Title
Deploying Transformer Models to Detect and Analyze Sponsored Content in Spotify Podcasts

Author
Fatima, Nishath

Publication Date
2023

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Deploying Transformer Models to Detect and Analyze Sponsored Content in Spotify Podcasts

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Applied Statistics

by

Nishath Fatima

2023

ABSTRACT OF THE THESIS

Deploying Transformer Models to Detect and Analyze Sponsored Content in Spotify Podcasts

by

Nishath Fatima
Master of Applied Statistics
University of California, Los Angeles, 2023
Professor Yingnian Wu, Chair

This research paper explores the application of Transformer models in the field of Natural Language Processing (NLP) to detect sponsored content in Spotify audio data. The paper discusses the evolution of Transformers in NLP, highlighting their efficiency and accessibility in creating meaningful narratives from large datasets. The study focuses on a dataset of 100,000 Spotify Podcasts and their descriptions, aiming to achieve three objectives: Classification, Named Entity Recognition, and Topic Modeling. The research utilizes Transformer models, specifically through the Hugging Face library, to fine-tune and implement state-of-the-art models for efficient analysis. The application of Transformer models, including BERT, promises to save time and resources compared to traditional methodologies.

The thesis of Nishath Fatima is approved.

Nicolas Christou

Akram M. Almohalwas

Frederic R. Paik Schoenberg

Yingnian Wu, Committee Chair

University of California, Los Angeles

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

The field of Natural Language Processing(NLP) and Information Retrieval(IR) has evolved beyond its machine learning counterparts in the last decade, paving the way for the boost in demand among Artificial Intelligence sectors. This progression has naturally allowed engineers to create and deploy models that are much more efficient and accessible than has ever been seen before. With the demand for more robust analyses in the data scientist role, and the need to make the most of the ever growing data ecosystem, the most valuable skill set poses to be one which can effectively use the tools that continue to evolve in an effort to create a narrative that allows company objectives to flourish.

This paper aims to explore and discuss the evolution of Transformers in the Natural Language Processing space, with an application of some of the most prominent techniques in the field used to gauge data and create a meaningful narrative of otherwise ineligible information. The application of these models will be on data acquired by 100,000 Spotify Podcasts and their descriptions. With three objectives: Classification, Named Entity Recognition, and Topic Modeling, the knowledge acquired from the data set will be especially useful for the marketing field as the primary objective aims to classify the podcasts into sponsored and not sponsored categories. Named Entity Recognition and Topic Modeling can then be applied as point of comparison and discovery between the two categories, for any distinctions that could be useful for marketers and podcast creators alike.

Moreover, the data has been acquired by Spotify in an effort to expose more modes for textual analysis beyond image and text. All data has been derived from the pure audio

recordings of the podcast, while providing a vast range of genres and topics of discussion within the ample data resource.

The pipeline of objectives will be executed using Transformer Models through the Hugging Face library- an open-source platform designed to assist machine learning engineers in fine-tuning and implementing state of the art models for limitless tasks. With the emergence of transformer models such as BERT, it is a promising endeavor that will likely save many engineers run time and resources compared to methodology that has been implemented in the past.

CHAPTER 2

Methodology: The Evolution of NLP and Transformers

2.1 Recurrent Neural Networks(RNN)

Recurrent Neural Networks (RNNs) have been the primary architecture for sequential data processing tasks for decades. They excel at capturing temporal dependencies and have been widely used in various domains, including natural language processing (NLP), speech recognition, and time series analysis. Its unique ability to use previous outputs as inputs with hidden states allows for advantages such as processing large inputs and accounting for historical information, but requires excessive resources and executes commands one by one.

2.1.1 Architecture of a Traditional RNN

A traditional Recurrent Neural Network (RNN) consists of a single recurrent hidden layer, where the output at each time step is fed back as an input to the network at the next time step. The architecture of a traditional RNN is illustrated in Figure 2.1.

At each time step, a traditional RNN calculates the activation and output based on the input and the previous hidden state. The equations for the activation (h_t) and output (y_t) at time step t are as follows:

$$h_t = \sigma(W_{ih}x_t + W_{hh}h_{t-1} + b_h)$$

$$y_t = \text{softmax}(W_{hy}h_t + b_y)$$

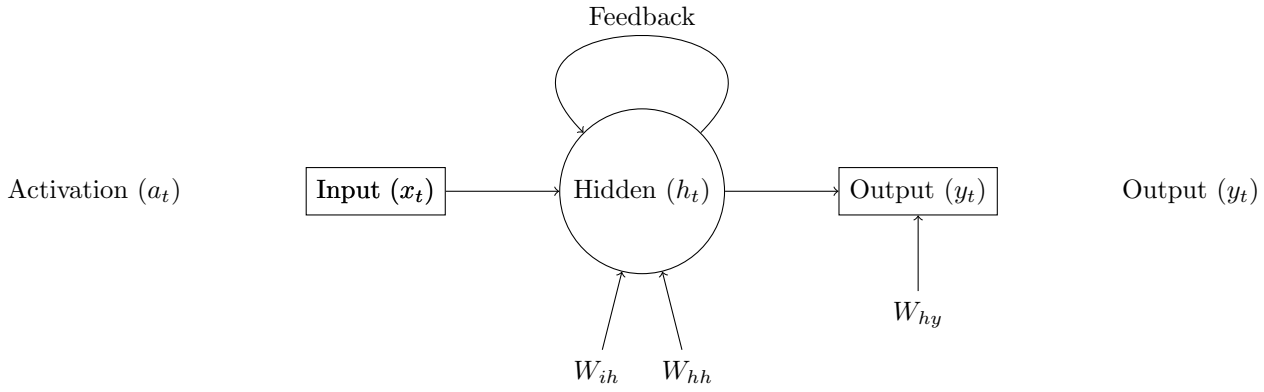

Figure 2.1: Diagram for a Traditional RNN

where x_t is the input at time step t, W_{ih} is the weight matrix for the input-to-hidden connections, W_{hh} is the weight matrix for the hidden-to-hidden connections, b_h is the bias for the hidden layer, W_{hy} is the weight matrix for the hidden-to-output connections, b_y is the bias for the output layer, σ is the activation function (e.g., sigmoid or hyperbolic tangent), and softmax is the softmax function used for multi-class classification tasks.

2.1.2 Applications for Different RNN Architectures

RNN can adjust its input and output ranges to accomplish several different machine learning tasks. In NLP, the one-to-one architecture can complete a sentiment analysis, one-to-many can caption images or generate music, and many-to-many can execute Named Entity Recognition. The appeal behind its architecture has thus been the versatile tasks that can be completed by building neural networks that accommodate for the structure of the objective.

