Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MASTERTHESIS

Identifying Advertisements in Podcasts

Felix V. J. Wolf

Field of Study: Master Informatik
Matriculation No.: 7199604
1st Examiner: Dr. Seid Muhie Yimam, Universität Hamburg
2nd Examiner: Prof. Dr. Chris Biemann, Universität Hamburg

Language Technology
Department of Informatics
Faculty of Mathematics, Informatics and Natural Sciences

University of Hamburg
Hamburg, Germany

A thesis submitted for the degree of
Master of Science (M. Sc.)

Submitted on July 22, 2025

Identifying Advertisements in Podcasts

Master's Thesis submitted by: Felix V. J. Wolf

Date of Submission: July 22, 2025

Supervisors:
Hans Ole Hatzel, Universität Hamburg
Robert Geislinger, Universität Hamburg

Committee:
1st Examiner: Dr. Seid Muhie Yimam, Universität Hamburg
2nd Examiner: Prof. Dr. Chris Biemann, Universität Hamburg

University of Hamburg, Hamburg, Germany
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics

Language Technology

Abstract

As podcasting developed from a passion project for hobbyist to a vastly popular entertainment medium, advertiser spending increased to nearly $2 billion by 2024. Today, podcasting is a popular form of entertainment, with close to 100 million users in the United States. Advertisement identification in podcasts is a challenging tasks due to the heterogeneous nature of content and advertisements. Being directly embedded into the audio stream, ad identification requires the segmentation of content based on features extracted from the raw audio signal as well as the transcriptions. For a lack of readily available podcast datasets with advertisement annotations, we create a dataset of popular podcast shows in the U.S, manually annotating over 150 episodes. As manual dataset annotation is tedious and time-consuming, we leverage annotations from an open source ad database for YouTube videos to build a second, larger dataset of podcast-like YouTube videos. Upon exploration of the created dataset, we find differences in ad-related statistics between sources, e.g. number of advertisements per episode, number of advertisements in a row and advertisement type and duration. We introduce the *local* classifier model architecture that uses the multimodal Transformer model LanguageBind to generate embeddings from audio and text data to classify single-sentence input samples from the in-domain audio podcat dataset. Multimodal embeddings outperform embeddings computed from single modalities. We also concatenate input samples to increase model context in the *superlocal* architecture, unable to meaningfully improve of single-sentence results. Due to factors like subpar annotation quality and lack of advertiser-produced advertisements in the out-of-domain YouTube dataset, training models on out-of-domain data to transfer learned feature characteristics for in-domain inference proved to be unsuccessful.

Contents

1 Introduction

Once started as a passion project for hobbyist in the early 2000s, podcasting has turned into a multi-billion-dollar market over the last 25 years. Joe Rogan's recent renewal of his Spotify partnership, reportedly valued at $250 million USD, highlights podcasting's evolution into a major and lucrative industry (Spangler 2024). As the industry continues to grow, more creators, advertisers, and media companies are investing in podcasts, making it a significant force in the global entertainment landscape.

Podcasts as a medium are audio documents containing speech content. They are typically distributed over really simple syndication (RSS) feeds and consumed via dedicated podcast players or more generic music distribution services like Spotify and Apple Podcasts. Watching video podcasts on video platforms like YouTube is increasingly popular. Podcasts are characterized as covering "[…] a wide range of topics in a highly flexible format that contains monologues, multi-dialogues, interviews, or discussions" (Feldstein Jacobs 2022).

The history of podcasts is closely related to that of several technologies in the early 2000: portable music players (mp3 players), audio compression and the RSS feed (Pot 2013). In late 2000, the proposed specifications for the RSS standard introduced the ability to include audio or video files, allowing users to subscribe to more than just text content. In 2001, Apple introduced the iPod, a portable music player that would accelerate several shifts in the music industry. Apple would sell close to 500 Million products of the iPod family (Mickle 2022). In 2005, podcasts were added as a category to the Apple iTunes Store, cementing its popularity (Pot 2013). The term podcast is a blend of the words 'iPod' and 'broadcast', hinting at its root in the Apple ecosystem and the technology behind it (Dictionary.com 2025).

For most of its history, listening to a podcast meant listening to an audio file distributed via RSS and consumed by a podcast player. Nowadays, podcasts are not just limited to the audio format, with video podcasts steadily rising in popularity (Mayer

and Riismandel 2023; Escandon 2024). This change is pushed on one hand by young consumers who prefer video over audio (Coats 2024) and on the other by the fact that the video format offers additional monetization opportunities over audio only such as visual advertisements and sponsorships (Steele 2024). With the prevalence of video podcasts, distribution of podcasts today is typically done via these channels: YouTube, Spotify, Apple Podcasts and traditional RSS feeds.

Over the past five years, the popularity of podcasts in the U.S. has steadily risen, from 50 million users of podcast in 2020 to 76 million in 2024 and the number of users is predicted to exceed 100 million by 2029 (Statista 2024). In addition, the number of available podcast shows listed on public servers have doubled from 2019 to 2021 to a total of 48 million episodes (Feldstein Jacobs 2022). The growing popularity of podcasts has turned them into a valuable platform for advertisers, as they also often target niche and highly engaged audiences. In the U.S., advertising spending on podcasts surged 2.5 times, rising from $840 million in 2020 to $1.9 billion in 2023 (PwC; IAB (U.S.) 2024).

1.1 Motivation

As podcasts popularity increases and advertisers rightly identify podcasts as a profitable advertising medium, the study of advertising in podcast and possible detection mechanisms can be motivated from several directions.

First, it involves challenges similar to related fields of research, such as audio classification and content segmentation, speaker diarization and NLP, as podcasts are inherently spoken language documents, with advertisements often being different in content, speaker, tone and background music. Second, advertisers have an economical interest in the question of how often their ads are played and where they appear, as modern technologies like dynamic ad insertion (DAI) obfuscate this information. Third, podcast users are set to benefit from improved user experience and accessibility features, as automatic ad detection in podcasts could be used to skip ads altogether, allowing users to listen to their podcasts without interruptions. Accessibility wise, podcasts, like every audio medium, are difficult to index, as one cannot skim over a podcast episode like it can be done with a piece of text. Therefore, summaries of podcast are highly useful, which in turn should be content-agnostic from their ads.

Altering episode content and skipping advertisements introduce legal and ethical considerations, as the ever-ongoing cat-and-mouse game between ad blockers and ad agencies in web browsers show (Iqbal, Shafiq, and Qian 2017). This thesis explores possible machine learning architectures to detect advertisements in podcast data and will not consider legal and ethical concerns, as no ready-made application is provided.

In modern-day podcasts, characteristic of both content and advertisements are highly heterogeneous. Content can be anything from professionally told stories with sound effects and background music to informal conversations between multiple speakers and everything in between. Ads typically included in podcasts differ in length, style and topic, making them an interesting source of material for speech processing.

This master's thesis explores podcasts as a resource for language science related tasks. The main focus lies on creating a novel dataset of timestamped advertisements in podcasts, fetched from two different sources. The dataset is used to train basic as well as advanced classification algorithms, drawing conclusions from received results. We will also discuss how OOD data can be leveraged to train models for in-domain tasks.

1.2 Research Questions

To guide us through the topic, the following three central research questions (RQs) were formulated:

RQ1: How can advertisements in podcasts be characterized and what differentiates advertisements in audio podcasts from advertisements in video podcasts?

