us a five star review on Apple podcasts and subscribing to us on your favorite podcast listening median.'	0.174
						We want to take some time to introduce you to one of our sponsors and our personal podcast parent anchor is an all-in-one platform for creating successful podcast view your phone or computer.'	0.321
						They will even help you find other companies to sponsor your podcast.	0.493

5	history	8	Political Scandals	1	Scandal 36: Carroll Hubbard	He gave people up to one thousand dollars to make contributions to his wife's campaign as the book of love says if you love someone pay other people to love them too now, whatever do Looted confidence had convinced Hubbard.	0.257
						Folks he deserved their money.'	0.262
						Hubbard began skimming off large chunks of money from his campaign fund and passing them to his wife.	0.274

Appendix Table 1: Full view of runs with first iteration of podcast chatbot using the model pinecone/distiluse-podcast-nq.

```
index_review = {
    '_index': "spotify_podcast_transcripts",
    '_id': episode_uri,
    '_source': {'show_uri': show_uri,
                'transcript': joined_transcript,
                'sentence_tokens': sentence_tokens,
                'word_tokens': word_tokens,
                'num_sentences': len(sentence_tokens),
                'num_words': len(word_tokens)
                **episode_metadata}}
```

Appendix Figure 1. This is the document structure for the spotify_podcast_transcripts index. Each document or episode has an id that is the unique episode_uri given by Spotify. The episode_metadata represents the show_uri, show_name, show_description, publisher, episode_uri, episode_name, episode_description and duration.

I have this file structure that has many nested folders that all eventually lead to a podcast transcript that hasn't been cleaned yet. how would I read in this file structure to elastic search, so that I don't have to flatten the file structure manually. { '_index': 'podcast_transcripts', '_id': episode_id, '_type': 'episode_transcripts', '_source': { } } My _source will have podcast show name, podcast episode name, and podcast episode transcript here is the file structure:

```
\---spotify-podcasts-2020
+---podcasts-transcripts-summarization-testset |
+---0 ||
+---1 |||
+---show_015DbLwcXu2fK7e9jIfbFo |||
```

```
+---show_01DbRiALDPdvZdoiY8yQL6 |||
+---show_01eumErJvBdxCW4YJivbwc |||
```

To read the file structure and index the podcast transcripts into Elasticsearch without manually flattening the file structure, you can use a script or program to recursively traverse the directory structure, extract the necessary information, and index it into Elasticsearch.

Here's an example Python script that demonstrates the process:

```
```python
import os
from elasticsearch import Elasticsearch

Connect to Elasticsearch
es = Elasticsearch()

Define the root directory of the file structure
root_directory = 'path/to/spotify-podcasts-2020/podcasts-transcripts-summarization-testset/0'

Function to recursively traverse the directory structure
def index_podcast_transcripts(directory):
 for item in os.listdir(directory):
 item_path = os.path.join(directory, item)
 if os.path.isdir(item_path):
 # If it's a directory, recursively traverse it
 index_podcast_transcripts(item_path)
 else:
 # If it's a file, process the transcript
 podcast_show_name = directory.split('/')[-3]
 podcast_episode_name = directory.split('/')[-2]
 podcast_episode_transcript = open(item_path, 'r').read()

 # Index the transcript into Elasticsearch
 doc = {
 'podcast_show_name': podcast_show_name,
 'podcast_episode_name': podcast_episode_name,
 'podcast_episode_transcript': podcast_episode_transcript
 }
 es.index(index='podcast_transcripts', id=None, body=doc)

Start indexing from the root directory
index_podcast_transcripts(root_directory)
```
```

Before running the script, make sure to install the Elasticsearch Python client using `pip install elasticsearch`. Also, modify the `root_directory` variable with the appropriate path to the root directory of your file structure.

The script recursively traverses the directory structure, extracting the podcast show name, episode name, and transcript for each file it encounters. It then indexes this information into Elasticsearch using the Elasticsearch Python client.

Please note that this is just a basic example, and you may need to modify it according to your specific requirements and file structure.

I just thought that since the podcast episode transcript in the file structure has not been cleaned yet maybe I should make a section part of the `_source` that just has the unclean transcript and create another script to clean the transcript and then add it to the elasticsearch index...

That's a good approach! It's often recommended to perform any necessary data cleaning or preprocessing before indexing the data into Elasticsearch. By adding a separate section in the `_source` field for the unclean transcript and then later cleaning and adding the cleaned transcript, you can maintain both versions of the transcript in your Elasticsearch index.

Here's an updated version of the Python script that includes a cleaning step before indexing the transcript:

```
```python
import os
from elasticsearch import Elasticsearch

Connect to Elasticsearch
es = Elasticsearch()

Define the root directory of the file structure
root_directory = 'path/to/spotify-podcasts-2020/podcasts-transcripts-summarization-testset/0'

Function to clean the transcript
def clean_transcript(transcript):
 # Add your cleaning logic here
 cleaned_transcript = transcript.replace('
', ' ') # Example cleaning: remove
 tags

 return cleaned_transcript

Function to recursively traverse the directory structure
def index_podcast_transcripts(directory):
 for item in os.listdir(directory):
 item_path = os.path.join(directory, item)
 if os.path.isdir(item_path):
 # If it's a directory, recursively traverse it
 index_podcast_transcripts(item_path)
 else:
 # If it's a file, process the transcript
 podcast_show_name = directory.split('/')[-3]
 podcast_episode_name = directory.split('/')[-2]
```



```

podcast_episode_transcript = open(item_path, 'r').read()

Clean the transcript
cleaned_transcript = clean_transcript(podcast_episode_transcript)

Index the transcript into Elasticsearch
doc = {
 'podcast_show_name': podcast_show_name,
 'podcast_episode_name': podcast_episode_name,
 'unclean_transcript': podcast_episode_transcript,
 'cleaned_transcript': cleaned_transcript
}
es.index(index='podcast_transcripts', id=None, body=doc)

Start indexing from the root directory
index_podcast_transcripts(root_directory)
'''

```

In this updated script, a `clean\_transcript()` function is added to perform the cleaning logic on the transcript. You can modify this function to include your specific cleaning steps. Once the transcript is cleaned, both the uncleaned and cleaned versions are added to the Elasticsearch document (`unclean\_transcript` and `cleaned\_transcript`, respectively).

By keeping both versions in the index, you have the flexibility to query and analyze the data using either the original uncleaned transcript or the cleaned version, depending on your requirements.

Appendix Table 1: Chat-GPT (3.5) interaction to help build Elasticsearch index for podcast transcripts