Figure 2.2: One-to-One RNN Architecture

Discussing RNN and its architecture is an excellent primer as it and a majority of models in NLP operate on what is possibly the most important aspect of the Natural Language

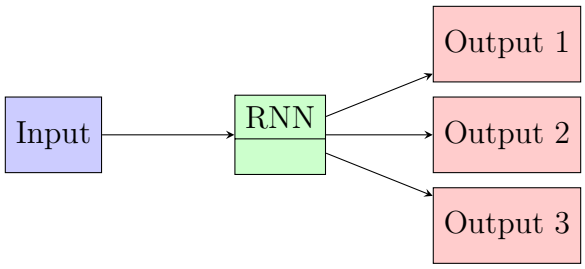

Figure 2.3: One-to-Many RNN Architecture

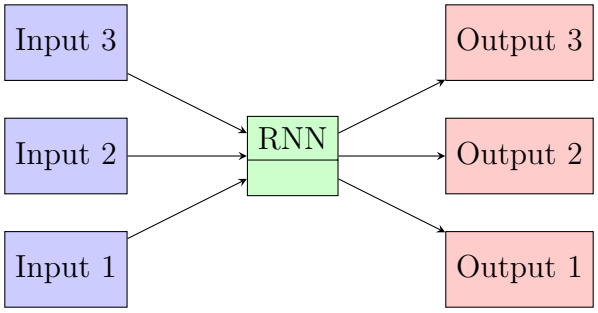

Figure 2.4: Many-to-Many RNN Architecture

Processing field: Word Embeddings and Word Vectors.

2.2 Word Representations in NLP

2.2.1 One-Hot Vectors

One-hot vectors are a simple and intuitive representation for words in NLP tasks. Each word in the vocabulary is represented as a binary vector of length equal to the vocabulary size. The vector has all zeros except for the position corresponding to the index of the word, which is set to 1. This representation indicates the presence or absence of a specific word in a given text.

Figure 2.5 illustrates how the words "cat," "dog," and "bird" can be represented as one-hot vectors in a vocabulary of size 3.

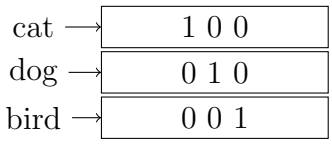

Figure 2.5: One-Hot Vectors

2.2.2 Word Embeddings

Word embeddings are dense vector representations that capture semantic and syntactic relationships between words. Unlike one-hot vectors, word embeddings are continuous-valued and have fixed dimensions.

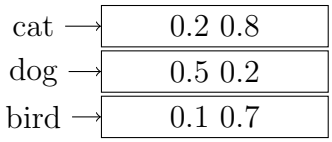

Figure 2.6: Word Embeddings

Figure 2.6 demonstrates the word embeddings for the words "cat," "dog," and "bird." These word embeddings have fixed dimensions, such as 2 in this example.

2.2.3 Word Vectors

Word vectors are continuous representations of words in a vector space, typically learned using techniques like Word2Vec. Word vectors capture semantic relationships between words and are often used in various NLP tasks. This will be especially relevant in the applications of the tasks for this paper such as Topic Modeling and Named Entity Recognition.

Figure 2.7 showcases word vectors represented as circles for the words "cat," "dog," and "bird." These word vectors are often high-dimensional and capture richer semantic relationships.

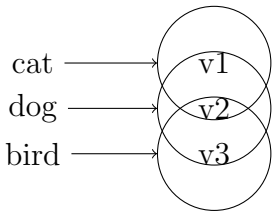

Figure 2.7: Word Vectors

2.3 Word2Vec Framework

Word2Vec is a popular framework for learning word embeddings, which are dense vector representations of words. The framework consists of two main models: Continuous Bag-of-Words (CBOW) and Skip-gram. These models aim to capture the semantic relationships between words based on their context.

2.3.1 CBOW Model

The CBOW model predicts a target word given its context words. It takes a fixed-size context window of surrounding words and tries to maximize the probability of the target word given these context words. The architecture of the CBOW model is illustrated in Figure 2.8.

2.3.2 Skip-gram Model

In contrast to CBOW, the Skip-gram model predicts the context words given a target word. It aims to maximize the probability of the context words given the target word. The architecture of the Skip-gram model is illustrated in Figure 2.9.

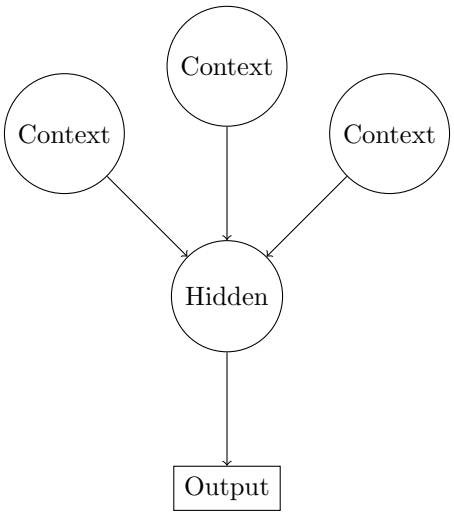

Figure 2.8: Architecture of the CBOW model

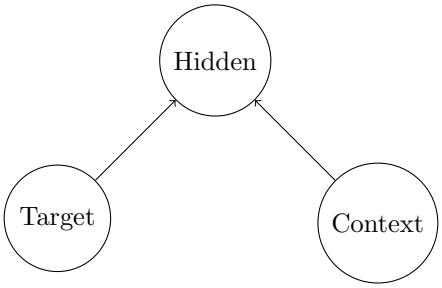

Figure 2.9: Architecture of the Skip-gram model

2.3.3 Training

Both CBOW and Skip-gram models are trained using a neural network with a single hidden layer. The hidden layer represents the word embeddings. During training, the model adjusts the weights of the neural network to minimize the difference between predicted and actual words in the given context.

The Word2Vec framework utilizes the backpropagation algorithm with stochastic gradient descent (SGD) to update the weights. The training process involves iteratively feeding the model with word-context pairs and adjusting the weights based on the prediction errors.

The trained word embeddings capture semantic relationships between words. Words with

similar meanings or appearing in similar contexts tend to have similar vector representations.

2.4 LSTM (Long Short Term Memory)

LSTM became the most recent development to the Natural Language Processing field as a solution to the major set-backs of previous RNN models. Its impact resides mainly in the adjustment of word representations and their iterative processing to capture long term dependencies that have been previously disposed of with other algorithms. The importance of the LSTM model cannot be expressed enough as it became increasingly more advanced over its improvement and usage.

LSTMs are designed to selectively retain or forget information over long sequences, making them well-suited for tasks involving text, such as machine translation, sentiment analysis, and language generation.

The structure of an LSTM consists of a memory cell that preserves information over time, as well as several gates that control the flow of information. The memory cell serves as a memory unit, allowing the LSTM to selectively retain or forget information. The three main gates in an LSTM are the forget gate, input gate, and output gate.