RQ2: How does model performance vary when using audio, text, or multimodal training data?

RQ3: Can training data from video podcasts be used to detect advertisement in audio podcasts?

1.2.1 RQ1: Advertisement Characteristics

This question involves inspecting the generated dataset and gaining insights about key advertisement metrics, like the position within an episode, the length and whether the advertisement is part of a longer ad break. We will visualize the metrics for better interpretability.

1.2.2 RQ2: Modality Information Gain

Since podcasts are not text document but rather audiovisual, it begs the question if we can use the information contained in the audio-signals to improve the classification of the text modality and vice versa. If the advertisements inspected in RQ1 not only differ from the content on the text level but also in the signal level, we can expect a performance increase when using the audio modality as additional features for our models. We will test this hypothesis by using the multimodal, Transformer-based model LanguageBind (Zhu et al. 2024) and comparing classifiers trained on embeddings generated from only text, only audio or multimodal input data.

1.2.3 RQ3: Out-of-Domain Training Data

To our knowledge, there is no dataset available of timestamped advertisements in podcasts. Researchers at Spotify released a dataset of 100,000 podcast episode transcriptions in 2020 in an effort to provide the scientific community with data to solve NLP tasks like chapterization (Clifton et al. 2020). The company has since ceased providing access to the dataset (Spotify 2023a) and it is unclear whether and to what extent it contained advertisements.

In this thesis, we will create our own dataset. A smaller subset of the data will be manually labeled podcast data sourced from RSS feeds, while the majority of the data will be crawled from YouTube, utilizing the community sourced advertisement annotation from the SponsorBlock project (Ajay Ramachandran 2024).

While using YouTube to build the dataset is efficient, it has to be evaluated if YouTube-sourced data is suitable as a replacement for real podcast data for training. To answer this, we will train classifiers on both real podcast data and YouTube-sourced videos and compare model performances.

Throughout the thesis, we will often make the distinction between audio podcasts from RSS feeds and YouTube videos covering mostly, but not exclusively, video podcasts. Going forward, we will call podcasts from RSS feeds *audio podcasts* and podcasts from YouTube *video podcasts*.

1.3 Overview

This thesis is structured as follows: Chapter 2 covers key concepts that are used in the thesis. In Chapter 3, we provide an overview of the previous work in the field of automatic ad detection in long form audio. Subsequently, in Chapter 4, we define the precise problem statement. Chapter 5 introduces the approach performed in this thesis to tackle the problem and answer the three outlined research questions. In Chapter 6, we explore the built dataset. In Chapter 7, the results of this thesis are presented. Chapter 8 holds the discussion and limitations of the results, before Chapter 9 will end with the conclusion.

2 Background

This chapter introduces key concepts employed throughout the thesis. We start by looking at how natural language processing (NLP) has evolved over the years, before diving deeper into neural networks, evaluation metrics and advertisements.

2.1 Natural Language Processing - A Brief History

The history of NLP and therefore the understanding and processing of natural language can be divided into the era of rule-based systems, the era of statistical methods and the still current era of artificial intelligence (Hirschberg and Manning 2015; Johri et al. 2021).

NLP "is the subfield of computer science concerned with using computational techniques to learn, understand, and produce human language content" (Hirschberg and Manning 2015). The first notable event was in the 1950s, when the Georgetown-IBM system demonstrated machine translations of more than 60 sentences from Russian to English (Hutchins 2004), motivated by the need of understanding the enemy during the Cold War.

The first period of NLP research was defined by researchers manually writing down vocabularies and rules of human languages. This proved to be difficult due to the inherent ambiguity and context-dependent interpretation of human languages (Hirschberg and Manning 2015). The goal was aiding human-human communication, especially via machine translation.

In the late 1980s and starting in the 1990s, researchers began to build statistical models over larger quantities of empirical language, creating so-called language corpora. This point in time marks the "first notable successes of the use of big data, long before the power of ML was more generally recognized or the term 'big data' even introduced" (Hirschberg and Manning 2015). The central objective during this time has been the research in tasks like part-of-speech (PoS) tagging and named-entity

recognition (NER), essentially understanding the semantics of languages on the word level. Training statistical models via large corpora proved to be highly effective, with many classifiers today being based on those methods.

The method of leveraging machine learning (ML) to enhance NLP marked a significant turning point. With the advent of ML techniques, particularly deep learning, the field of NLP was revolutionized (LeCun, Bengio, and Hinton 2015). In the early 2010s, researchers began developing models that could automatically learn from large datasets, removing the need for manual feature engineering. ML, and specifically techniques like artificial neural networks (ANNs), allowed systems to handle complex language tasks such as sentiment analysis, translation and question-answering with high accuracy.

Leveraging vast amounts of readily available textual data and the increasing power of computational resources, these learning algorithms enabled NLP systems to generate richer and more nuanced representations of linguistic content. Implementations of such models, notably recurrent neural networks (RNNs) (Schmidt 2019) and Transformers (Vaswani et al. 2017), demonstrated the ability to capture long-range dependencies in text, understanding context beyond individual words or sentences.

The emergence of pre-trained language models, such as BERT (Devlin et al. 2019) and GPT, further improved the power of transfer learning in NLP. These models are trained on comprehensive corpora and fine-tuned for specific tasks, achieving state-of-the-art results by understanding not only the syntax but also the underlying semantics of language. The implementation of these models in various applications, from voice assistants (Jin et al. 2023) to automated customer service bots (Yu, Chen, and Zaidi 2021), highlights the ongoing evolution and impact of artificial intelligence on the field of NLP.

2.2 Artificial Neural Networks

An artificial neural network (ANN) (later referred to as NN) is a computational model designed similar to the biological brain (Goodfellow, Bengio, and Courville 2016). It consists of multiple artificial neurons that work together to accomplish a task. In the broadest sense, the task is the approximation of some function f^*. A classifier $y = f^*(x)$ maps an input x to a category y. The network defines a mapping $y = f(x; \theta)$ and learns the values of the parameters θ that best approximate the relationship between x and y (Goodfellow, Bengio, and Courville 2016).

The neurons in a neural network (NN) are often grouped together in layers. A network typically has an input and an output layer, with a variable number of hidden layers between the input and the output. The neurons in the different layers have connections to neurons on the same or other layers, depending on the network's architecture. Artificial neurons take real numbers as input and produce real numbers as outputs.

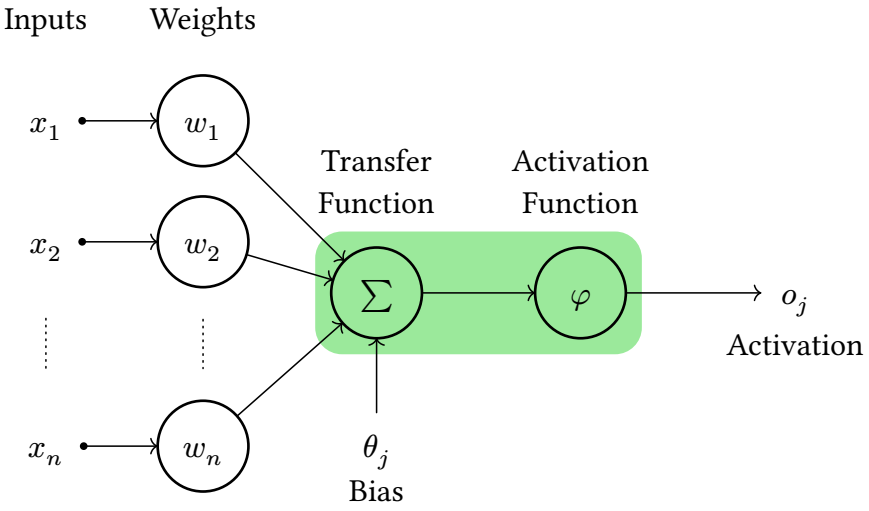


Figure 2.1: An artificial neuron (green) with inputs $x_1, ..., x_4$, a transfer function $\sum$ and an activation function φ, producting an activation o_j if $\varphi > \theta_j$.
Inspired by (Yacim and Boshoff 2018).

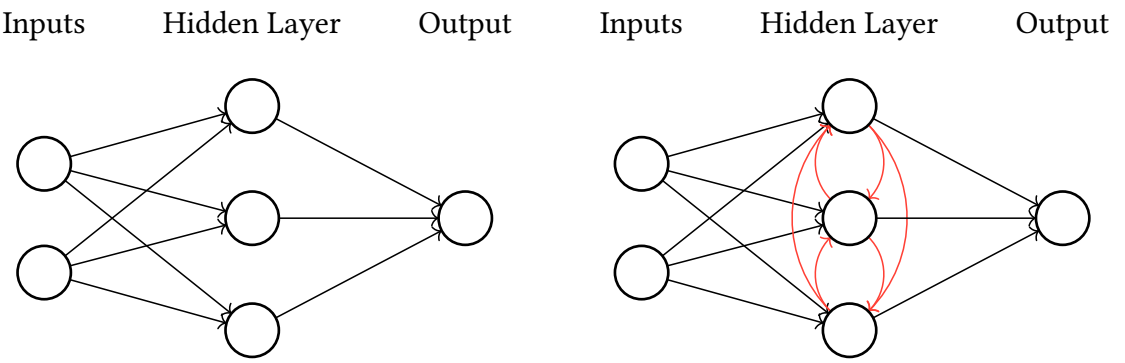
Figure 2.1 illustrates the typical structure of an artificial neuron: Their inputs are multiplied by their weights, the results are passed to the transfer function. Commonly, the transfer function sums the weighted inputs and applies a bias term. The function's output is evaluated in the activation function, a often non-linear function that produces the output o_j (Goodfellow, Bengio, and Courville 2016; Yacim and Boshoff 2018).

When speaking of a NN's parameters, we usually mean each neuron's weights and biases. It's these parameters that determine how well the network can approximate a specific function. While these parameters can be handcrafted, they are almost always initialized randomly and iteratively adjusted by the training process. This is done by calculating the deviation from the desired results using a loss function. The deviation is then used to go back through the network and adjust the weights. This is called backpropagation. Essentially, "Neural networks are, in the simple case, trained in a supervised manner, where the network's parameters are adjusted to match the known, desired output for a given input" (Hatzel 2020).

Once the training process is complete, the network is used on previously unseen inputs, predicting the expected value. A network's predictive accuracy depends not only on the dataset's size and how well it represents the unseen inputs, but also on factors such as model complexity, feature quality, the training process and data preprocessing. This accuracy is assessed using performance evaluation metrics, which are discussed in more detail in Section 2.3.

2.2.1 Traditional Architectures

After covering the fundamentals of NNs, we will briefly discuss different architectures. The simplest neural network architecture is the feedforward neural network (FNN) architecture, named after the flow of information in the network going from one layer to the next without loops (Goodfellow, Bengio, and Courville 2016). Since there are no



(a) A feedforward neural network (FNN). Information flows strictly from left to right.

(b) A recurrent neural network (RNN). It has connections within the same layer, highlighted in red.

Figure 2.2: Exemplary architecture of a FNN (a) and a RNN (b)

loops in the data flow, each input value is processed independently, making it unfeasible to capture sequential relationships. When neurons can activate themselves or other neurons within their layer to reflect on the previous state, we speak of recurrent neural network (RNN). Figure 2.2 illustrates how the two architectures differ from each other.

FNNs work with fixed-length input values, meaning variable-length inputs must be adjusted to fit. This can be done by either padding shorter inputs with placeholder values or trimming longer inputs to the required length. In contrast, RNNs can operate on variable length input, as for each timestep t, they operate on the input i_t and last timestep's output o^{t-1} (Hatzel 2020). They maintain a hidden state, that serves as the memory to store information about previous inputs.

While RNNs are able to capture sequential relationships better than feedforward networks, long-term relations are still difficult to capture due to the problem of vanishing / exploding gradients (Goodfellow, Bengio, and Courville 2016). In RNNs, we use a variation of backpropagation in the training process, backpropagation through time (BPTT), to unfold a network in time and compute the gradients (Werbos 1988). Since the gradient of a single weight can be calculated many times due to the recurrent nature of the network, the gradients of earlier weights will be exponentially smaller (or larger) than the gradients of later weights. While vanishing gradients lead to only small weight adjustments, prolonging the training process, exploding gradients make the training process unstable (Goodfellow, Bengio, and Courville 2016).

To capture longer sequential relationships and mitigate the vanishing / exploding gradient problem, Vaswani et al. (2017) introduce the Transformer architecture, marking a turning point in the field of machine learning.

2.2.2 Transformer Architecture

The Transformer architecture, introduced by Vaswani et al. (2017), is a non-recurrent approach to sequence modeling that differs from RNNs, which rely on sequential processing where each step depends on the previous one. By eliminating recurrence,

Transformers can utilize parallelization, leading to performance improvements over recurrent models. This architecture has achieved state-of-the-art results in machine translation tasks with the BERT model family (Devlin et al. 2019) and has had a large impact on machine learning and language processing.

Transformers generally follow an encoder-decoder structure, where the encoder generates a hidden state from the input, and the decoder uses this hidden state to produce the output. The model processes input through multiple layers, transforming it into a hidden representation before another set of layers generates the final output. Both input and output consist of a fixed number of tokens, known as the context size. A key feature of this architecture is attention, which allows the model to focus on relevant elements within the input, rather than processing the entire sequence uniformly.

2.2.3 Word Embeddings

While using real numbers as input for a neural net might be straight forward in applications where the data is already numerical, there are areas where the initial input first requires conversion into the numerical space, such as in language processing. One way of representing a word as a real number is by using one-hot encoding (Goodfellow, Bengio, and Courville 2016). One-hot encodings are vectors the size of the vocabulary, where each index in the vector corresponds to a fixed word in the vocabulary. To encode a word w, all entries in the vector are set to 0 except the value at index i_w is set to a non-zero value (typically 1).

One-hot encodings as word representations are easy to compute but have downsides: The vectors are sparse, with most of the values being 0, and therefore memory-inefficient. They are also unable to capture the order if words in a given sentence. To overcome these issues, more sophisticated embedding techniques were developed, such as Word2Vec (Mikolov et al. 2013). Word2Vec produces dense vector representations of words by training a next-word-prediction tasks over large amounts of data. The produced vectors have a lower dimensionality than one-hot encodings and capture meaning and relationship, as related words are close to each other in the vector space by measures such as cosine similarity. The vectors can be used for arithmetic operations, such as the well-known $king - man \approx queen - women$, as seen in Figure 2.3 (Sutor et al. 2019).