2.4.1 LSTM Cell

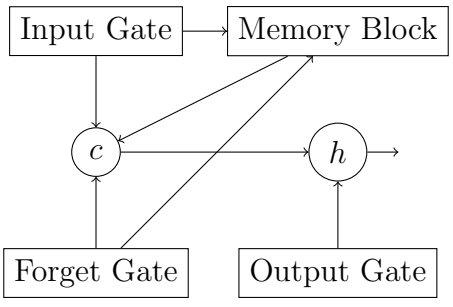

Figure 2.10: LSTM Cell Architecture

2.5 LSTM Architecture

The forget gate determines which information from the previous time step to discard. It takes the previous hidden state h_{t-1} and the current input x_t as inputs, and applies a sigmoid activation function to the weighted sum of these inputs. The output of the forget gate, denoted as f_t, determines how much of the previous cell state c_{t-1} should be retained.

The input gate controls the information to be updated in the memory cell. It takes the previous hidden state h_{t-1} and the current input x_t as inputs, and applies a sigmoid activation function. Additionally, it uses a tanh activation function to determine the values to be added to the memory cell. The output of the input gate, denoted as i_t, determines the amount of new information to be stored in the memory cell.

The output gate regulates the information to be output from the memory cell. It takes the previous hidden state h_{t-1} and the current input x_t as inputs, and applies a sigmoid activation function. The memory cell is passed through a tanh activation function, and the output of the output gate, denoted as o_t, determines the final output of the LSTM.

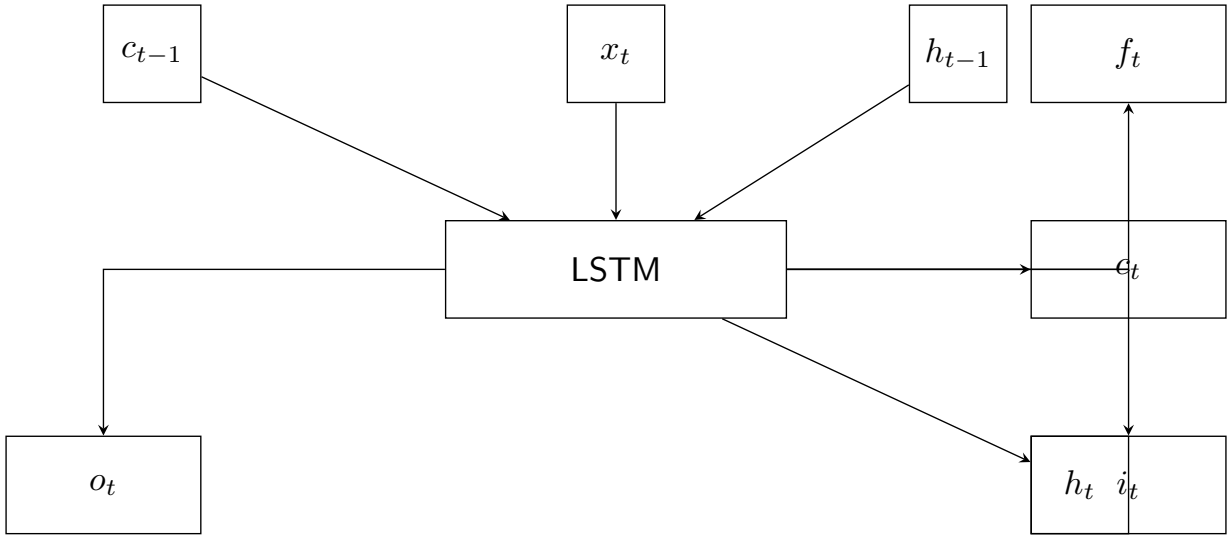

Figure 2.11: LSTM Architecture

Thus, the equations of the LSTM are as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.2}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.3}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.4}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.5}$$

where W_f, W_i, W_c, W_o are weight matrices, b_f, b_i, b_c, b_o are bias vectors, and σ represents the sigmoid activation function.

In summary, the LSTM model in NLP employs a memory cell and gating mechanisms to selectively retain or forget information over time. The forget gate decides which information from the previous time step to discard, the input gate determines the new information to be stored in the memory cell, and the output gate controls the output of the LSTM. These mechanisms enable LSTMs to capture long-term dependencies in text data, making them powerful models for various NLP tasks. This will be especially important to note over the discussion of transformers.

2.6 Training and Backpropagation

The training process of an LSTM involves optimizing its parameters to make accurate predictions on the given task. This is typically done through a process called backpropagation through time (BPTT), which is a variant of the backpropagation algorithm for RNN. Note

that backpropgation is a very common practice in most machine learning methods, but its implementation in LSTM was particularly ground breaking for an NLP model.

BPTT works by unfolding the LSTM over time, creating a sequence of interconnected copies of the LSTM cells. Each LSTM cell in the unfolded network corresponds to a specific time step in the input sequence. The goal is to compute the gradients of the loss function with respect to the parameters of the LSTM, allowing us to update the parameters through gradient descent.

To calculate the gradients, LSTM forward propagates the input sequence through its unfolded network to obtain the predicted outputs. The predicted outputs are then compared to the true targets using a suitable loss function, such as mean squared error or cross-entropy loss.

After calculating the loss, it initiates the backward pass by computing the gradient of the loss with respect to the output of each LSTM cell. These gradients are then used to update the parameters of the LSTM cells.

During the backward pass, the gradient at each time step is not only propagated to the previous time step but also to the future time steps. This is because the LSTM cells are recurrently connected, and the future states depend on the current states. The backward pass continues until reaching the first time step of the input sequence.

As the backward pass progresses, the gradients at each time step are accumulated and used to update the parameters of the LSTM cells. This process is often performed using an optimization algorithm such as stochastic gradient descent (SGD) or its variants (e.g., Adam or RMSprop). The parameters are updated in the opposite direction of the gradients, aiming to minimize the loss function.

The gradient calculation in BPTT involves computing the gradients of the LSTM's internal gates (forget gate, input gate, and output gate) and the memory cell states. These gradients are then used to update the weight matrices and bias vectors of the LSTM cells.

The specific equations and formulas for gradient calculation in LSTMs can be derived using the chain rule of calculus and the specific architecture of the LSTM. This is standard practice for a back propagation process.

It is worth noting that training an LSTM can be computationally expensive, especially for long sequences, due to the unfolding process and the potentially large number of time steps. Techniques such as truncated BPTT, which involves back propagating gradients only over a limited number of time steps, are often employed to mitigate this computational burden while still capturing long-term dependencies in the sequence. This limitation is what prompted the research leading to the renowned transformer model.

CHAPTER 3

Models

3.1 The Transformer Model

After the development of several previous NLP methodologies, the concept of "Attention" revolutionized what has created some of the most advanced AI Language Processors in the world: The Transformer Model. With its ability to learn deeply at a much faster rate than its previous counter-parts, and added capabilities such as parallelization, The transformer model is the new standard for a vast majority of NLP tasks. Its platform to efficiency due to the concept of Attention was discovered by a group of engineers at Google responsible for the publication "Attention is All You Need", which represents a significant departure from previous sequence-to-sequence models, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs), by relying solely on the attention mechanism for capturing dependencies between input and output sequences. Note that even at the heart of some of the most advanced techniques for NLP is the importance of word embeddings and model training to their accuracy.