Transformers often use subword tokenization techniques, such as byte-pair encoding (BPE) (Gage 1994) and WordPiece (Schuster and Nakajima 2012). Here, the input is not split into words but rather smaller, reusable subwords in an effort to reduce vocabulary size. Additionally, this largely avoids the out-of-vocabulary (OOV) error, as at least parts of new, unseen words will be recognized by the tokenizer. Word-token-mappings are created during a separate training process of the tokenizer.

2.3 Evaluation Metrics

In this section, we will discuss different evaluation metrics and their effectiveness for evaluating the performance classification systems.

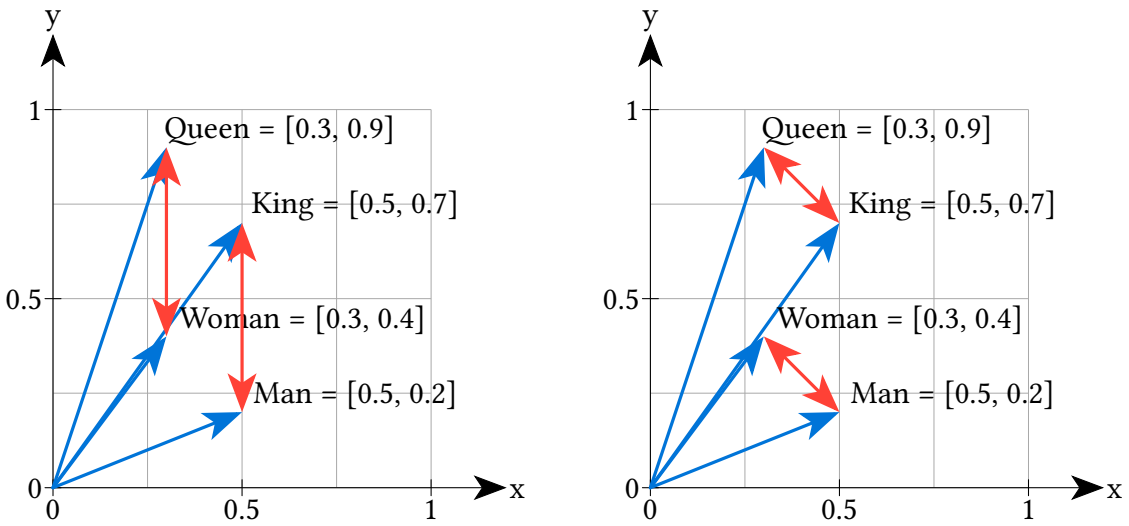


Figure 2.3: Cosine similarity of $king - man + women \approx queen$ word embeddings in 2D. It must follow that $king - man \approx queen - women$, as shown in red arrows. Similar direction of red arrows indicate similar relational meaning. Idealized illustration. (Sutor et al. 2019)

2.3.1 Accuracy, Recall, Precision, and F_β Score

When performing a binary classification task, we assign elements one of two groups. In the case of ad detection, we classify text samples into `AD` and `NO_AD` classes. By design, classifying a sample in a binary classification task has four possible outcomes, as shown in Table 2.1: true positive (TP), false positive (FP), true negative (TN) and false negative (FN).

Predicted / **Actual**	**Positive**	**Negative**
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

Table 2.1: The four possible outcomes of a binary classification decision.

They are multiple performance metrics for (binary) classification tasks, each measuring different aspects of the classifier's performance.

Accuracy is used to measure the number of correct predictions: $\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$ (Sammut and Webb 2011; Hatzel 2020). While accuracy works well for balanced datasets, where each class has roughly the same number of elements, it is less meaningful in imbalanced datasets. If the positive class appears a lot less than the negative class, the classifier can always predict the negative and achieve a high accuracy score. To better evaluate classifiers on imbalanced datasets, we use precision and recall.

Precision ($\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$) measures how many of the predicted positive classes were actually positive, thus indicating how many elements that are retrieved by the classifier

are indeed relevant (Sammut and Webb 2011). In the example above, where a classifier would only predict negatives to achieve a high accuracy score, its precision would be 0 as 0 elements were retrieved.

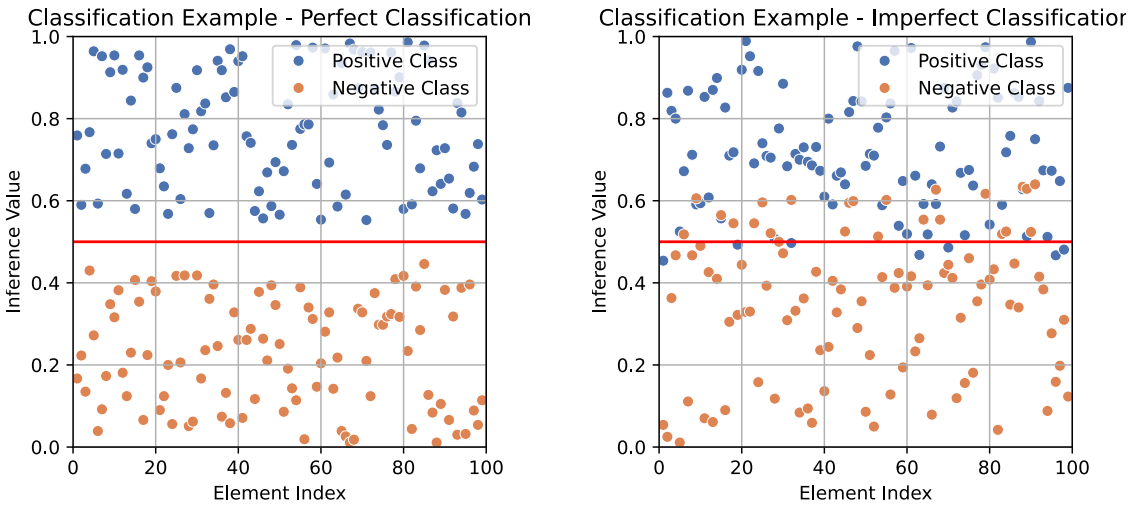
Recall, also *true positive rate*, measures how many of the positive elements a classifier correctly predicts: $\frac{\text{TP}}{\text{TP} + \text{FN}}$. In other words, it measures how many of the positive elements the classifier can find. We also know the false positive rate (FPR) ($\frac{\text{FP}}{\text{FP} + \text{TN}}$), which measures incorrectly predicted negative elements in relation to the total number of negative elements.

Precision and recall are often combined in the $\boldsymbol{F_\beta}$ **Score** to obtain a single variable for easier comparability: $(1 + \beta^2) * \frac{\text{precision} * \text{recall}}{(\beta^2 * \text{precision}) + \text{recall}}$ where β is a positive real number. β is chosen such as recall is considered β times as important as precision. When precision and recall are of equal importance, β is set to 1. We know this as the F_1 score.

2.3.2 ROC Curve

Output layers of NN based classifiers often consist of a single neuron. When using the classifier for inference, the neurons real-numbered output is mapped to either class (0,1) using a threshold, e.g. 0.5. In an ideal scenario, the classifier produces outputs for one class that are easily separable from outputs of the other class. Most of the time, classifiers do not achieve perfect scores. In this case, the threshold's value is a trade-off between precision and recall. Figure 2.4 visualizes the two scenarios.

In Figure 2.4b, the two classes cannot be predicted perfectly, regardless of the chosen threshold. If we increase the threshold to > 0.65, we increase the precision, since more



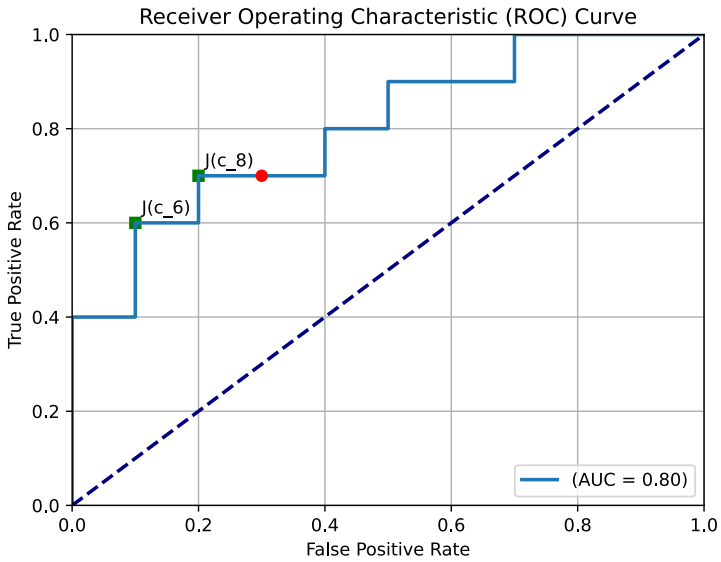
(a) A classifier perfectly predicting all samples. Chosen threshold (red line) splits classes correctly.

(b) Classifier predicting overlapping values for both classes. Threshold unable to perfectly separate the two classes.

Figure 2.4: Two classifiers predicting the class of elements. The true class is denoted by color. Elements with values above the threshold (red line) are interpreted to be predicted positive, all elements below as negative.

Class	Score
1	0.98
1	0.93
1	0.87
1	0.84
0	0.79
1	0.73
1	0.67
0	0.62
1	0.57
0	0.54
0	0.48
1	0.43
0	0.37
1	0.34
0	0.28
0	0.24
1	0.18
0	0.12
0	0.09
0	0.03

(a) Classifier decision results. Red line denotes exemplary threshold of 0.5.



(b) receiver operating characteristics (ROC) curve to classifier decisions in a). Best performing points on curved indicated by green squares, calculated by maximizing the Youden index over all thresholds of the curve.

Threshold	Actual \ Predicted	Positive	Negative
$c_1 = 0.5$	**Positive**	TP: 7	FN: 3
	Negative	FP: 3	TN: 7
$c_1^* = 0.57$	**Positive**	TP: 7	FN: 3
	Negative	FP: 2	TN: 8

(c) Confusion Matrix for classifier decisions.

Figure 2.5: Classifier result (a)) with corresponding ROC curve, indicated Youden indices (b)) and confusion matrix (c)). Inspired by Sammut and Webb (2011)

of our predicted positive values are actual positive, but decrease the recall, since we miss more of the actual positive elements. If we lower the threshold, say to < 0.4, we capture all our actual positive elements, thus increasing the recall, but also decreasing the precision, as more of our predicted positive elements are actually negative.

This trade-off between precision and recall over a range of threshold values can be visualized using a ROC curve (Figure 2.5). In Table 2.5a, the true class and a classifier's prediction is shown. The dashed red line represents a threshold of 0.5. To the right in Figure 2.5, we see the resulting ROC curve for all thresholds from 0 to 1. Each threshold corresponds to a specific true positive rate and true negative rate. The exemplary threshold of 0.5 is marked as the red dot.

The performance of the model expressed as $\mathrm{Tpr}(c) - \mathrm{Fpr}(c)$ is also referred as the Youden index $J(c)$ (Drumond et al. 2024). The best performing threshold c^* is the point on the curve that maximizes

$$\begin{aligned} c^* &= \arg\max_c \mathrm{Tpr}(c) - \mathrm{Fpr}(c) \\ &= \arg\max_c J(c). \end{aligned} \tag{2.1}$$

For the given data in Table 2.5a, two thresholds yield the same performance (marked as green squares), which are $c_1^* = 0.57$ and $c_2^* = 0.67$

Why these thresholds perform better than $c = 0.5$ is also apparent if we compare the confusion matrices for c_1 and c_1^* in Table 2.5c. In summary, we use the ROC curve and it's Youden indices to evaluate a classifier's performance over multiple decision thresholds.

2.4 Imbalanced datasets

As we will see in Chapter 6, advertisements in podcasts make up less than 10% of an average episode's duration. This is similar to many other fields of research, where the target class occurs less often than the other class, such as computer vision (Huang et al. 2016), text classification (Estabrooks, Jo, and Japkowicz 2004), fraud detection (Estabrooks, Jo, and Japkowicz 2004) or medical science (He and Garcia 2009). In most cases, we speak of an imbalance if one class occurs three times more than the other, but in extreme cases, imbalance can be as high as 1:5000 (He and Garcia 2009).

The reason for the imbalance can be both intrinsic and extrinsic (He and Garcia 2009). Intrinsic imbalances are a direct result of the nature of the data space, e.g. when training a classifier to detect a rare disease, the disease is naturally less occurring than non-disease images (given a medical diagnosis dataset of normal distribution). Extrinsic imbalance on the other hand can stem from various factors, such as the time the dataset was created.

Training on imbalanced data with canonical ML algorithms that assume an equal sample distribution over all classes leads to biases towards the majority classes (He and Garcia 2009). This is because the algorithms are designed to minimize the overall error rate, rather than paying extra attention to the positive, minority class (Chen, Liaw, and Breiman 2004). This poses a problem as the minority class is often the class of higher importance.

The scientific domain of working with datasets where class frequency is imbalanced is referred to as learning from imbalanced data and has been researched for well over two decades (Japkowicz and Stephen 2002). The consensus is that while class imbalance might not be the only factor in learning difficulties, it certainly is an important point to address when building robust ML systems (He and Garcia 2009). In general, the means to overcome the class imbalance problem are either resampling the dataset or assigning custom class weights to the training algorithm's loss function (He and Garcia 2009). We will briefly look at their workings as well as when to use which strategy.

2.4.1 Resampling

Resampling describes the strategic of oversampling the minority class or the undersampling of the majority class in order to adjust the class distribution. Oversampling the minority class can either mean to feed the algorithm samples of the minority class more than once or creating additional, artificial minority class samples. Duplicating samples has the risk of overfitting, while creating artificial samples might not always be possible (e.g. if the data is hard to produce, like medical imagery (He and Garcia 2009)).

Undersampling usually entails selecting fewer majority class samples in training. This strategy can become problematic when the dataset is small and further limiting the sample size via undersampling can lead to a performance decrease due to small sample size. Estabrooks, Jo, and Japkowicz (2004) find that neither oversampling nor undersampling is always the best strategy to use, as it is data and problem dependent.

In summary, resampling strategies can work well if the given dataset is large enough to support the redistribution.