The Transformer architecture consists of an encoder and a decoder, both composed of multiple layers of self-attention and feed-forward neural networks. The encoder processes the input sequence and produces a set of encoded representations, while the decoder generates the output sequence based on these representations. The attention mechanism facilitates the flow of information between the encoder and decoder layers.

3.1.1 Self-Attention

The self-attention mechanism allows the model to capture relationships between different words or tokens within the same sequence. It computes weighted representations of each word based on the relevance of other words in the sequence. Self-attention can be understood as a process of attending to different parts of the input sequence to gather information for better representation.

In the self-attention mechanism, each word/token in the sequence is represented by three vectors: the Query vector (Q), the Key vector (K), and the Value vector (V). These vectors are derived from the input sequence using linear transformations.

The attention score between two words i and j is computed as the dot product between the Query vector of word i and the Key vector of word j, divided by a scaling factor ($\sqrt{d_k}$). The scaling factor is applied to prevent the dot product from getting too large or small.

The attention weights, representing the importance of each word in the sequence with respect to the current word, are obtained by applying a soft max function to the attention scores. The weighted sum of the Value vectors, using the attention weights as weights, produces the attended representation for the current word.

In addition to the self-attention mechanism, the Transformer model also incorporates position-wise feed-forward neural networks. These networks consist of two linear transformations with a non-linear activation function (usually a ReLU) applied in between. This feed-forward network operates independently on each position in the sequence, allowing for the integration of local context information.

3.1.1.1 Encoder and Decoder

The encoder processes the input sequence and produces a set of encoded representations. It consists of multiple layers, with each layer composed of a self-attention mechanism followed by a position-wise feed-forward network. The output of the final encoder layer is the encoded

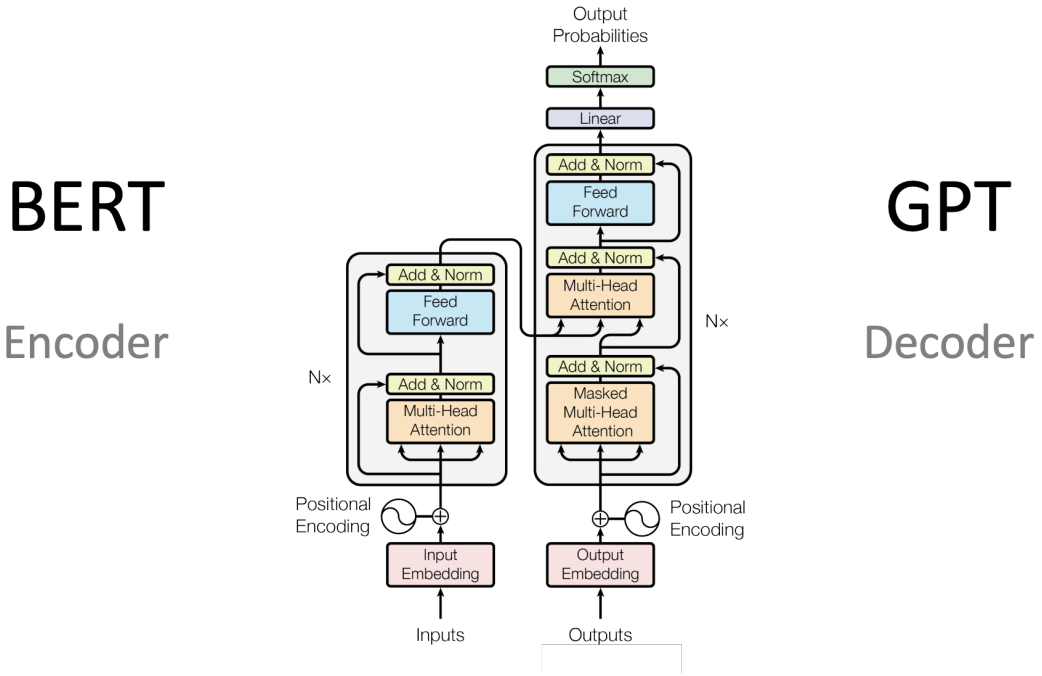

Figure 3.1: Self Attention Matrix Calculation

representation of the input sequence, which captures both local and global dependencies.

The decoder takes the encoded representations from the encoder and generates the output sequence. Like the encoder, the decoder also consists of multiple layers with self-attention and feed-forward sub-layers. However, it includes an additional attention mechanism called encoder-decoder attention.

The encoder-decoder attention allows the decoder to attend to the relevant parts of the encoded input sequence. During each decoding step, the decoder self-attends to the previously generated positions in the output sequence and also attends to the encoded representations of the input sequence. This dual attention mechanism helps the decoder generate accurate and contextually-aware translations.

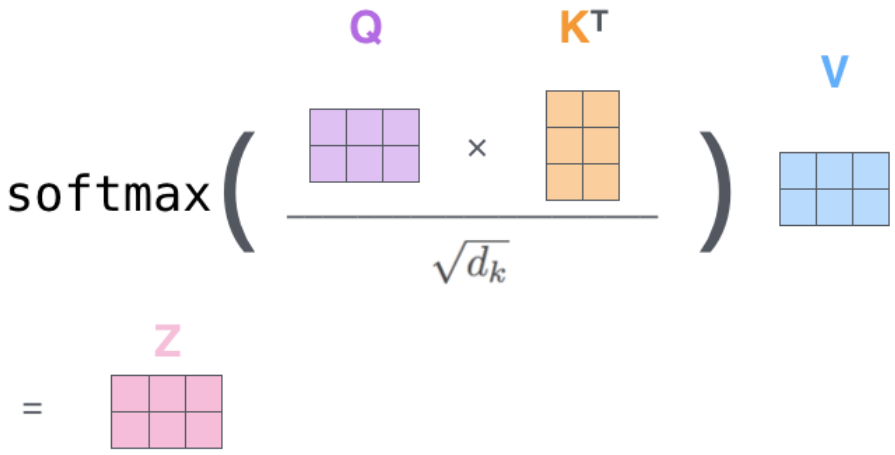

Figure 3.2: Self Attention Matrix Calculation

3.1.2 Matrix Calculation

Matrix calculation is an integral part of the Transformer model. It involves transforming the input sequence into matrices, performing matrix multiplications, and applying non-linear activation functions.

To perform the self-attention mechanism, the input sequence is transformed into Query (Q), Key (K), and Value (V) matrices using learned weight matrices. These matrices are then used to compute attention scores, which are further used to calculate attention weights. The attention weights are applied to the Value matrix to obtain the attended representation.

Similarly, the position-wise feed-forward networks involve matrix multiplications and activations. The input sequence is transformed into a matrix and passed through a linear transformation followed by a non-linear activation function. The resulting matrix is again transformed using another linear transformation to produce the output representation.

3.1.3 Training and Testing

The Transformer model is trained using a variant of the Adam optimizer and the cross-entropy loss function. During training, the model learns to minimize the discrepancy between its predicted output sequence and the ground truth output sequence.

In the training process, teacher forcing is often used, where the model is provided with the correct previous output tokens as inputs during decoding. This helps the model learn to generate accurate translations.

During testing, the model uses its own predicted output tokens as inputs during decoding, creating an auto-regressive process. The decoding is performed iteratively, with each step generating the next token based on the previous tokens until an end-of-sequence token is generated or a maximum sequence length is reached. Thus, the birth of the transformer model and its pipeline to the most efficient NLP models has opened doors to a world of data analysis techniques and information. With access to open-source libraries that have already been trained on relevent data, automated ML techniques have blossomed into an accessible and deployable resource for otherwise difficult tasks.

3.2 Fine-Tuning Pre-Trained Hugging Face Models and BERT

BERT (Bidirectional Encoder Representations from Transformers) stands out as a powerful language model that has achieved remarkable results across a wide range of NLP tasks. As discussed generally with transformer models, it uses the power of self-attention mechanisms to capture contextual information from both left and right contexts of a word. However, unlike previous models that processed text in a sequential manner, BERT is a bidirectional model that considers the entire input sequence at once. This enables it to learn deeply contextualized representations, leading to better performance on various NLP tasks.

BERT is initially pre-trained on large-scale unlabeled corpora, such as Wikipedia or

books, using two unsupervised tasks: masked language modeling (MLM) and next sentence prediction (NSP). During pre-training, BERT learns to predict masked words within a sentence and understand the relationship between two consecutive sentences.

After pre-training, the pre-trained BERT model can be fine-tuned for specific NLP tasks. Fine-tuning involves training the model on a smaller labeled dataset that is task-specific. This allows the model to adapt its learned representations to the specific nuances and requirements of the target task.

3.2.1 Sponsored Content Classification

For classification tasks, the transformer model focuses on the encoder part of its architecture. As represented above, the encoder consists of multiple layers, each comprising a multi-head self-attention mechanism and position-wise feed-forward neural networks. These layers enable the model to capture both local and global dependencies, creating a holistic understanding of the input sequence

Unlike sequential models like RNNs, the Transformer model captures global context efficiently. With self-attention, each position attends to all other positions, generating attention weights that reflect the significance of each position in relation to the current position. This global context representation empowers the Transformer to make informed predictions based on a comprehensive understanding of the entire sequence.

Given an input sequence $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, the self-attention mechanism computes the query, key, and value matrices:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V$$

where $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are learnable weight matrices.

The context matrix, representing the contextualized representation of the input sequence, is computed using the self-attention mechanism as discussed earlier:

$$\text{Context}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$$

On top of the Transformer encoder, a classification head is added for the classification task. It typically consists of a fully connected layer followed by a softmax activation function.

The final hidden state $\mathbf{H}$ from the Transformer encoder is used as input to the classification head, producing class probabilities:

$$\text{Class Probabilities} = \text{softmax}(\mathbf{H}\mathbf{W}_C)$$

where $\mathbf{W}_C$ is the weight matrix of the classification head.

In this application, the methodology will be applied to the word corpus of episode descriptions in Spotify Podcasts, to determine whether the description and resulting episode contains sponsored content.

3.2.2 Topic Modeling

Recently, Transformer models have demonstrated remarkable success in various NLP tasks, including topic modeling. BERTopic is a topic modeling approach based on the powerful BERT (Bidirectional Encoder Representations from Transformers) model, which combines the power of the BERT model with clustering algorithms to perform topic modeling. Similar to its ancestors in topic modeling, BERTopic still uses the TD-IDF approach.

In traditional TD-IDF, TF measures the frequency of a term within a document. It quantifies how often a term appears in a document relative to the total number of terms in that document. It can be calculated using the following equation:

$$\text{TF}(t, d) = \frac{\text{Count}(t, d)}{\text{Total terms in } d}$$

where t is a term and d is a document.

IDF measures the informativeness of a term across the entire document corpus. It penalizes terms that appear frequently across multiple documents, as they are considered less informative. IDF can be calculated using the following equation:

$$\text{IDF}(t) = \log\left(\frac{\text{Total documents}}{\text{Number of documents containing } t}\right)$$

where t is a term.

TF-IDF is the product of the TF and IDF values. It provides a weight that reflects both the frequency of a term within a document and its significance in the entire corpus. The TF-IDF value for a term t in a document d can be calculated as follows:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

The TF-IDF value is higher for terms that appear frequently in a particular document (high TF) and infrequently across the entire corpus (high IDF).

As opposed to this traditional, but static, technique; BERTopic is a dynamic topic modeling technique that leverages the concept of c-TF-IDF (class-based TF-IDF) to model topics in a document corpus.

c-TF-IDF is an extension of the traditional TF-IDF approach that considers the context of a term within a specific class or topic. It calculates the importance of a term within a topic by considering its frequency in documents belonging to that topic relative to its frequency in the entire corpus. c-TF-IDF can be calculated using the following equation:

$$\text{c-TF-IDF}(t, \text{topic}) = \text{TF}(t, \text{topic}) \times \text{IDF}(t)$$

where t is a term and topic is a specific topic.

This approach recognizes that the same topic can appear across different times, but with

potentially different word representations. It aims to generate global and local representations of topics that capture the underlying themes across different time periods.

To create a global representation of topics, regardless of their temporal nature, BERTopic is first fitted on the entire corpus without considering the temporal aspects. This process generates a global view of topics, where certain words like "car" and "vehicle" may be associated with a general topic about cars.

To account for the temporal variations within topics, BERTopic develops a local representation for each topic. This is achieved by multiplying the term frequency (tft) of documents at timestep i with the pre-calculated global IDF (Inverse Document Frequency) values. The formula for the local representation is as follows:

$$W_{t,c,i} = \text{tft}_{t,c,i} \cdot \log(1 + \text{A}_{tft})$$

where $W_{t,c,i}$ represents the local representation of term t in topic c at timestep i. $\text{tft}_{t,c,i}$ is the term frequency of term t in topic c at timestep i, and A_{tft} is a scaling factor.

A major advantage of using c-TF-IDF representations in BERTopic (compared to its LDA, Top2Vec, and NMF alternatives) is the ability to create local representations without the need for embedding and clustering documents, which results in faster computation. Additionally, this technique can be extended to model topic representations based on other metadata, such as author or journal, allowing for more nuanced analysis and interpretation.

3.2.3 Named Entity Recognition

Named Entity Recognition (NER) plays a crucial role in extracting and categorizing named entities from text. In recent years, transformer-based architectures, such as BERT, have shown promising results in NER tasks.