2.4.2 Weighted Loss Function

Weighted loss functions, as used in cost-sensitive learning, describes assigning custom weights to the loss function of a ML algorithm, which results in higher misclassification costs of the minority class (Huang et al. 2016). This can be a viable approach if the dataset is rather small and reducing the sample size via undersampling would lead to too little data. Chen, Liaw, and Breiman (2004) found that when comparing random forest algorithms with either resampling or cost-sensitive learning, both adaption performed better than algorithms without any imbalance strategy. But, they note that paying closer attention to minority class members via cost-sensitive learning makes the algorithm more vulnerable to noise in the minority class samples. They also observe the algorithm using resampling to be more computationally efficient as it operates on a smaller subset of the dataset.

2.5 Advertisements

The American Marketing Association (AMA) defines advertising as "[…] the placement of announcements and messages in time or space by business firms, nonprofit organizations, government agencies, and individuals who seek to inform and/or persuade members of a particular target market or audience regarding their products, services, organizations or ideas" (American Marketing Association 2025). A single announcement or message is commonly called advertisement. Nowadays, advertisements are a significant component of social and entertainment media, playing an important role in marking strategies, often as the primary revenue source, e.g. in radio broadcasts (Álvarez et al. 2024).

According to Spotify, the top monetization strategy for podcast is advertisements, followed by subscriptions and merchandising (Spotify 2023b). Since podcasts are usually distributed to listeners for free, advertising is an effective mean of monetization, as proven in other free online media, such as videos on YouTube. Generally,

advertisements in podcasts can be classified into either traditional advertisements or sponsorships (Ritter and Cho 2009; Bulakh et al. 2023; Brinson and and 2023; Moe 2023).

In this section, we will provide the first answer to

RQ1: How can advertisements in podcasts be characterized and what differentiates advertisements in audio podcasts from advertisements in video podcasts?

While the findings in this section are taken from literature, we will revisit the question in Chapter 6 with practical insights when exploring the generated dataset.

2.5.1 Advertisements in Audio Podcasts

Advertisements in podcasts follow the aforementioned classification of traditional advertisements and sponsorship. Traditional advertisements are prerecorded advertisements as found on TV and radio broadcasts. They are often 15 seconds or 30 seconds long and produced by the advertiser. We will call traditional advertisements *inserted ads*, as they are inserted into the content.

Sponsorships are affiliations of the podcast's host with a brand. Sponsorship ads are read by the host and or their team members. We will call these *self-voiced ads*. They are produced at the podcast producer's end, allowing for a 'look and feel' that is close to the regular content.

Ritter and Cho (2009) find that while inserted ads often contain direct selling messages, while self-voiced ads indirectly deliver sponsor information. This might be one of the reasons why podcast listeners prefer self-voiced ads over inserted ads, with 62% of participants stating they prefer self-voiced over inserted advertisements in a study in the U.S. in 2023 (Cumulus Media, Signal Hill Insights 2024). This is in contrast to 15% preferring inserted ads and 23% or participants have no opinion.

According to a study from 2022, self-voiced advertisements account for more than half (55%) of the revenue of podcasts in the U.S. (IAB (U.S.) 2022), while some sources indicate that self-voiced advertisements are also more expensive for the advertiser to place in a podcast than running inserted ads (SiriusXM Media 2024). This might be because studies have found that people generally have more goodwill towards sponsorships compared to traditional advertisements, increasing their effectiveness (Meenaghan 2001).

While inserted ads were often statically 'baked into' the audio a few years ago, these days they are usually distributed using dynamic ad insertion (DAI) systems. These systems dynamically insert advertisements into the content, usually when the podcast is requested via the RSS feed consumer (Acast 2023). This is used to always embed new and relevant advertisements into podcast episodes, regardless of when they were produced. DAI systems can leverage consumer data to ensure close ad targeting and relevance. When comparing audio podcasts to video podcasts as found on YouTube, the use of DAI is not possible as videos are fixed in content from the moment they are uploaded.

2.5.2 Advertisements in Video Podcasts

The first major difference between advertisements in audio podcasts and those in video podcast is that in the case of video podcast, the distribution platform might have their own, separate ad logic employed. YouTube, as the leading platform for video podcasts (Mayer and Riismandel 2023), offers three different ad formats, varying in style, length and whether the user can interact (skip) with the ad. They can appear before, during or after the video (YouTube Help 2025a).

YouTube highlights their ads when running them by differently coloring the progress bar and showing information about the advertiser where the regular video's discription would be. When crawling YouTube videos for corpus generation done by automated tools like YT-DLP (yt-dlp 2025), these types of ads are not fetched, as they are not part of the video (although YouTube has experimented with embedding ads more natively into videos in an effort to combat ad blockers (Anu Adegbola 2024)). Furthermore, as they are strictly not part of the podcast episode, uploading a podcast to a different distributor would not include them. We therefore exclude these kinds of advertisements from our research and focus on advertisements directly embedded into the episode. This means that all ads shown as part of the YouTube ad program are ignored.

When analyzing advertisements in video podcasts embedded by the creator, we observe notable differences from traditional audio podcasts. A significant distinction is the absence of inserted ads in video podcasts on YouTube. This lack of inserted advertisements can be directly attributed to YouTube's policy, which explicitly prohibits such ads in its terms of service (YouTube Help 2025b) (Listing 2.1).

> `Does this mean I can burn video ads (pre-rolls, mid rolls, and post rolls) into my videos?`
>
> No. YouTube's ads policies don't let you burn or embed advertiser-created and supplied video ads or other commercial breaks into your content.

Listing 2.1: Excerp from YouTube's terms of service prohibiting embedded advertisements. (YouTube Help 2025b)

Consequently, as YouTube has become the leading platform for video podcast consumption in recent years (Mayer and Riismandel 2023), this restriction contributes to the overall landscape of advertising in video podcasts.

Another difference between advertisements in audio and video podcasts is the lack of DAI systems for YouTube advertisements. Here, advertisements are still 'baked in', as YouTube does not support changing a video once it has been uploaded. This leads to static ad content in video podcast.

3
Related Work

The task of identifying advertisements in podcasts is closely related to tasks in other fields of research, such as speech recognition, content detection and content segmentation. In a broad sense, ad detection in podcasts can be generalized to the task of content detection in long form audio. In this section, we will discuss research on advertisement detection in long form audio, its related tasks and how this thesis fits into the existing landscape.

3.1 Content Detection in Long-Form Audio

Content detection in long-form audio refers to the segmentation and classification of different content types in an audio stream. Content types include speech, music, advertisement and silence. The task has been studied in a variety of domains, such as speaker diarization, music information retrieval and speech-to-text transcriptions.

In the beginning, researchers utilized handcrafted, low-level audio features for analysis. Lu, Zhang, and Jiang (2002) propose a content detection algorithm that first discriminates between speech and non-speech content before further categorizing non-speech content into music, environmental sounds and silence. Distinction between speech and non-speech is largely done via analyzing handcrafted signal processing features, such as the "high zero-crossing rate ratio (HZCRR), low short-time energy ratio (LSTER) [...] [and] spectrum flux (SF) [...]" (Lu, Zhang, and Jiang 2002) and using classifiers such as K-nearest-neighbor (KNN) and linear spectral pairs-vector quantization (LSP-VQ). They achieve promising results with precision and recall metrics between 0.8 and 0.9.