Fine-tuning approaches, unlike traditional LSTM-based approaches, incorporate a single linear layer into the transformer architecture and fine-tune the entire model on the NER

task. By leveraging the power of the transformer, these approaches model both subtoken representations and token-level predictions in a single architecture. This holistic approach offers the advantage of capturing complex contextual relationships and improving overall performance.

To bridge the gap between subtoken modeling and token-level predictions, fine-tuning approaches employ subword pooling techniques. This process aggregates subtoken representations to create token-level representations, which are then fed into the final linear layer. By combining subtoken information, these approaches effectively handle variable-length named entities and improve the accuracy of entity recognition.

Furthermore, fine-tuning approaches benefit from the ability to leverage document-level features. In BERT-based models, document-level features are obtained by including the surrounding context of a sentence during training. By considering a context window of left and right subtokens, the model captures the broader context within which each sentence resides. This approach ensures that the model can understand the influence of neighboring sentences, providing a richer representation of the named entities.

In contrast to fine-tuning, traditional LSTM-based approaches use the transformer solely to generate word embeddings, which are then fed into a standard sequence labeling architecture such as LSTM-CRF. This feature-based approach allows for a well-understood training procedure but lacks the holistic modeling capability of fine-tuning. Additionally, in feature-based approaches, the transformer weights are frozen, limiting the model's ability to adapt to specific NER tasks. Thus, a combination of document-level features and fine-tuning on the podcast corpus will likely product measurable results at a much faster rate.

CHAPTER 4

Results

4.1 Spotify Podcast Sponsored Content Narrative

The Spotify podcast dataset provides valuable information about various podcasts available on the platform. With 464.7 million global podcast listeners in 2023, reaching a predicted 504.9 million by 2024, the podcast industry is experiencing rapid growth. Additionally, the market size of the podcast industry stands at 23.56 billion, indicating its significant impact and potential for various stakeholders, including content creators, publishers, and advertisers.

Sponsored content involves collaborations between podcast creators and brands to promote products, services, or events. By utilizing named entity recognition (NER) and topic modeling techniques, we can delve into the dataset to identify instances of sponsored content and explore the associated brands. First, the sponsored content needs to be classified.

Using a fine-tuned BERT classification model for sponsored content, we can filter episode descriptions in the metadata by looking for any promotions/sponsored information and creating a new corpus for other applied techniques.

Using NER, we can extract relevant entities such as brand names, publishers, and podcast names from the dataset. The "publisher" field provides insights into the podcast creators and the companies behind them. This information helps us identify the podcast publishers involved in producing the content and potentially attracting sponsorships.

The "episodename" field contains the names of individual podcast episodes, and the "episodedescription" field provides descriptions of those episodes, as fore mentioned. Ana-

lyzing this information enables us to identify specific episodes that may feature sponsored content or discussions related to brands. By considering the duration of the episodes, we can also explore the duration of sponsored segments, potentially indicating the level of integration and promotion within the episodes.

To gain insights into the brands associated with sponsored content, we can leverage topic modeling techniques. By applying topic modeling algorithms to the dataset, we can identify recurring topics or themes across podcasts and episodes. These topics can include brand-related discussions, product reviews, or specific industry-related content. By examining the keywords and context associated with these topics, we can pinpoint the brands being mentioned, promoted, or discussed in the podcast episodes.

As the dataset includes information such as show URIs, RSS links, and episode URIs, we can leverage these identifiers to establish connections and track the presence of sponsored content across different episodes and shows, though these identifiers will be used less in the analysis.

By combining NER, topic modeling, and the provided dataset, we can create a comprehensive narrative showcasing sponsored content and the brands associated with it. This analysis will shed light on the podcast landscape, reveal potential opportunities for advertisers, and provide insights into the evolving relationship between content creators and brands in the podcasting industry.

With the growing podcast market and the increasing adoption of branded content, understanding the dynamics of sponsored content in podcasts is crucial for advertisers, podcast creators, and listeners alike. This analysis will contribute to a deeper understanding of the role of sponsored content, highlight successful brand integrations, and inform future strategies for content monetization and advertising within the podcasting space.

4.2 Exploratory Data Analysis

The data set will consist of several points of comparison even in its raw form, but as mentioned before- the objective of this project is to classify the data set into sponsored/not sponsored content and to look individually at the newly filtered data set with only sponsored episodes for differences between its original counterpart. Additionally, NER will aid in detecting brands within the sponsored content that are most frequently mentioned.

Assessing the impact of each podcast on the data set as a whole will help visualize how different podcasts will push or pull against the information that will be retrieved from the classification model. The corpus will be built off of the "episode description" column of the data set, to assess whether a particular episode is being sponsored. This means there will be several episodes for a particular podcast's show- not all of which could be sponsored. The overall frequency of the podcast episodes per show will help understand which shows are prevalent and whether these will be the same shows that show up in the data set that will only consist of sponsored data.

4.2.1 Average Podcast Duration

The duration of a podcast has historically been linked to its popularity, as shorter podcasts are generally more digestible and accessible to its consumers. Though this dataset consists of 90% small creator shows and only 10% large creators, a 20 minute podcast seems to be the standard for most media of this kind.

4.2.2 Most Frequented Podcasts

The highest podcasts by frequency provide information on which shows will likely show up most often in both sponsored and non-sponsored data sets (Figure 4.2).

4.2.3 Word Clouds

A majority of this project is assessing the semantic relationship between words and their impact on the corpus for sponsored content. Thus, the preliminary step to visualizing the language content of a corpus is to visualize it through word clouds.

Individual Episodes have a heavy emphasis on the mention of "Anchor FM", note that this particular platform sponsors small podcast creators to discuss brand opportunities with Spotify. Anchor FM itself is interestingly mentioned most often in the episode corpus as opposed to show, indicating many creators being sponsored by the platform on an episode basis.

The show word cloud depicts a consistent mention of "support". Note that this will mostly include support requests and links to episodes and shows on other platforms to competitors, which is still impactful to the analysis and classification model.

Identifying the most frequent words will help in building and preparing the corpus for the classification input. Stop words and any insignificant words should be removed, but BERT has the capacity to tokenize and adjust regardless of whether the corpus is cleaned. Still, it will help with processing and context to be aware of these words.

4.3 Corpus Analysis: Topic Modeling

Topics were widely ranging but depict what is expected of most podcast content. The highest topic scores were for fitness and nutrition, and this was the case for sponsored content as well. Fitness and nutrition brands may find it worthwhile to invest in segments with podcast content that aligns with their product. While not depicted in this paper, a cluster analysis found that podcast content has a wide range and variety of topics within discussion, but it appears that most topics can fall within the range of the four plotted bar graphs from BERTopic modeling. The words were highly associated with each other and

appear frequently in their respective "documents" from the corpus. The confusion matrix in Figure 4.7 shows the similarities between different key words and topics.