Speech / non-speech (SNS) detection is also examined by Zibert, Vesnicer, and Mihelic (2007). They differentiate between SNS classification and segmentation, stating that classifying non-speech content is especially hard as it can be produced by various acoustic sources. They mark that most approaches process the audio signal by extracting low level acoustic audio features such as mel-frequency cepstral coeffi-

cients (MFCCs) (e.g. in Logan and others (2000)) to perform the detection tasks. They introduce a higher-level approach using phoneme-recognition features designed to be language agnostic, slightly approaching the problem from a NLP point of view. They conclude that they expect the most suitable representation of audio signals for the SNS segmentation tasks is a combination of acoustic and recognition-based features.

Zibert, Vesnicer, and Mihelic (2007) employed hidden Markov models (HMMs) for the evaluation of their approach, a statistical model architecture that is widely used in the field of content detection a few years ago. The domain of speech detection has seen heavy use of HMMs, e.g. in Rabiner (1989) Juang and and (1991) and Gales and Young (2008). In short, HMMs are statistical models operating on sequential data, such as speech, that are assumed to follow the properties of a Markov chain with hidden states, which states that the probability of each event depends only on the state at the previous event. In speech recognition, the underlying phonemic and linguistic units can be modelled as the hidden states and the temporal dependencies of speech can be effectively modelled with HMMs due to the Markov chain.

With the introduction of the Transformer architecture by Vaswani et al. (2017), content detection especially in text but also in audio format has seen a shift from the proven probabilistic models like the HMM and neural methods like long short-term memorys (LSTMs) to Transformer-based approaches.

Zaman et al. (2025) give an overview how the Transformer architecture is used in the field of audio detection tasks. They elaborate how the Transformer architecture, originally designed for the field of NLP, has been successfully used in audio detection tasks, due to its ability to capture "[...] long-range dependencies and complex patterns in audio signals" (Zaman et al. 2025), made possible by the self-attention mechanism. Additionally, the end-to-end nature of the Transformer eliminates the need for extensive preprocessing of the signal. While having set new benchmarks in complex audio detection tasks, the architecture does come with its downsides, notably higher computational cost and complexity, significant training resource requirement and expertise in deep learning.

In summary, we see that approaches in content detection have evolved over the last two decades from using handcrafted, low level audio features to processing using probabilistic models like the HMM and more advanced neural network architectures like LSTMs, to finally adopting the more performant but also more resource intensive Transformer architecture. In this thesis, we will follow the latest development and leverage the Transformer in our model architecture. This choice allows us to achieve high-performance results while preprocessing minimizing complexity and simplifying the development process, thanks to the end-to-end capabilities of Transformer models.

3.2 Advertisement Detection in Digital Media

The detection of advertisement in digital media has been a topic of interest since the start of the millennium. The existing work is mostly focused on detection in the context of radio and television broadcasts. We will examine the strategies employed in the following paragraphs.

In the early 2000s, researchers at Dublin City University were building a digital video system to efficiently record, analyze, browse and view digitally captured television programms (Marlow et al. 2001). Upon users requesting the option of automatically deleting advertisement breaks, they begin to developed solutions for the problem of advertisement detection. As many popular TV stations used a series of black, silent image frames to separate advertisements from regular content and separate single advertisements within larger ad blocks, they propose a system relying on the detection of said frames for content segmentation. While achieving an average F_1 Score of 0.97 for ads of stations using black frames, the proposed solution is not resilient against stations not using black frames for segmentation. Here, they measure the average shot length, hypothesizing that commercials have a higher shot cut rate to maximize the visual impact of ads. This proves to be less reliable, averaging a F_1 of 70.6. Marlow et al. (2001) try to exploit specific knowledge about how ads are embedded into the broadcast, which proves unreliable when the expected criteria are not met. It is unclear how this method transfers to the domain of podcast.

In 2006, Covell, Baluja, and Fink (2006) utilize acoustic and visual fingerprinting in order to detect and replace advertisements in television broadcast streams. They motivate the replacement of ads to insert advertisements that are more tightly targeted to the viewer. Their two stage approach first builds acoustic fingerprints for possible match detection with fingerprints of known advertisements, before a computationally expensive visual verification is performed. While they achieve high results, they rely on the fact that most advertisements in television are "re-purposed footage" (Covell, Baluja, and Fink 2006), meaning an advertisement is produced once and broadcasted multiple times. This is not necessarily the case for podcasts, as self-voiced advertisements are cheap to produce, resulting in some advertisements to be recorded on an episode basis.

Cardinal, Gupta, and Boulianne (2010) take a similar approach to Covell, Baluja, and Fink (2006), developing a system to monitor the shown advertisements in television broadcast. They motivate their work by stating that it's hard for advertisers to know if their advertisements are broadcasted as requested and paid for. Computing a fingerprint of both audio and video of each frame, they slide the known advertisement over the content and count the number of matched frames. This way, the task procedure for "searching specific ads in an audio stream is very similar to the copy detection task" (Cardinal, Gupta, and Boulianne 2010). But, as previously noted, we cannot assume advertisements being repeatedly used in the domain of podcasts, given the prevalence of self-voiced advertisement segments.

Ramires, Cocharro, and Davies (2018) propose a different approach that is closer to Marlow et al. (2001) by listening for short silences as segment boundary markers between regular programming and advertising. They state that they can differentiate between proper segment boundary silences and silences in regular content by examining the frames before and after the silence, as a boundary silence is expected to be "short in duration, have a low minimum value, and be surrounded by regions of much higher energy" (Ramires, Cocharro, and Davies 2018). They feed the signal data to a multiple linear regression model, outperforming a freely available audiovisual approach.

All the work discussed so far have approached the problem from a signal processing perspective. In 2024, Álvarez et al. (2024) introduce a novel approach by utilizing a customized RoBERTa (Liu et al. 2019) model to solve the task via text classification. Their solution removes the need for prior knowledge of the broadcast content, having performed extensive supervised training of audio broadcast material. The training data is sourced from high-audience Spanish radio stations and self-annotated using open source annotation software. The annotation process is overseen by a computational linguist in an effort to secure precise and accurate annotations.

The annotated audio is segmented by either exact or non-exact segmentation. The former segments the audio into chunks of n seconds before transcribing, the latter does the transcription first and segments the text afterward. Transcription is done using OpenAI's Whisper (Radford et al. 2023). Before training on the transcriptions, the text is split into window lengths of $x, x \in \{10, 20, 40\}$ seconds. They find that their model performs best when training with a window length of 10 seconds and evaluating with a window length of 40 seconds, with transcriptions coming from Whisper's medium-sized model. With a F_1 Score of 0.87, they outperform approaches of prior studies.

Concluding, we see that advertisement detection in audio media in the past has been done using primarily signal processing, until the introduction of the Transformer architecture, which enabled text classification approaches to outperform previous attempts. The source data was mostly from television, with Álvarez et al. (2024) using radio broadcasts as input. In our study, we will combine the audio and text modalities in an effort to capitalize on a wider feature space. Additionally, we will apply this approach to in-domain and out-of-domain podcast data.

3.3 Advertisement Detection In Podcasts

Now that we have looked at advertisement detection in the domain of television and radio, we will turn to podcasts, as other studies have previously suggested approaches specific to this domain.