4.4 Modeling Results and Analysis: Sponsored Content Classification

Fine-Tuning the Model successfully provided a weighted accuracy of 92% on the test set. The model can continue to improve with proper training and additional adjustments to the model structure. The resulting dataset was filtered with a remainder of 66% being classified as sponsored content. This can include self-branding where podcasters encourage their audience to visit competitor websites through their links as well as social media sites. This can also be further adjusted to exclude links and self promotion by either cleaning the corpus or by creating a more sophisticated model and inputting audio data from podcast recordings. That being said, this still falls under the category of sponsorships and will not appear often enough to overthrow any big brands that are mentioned for actual sponsorship deals. The F-1 score is satisfactory against false negatives but not quite there for false positives, though for detecting sponsorships having a higher rate of false negatives is worse as a trade-off.

Classification models using BERT save significant run-time and computational resources. This model was deployed and completed on the test set (80/20 split) within 5 hours via high RAM and advanced GPU. At maximum, the model would take 10 hours to tokenize and process, compared to previous classification methods- which could take up to 25 hours (the experiment was attempted against an alternate model but resources were limited).

4.5 Modeling Results and Analysis: Named Entity Recognition

Comparatively, the Named Entity Recognition model used its input of sponsored content after the classification was complete. Its accuracy F-1 score was 96%, which was sufficient for the problem. Table 4.2 portrays each fluctuating and final classification F-1 score by label: LOC (location), MISC (miscellaneous), ORG (organization), and PER (person). The resulting dataset was filtered for ORG (organization) labels only, and the frequency by brand was calculated to observe which brands were mentioned most frequently in sponsored content. Surprisingly, despite its frequent appearance and impact in the word clouds, Anchor FM was third on the list after Amazon and Spotify, though these could each have their respective context (Amazon storefronts, products, Spotify being the platform on which the podcast is being hosted), which is could be investigated further with a multimodal analysis of the audio content and written descriptions of each podcast.

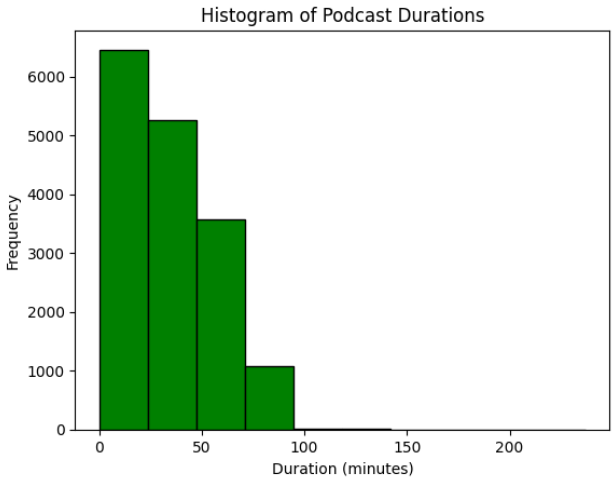

Figure 4.1: Average Duration for Podcast was Roughly 20 Minutes

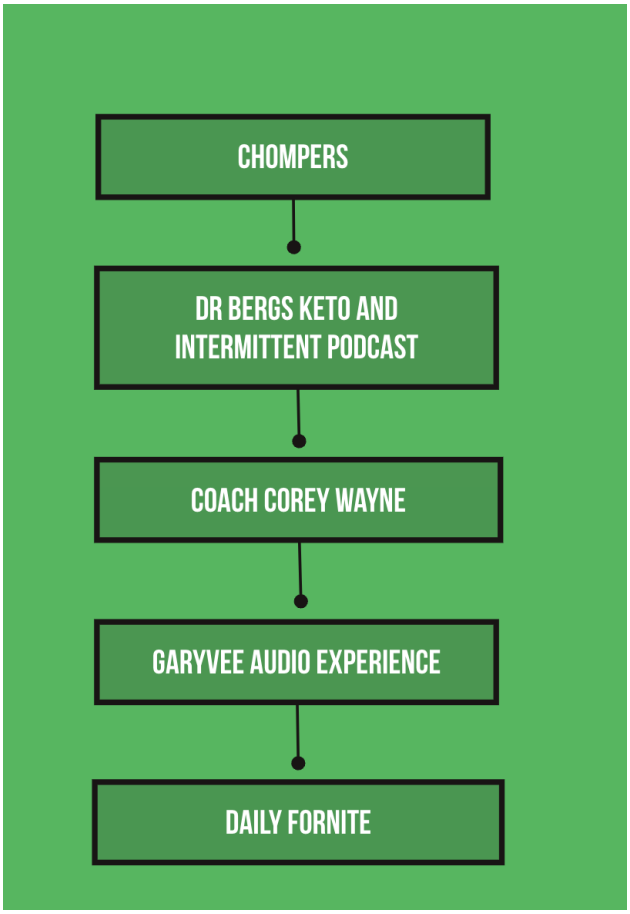

Figure 4.2: Top Five Shows with Highest Episode Frequency

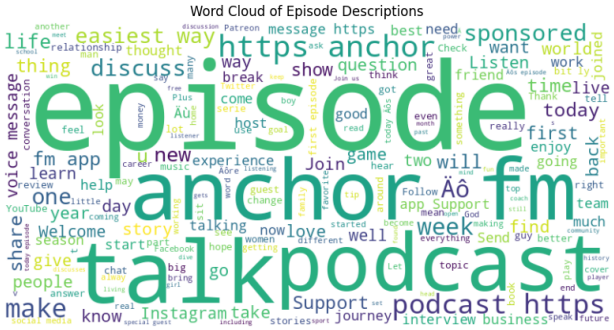

Figure 4.3: Word Cloud Visualization for Episode Corpus

Figure 4.4: Word Cloud Visualization for Show Corpus

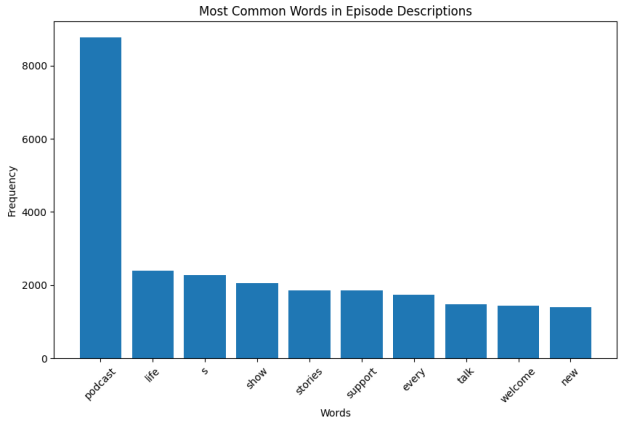

Figure 4.5: Most Mentioned Words in Corpus

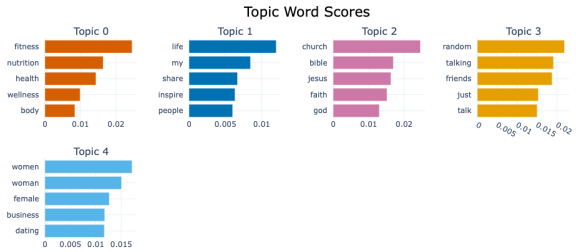

Figure 4.6: BERTopic Word Scores

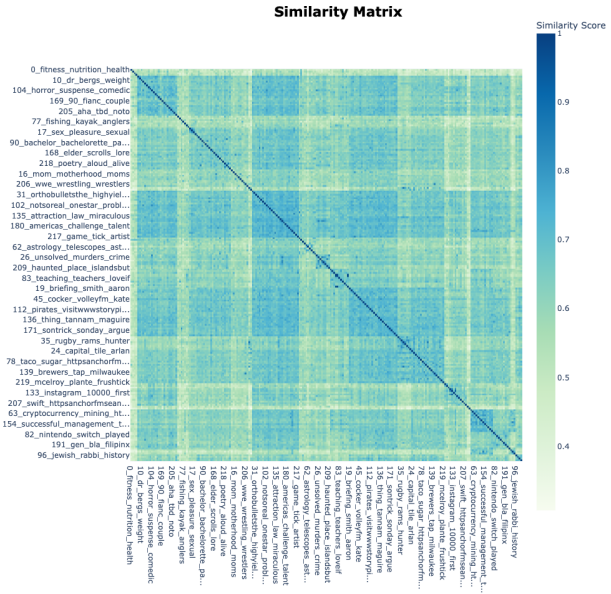

Figure 4.7: BERTopic Similarity Matrix

	Precision	Recall	F1-Score
0	0.95	0.93	0.94
1	0.88	0.91	0.89

Table 4.1: Classification Results on Test Set

Entity	F-1 Score Change	Accuracy
LOC	+0.42	0.94
MISC	+0.21	0.91
ORG	+1.20	0.95
PER	+1.18	0.93

Table 4.2: Named Entity Accuracy Results

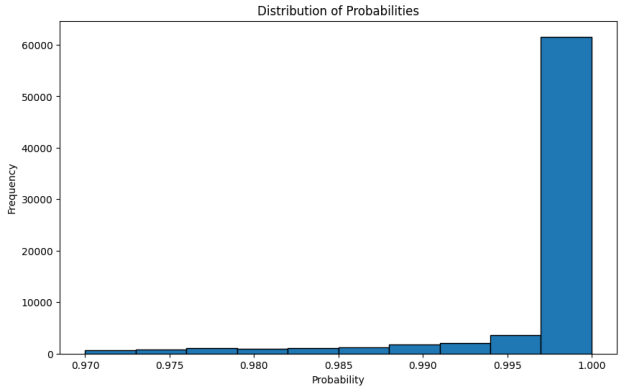

Figure 4.8: Distribution of Organization Labels

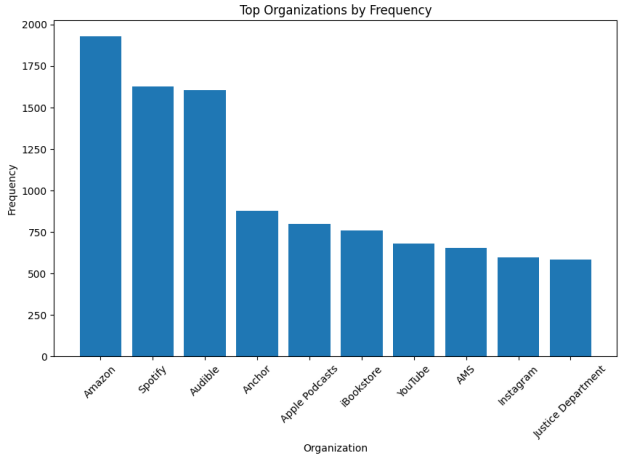

Figure 4.9: Top Organizations Mentioned in Sponsored Content

CHAPTER 5

Conclusion

By exploring the evolution of Transformers in the field of Natural Language Processing (NLP), this endeavor demonstrates their significant impact on data analysis and narrative creation. By applying various techniques, such as Classification, Named Entity Recognition (NER), and Topic Modeling, to a dataset of 100,000 Spotify podcasts and their descriptions, valuable insights were gained for the marketing field in the context of podcasts and depicts potential for far beyond.

The application of fine-tuned Transformer models, particularly BERT, proved to be highly effective in classifying podcasts into sponsored and non-sponsored categories. This classification provided a foundation for further analysis and comparison between the two categories. NER techniques allowed for the extraction of important entities, such as brand names, publishers, and podcast names, enabling a deeper understanding of the relationships between content creators and brands in the podcasting industry. Additionally, Topic Modeling revealed recurring themes and discussions related to brands, product reviews, and industry-specific content, facilitating the identification of brands associated with sponsored content.

The findings from the dataset and the models employed highlight the usefulness and convenience of fine-tuning models in NLP tasks. The ability to process and understand large amounts of textual data, such as podcast descriptions, will open up new avenues for analysis, allowing for more comprehensive narratives and informed decision-making.

The dataset analysis also provided valuable insights into the podcast landscape and the

growing influence of branded content. As the podcast industry continues to experience rapid growth, understanding the dynamics of sponsored content is crucial for advertisers, podcast creators, and listeners alike. By uncovering successful brand integrations and informing future strategies for content monetization and advertising, this research contributes to a deeper understanding of the evolving relationship between content creators and brands in the podcasting space.

In conclusion, the combination of fine-tuned Transformer models for NLP techniques offers a powerful toolkit for extracting valuable information from textual data and creating meaningful narratives. The advancements in NLP, driven by the capabilities of Transformers, have paved the way for more robust analyses and deeper insights into various domains. As the field continues to evolve, the ability to effectively utilize these tools will remain a valuable skill set for data scientists and practitioners seeking to harness the full potential of the ever-growing data ecosystem.

REFERENCES

[1] Stefan Schweter and Alan Akbik. *FLERT: Document-Level Features for Named Entity Recognition.* CoRR, abs/2011.06993, 2020.

[2] Ann Clifton et al. *100,000 Podcasts: A Spoken English Document Corpus.* In Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, Spain (Online), December 2020.

[3] Edgar Tanaka et al. *Cem Mil Podcasts: A Spoken Portuguese Document Corpus.*, 2022.

[4] Ashish Vaswani et al. *Attention Is All You Need.* CoRR, abs/1706.03762, 2017.

[5] Lorenzo Vaiani et al. *Leveraging Multimodal Content for Podcast Summarization.* In Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, SAC '22, New York, NY, USA, 2022. Association for Computing Machinery.

[6] Maarten Grootendorst. *BERTopic: Neural topic modeling with a class-based TF-IDF procedure.*, 2022.