In 2010, Nguyen, Tian, and Xue (2010) propose an efficient method for unknown (but repeated) advertisement discovery in podcast using a two-step process: Candidate segmentation and sampled search. They split the podcast into potential candidates and non-candidates by classifying one-second windows of audio data based on signal features (e.g. energy-entropy block (EEB) and high-zero-crossing-rate ratio (HZCRR)) using a fuzzy neural network (FCMAC-BYY). Afterward, they fingerprint the candi-

dates and compare them with other candidates' fingerprints. This work, similar to others in the domain of television broadcast, uses signal processing, as it was the de facto approach at the time.

Reddy et al. (2021a) use the Spotify podcast dataset (Clifton et al. 2020) (see Section 3.4) to introduce classifiers to detect extraneous content in podcast, such as "sponsor advertisements, promotions of other podcasts, or mentions of the speakers' websites and products" (Reddy et al. 2021a). Extraneous content is a source of noise for NLP tasks, which motivates the removal. They first fine-tune pretrained BERT (Devlin et al. 2019) models on podcasts descriptions and transcriptions (one model each) and then train the obtained models for classification on self-annotated podcast data. On the sentence level, they measure higher classification performance for the description corpus compared with the transcription corpus. They speculate that incorporating audio features might improve performance as extraneous may appear as pre-recorded audio or with difference speaking pitch and cadence.

In his bachelor's thesis Dahlin (2024) explores the idea of an adblocking system for podcasts, similar to adblockers for web browsers. His idea is to exploit the fact that many podcasts use DAI to inject targeted ads based on the user's location, among other features. By fetching an episode by two different IP addresses using a proxy, the two files can be compared and checked for dissimilarities. He states that while his solution suffers from several limitations, it's lightweight in nature and capable of removing 100% of ads in a podcast, given the podcasts contains only geolocated, dynamically inserted advertisements.

With our third research question, we analyze how models trained on video podcast perform when evaluated on audio podcasts:

RQ3: Can training data from video podcasts be used to detect advertisement in audio podcasts?

We find that previous studies have not tried to classify advertisements in podcasts with classifiers that were trained on OOD data. They used either the Spotify dataset (Clifton et al. 2020) or created their own. Since data annotation is a labor-intensive task, using possibly pre-annotated OOD data could reduce the required effort.

Using transcriptions of YouTube videos to train a neural network identifying advertisement segments is not a new idea. GitHub user Andrew Lee has built a neural network to predict timecodes of ad segments in YouTube videos (Lee 2023). The project uses a bi-direction LSTM network trained on automatically generated video subtitles with user-generated advertisement timecodes from SponsorBlock (Ajay Ramachandran 2024) (see Section 5.1.2). The 'YouTube Transcript API' python package is used to source the subtitles. The package uses an "undocumented part of the YouTube API, which is called by the YouTube web-client"(Depoix 2024).

It is unclear how accurate the transcripts are in terms of word error rate or how they are generated, other than likely by Google Cloud's automatic speech recognition (ASR) tools. Performance data of NeuralBlock's LSTM model is not available. The

author hints at using different transcription techniques as a means of improving the system, as YouTube creators can disable the generation of closed captions altogether and the quality of the available captions is unknown (Lee 2023). The project had its last meaningful changes in September 2020.

3.4 Podcasts As A Research Resource

Podcasts have been used as a research basis by numerous studies, albeit not for ad detection. In 2020, researchers at Spotify released a dataset of 100,000 audio files and their respective transcription of English podcast episodes (Clifton et al. 2020). They demonstrate the complexity of the domain with two exemplary tasks: Passage search and summarization. Later released research on podcasts often use this dataset as a basis (Reddy et al. 2021a; Vaiani et al. 2022). It has since been pulled and is no longer available (Spotify 2023a). It seems also unlikely that it contained advertisement time codes.

Reddy et al. (2021b) use the Spotify dataset to investigate how linguistic style and textual attributes in podcasts relate to listener engagement in a quantitative analysis. They extract multiple linguistic features such as episode length, vocabulary diversity, proportion of ads, swearing etc. using term frequency–inverse document frequency (TF-IDF), latent Dirichlet allocation (LDA) and other means. These features are compared with Spotify streaming numbers of first time listeners. They find that engaging podcasts tend to contain positive sentiments, are longer, have fast speaking creators and are less likely to contain swearing. The correlation between ads and engagement is mixed, as large amounts of ads in transcripts are associated with lower engagement, but ads in podcast transcriptions don't seem to have a large impact.

Zhu et al. (2023) propose the first spoken language identification (SLI) system using speaker embeddings from podcast data. Previous SLI systems are primarily designed for short audio clips, showing low performance when applied to long-form audio. Generating VGGVOX (Chung, Nagrani, and Zisserman 2018) speaker embeddings during unsupervised speaker diarization, they train a multi-class FNN, achieving F_1 scores of 0.91 and 0.81 for long-form and short-form audio respectively.

Lastly, Ghazimatin et al. (2024) present a fine-tuned encoder-decoder Transformer model to segment conversational data as found in podcasts. Podcast pose a difficult source for segmentation tasks, as episodes tend to be less structured than written text and often contain spontaneous side tracks before returning to a larger topic (Figure 3.1). By providing the podcasts metadata and previously generated chapter titles, their model generates chapter transitions and titles simultaneously, augmenting input text samples with global context. Doing this, they are able to outperform existing long-dependency text segmentation approaches (e.g. CATS as in (Glavăs and Somasundaran 2020)).

Figure 3.1: Chapters (purple circles) for (a) an episode about training tips vs. (b) a structured Wikipedia article about training. The episode chapters have short tangential discussions (gray circles), shared context (Peter's experience), and a consistent title style. In contrast, Wikipedia chapters focus on the main topic with short titles that lack global context. (Ghazimatin et al. 2024)

3.5 Existing Podcast Ad Detection Systems

In this section, we will briefly look at an existing podcast ad detection system: Adblock Podcast (Engle-Eshleman 2025). Created by Engle-Eshleman as a side project, Adblock Podcast is a paid podcast player that automatically detects and skips advertisement for the user. User payments are shared with creators whose ads were skipped as means of compensation. During brief e-mail correspondence, the creator explained that he had looked into several techniques to detect advertisements, prioritizing speed and efficiency. Ultimately, he employs "simple heuristics layered on top of each other" (Engle-Eshleman (2025) E-mail to Felix Wolf, 1. April). He states that the most annoying ads are often the easiest to detect, with his system struggling with self-voiced ads. He is planning to possibly employ a large language model (LLM) to improve the service.

3.6 Advertisement Type Categorization

Bulakh et al. (2023) have examined podcast advertising and propose a classification of podcast advertising. Advertisements in podcasts can be classified according to multiple overarching categories, such as the method of providing advertising information, the location of the advertisement in the podcast as well as the method of insertion of the ads. An excerpt of their classification is shown in Table 3.1. While they come up with seven categories for types of advertisements in podcasts, only the first two are of relevance for this thesis, as they describe the types of short, content-interrupting segments we try to identify:

1. Pre-recorded segments produced by the advertiser (*inserted*)
2. Host-read advertising announcements (*self-voiced*)

These two types of advertisements are also present in other research (Brinson and and 2023; Moe 2023; Ritter and Cho 2009). In addition to the type of advertisements,

Classification criteria	Types of advertising	Characteristics
According to the method of providing advertising information	re-recorded commercial audio clip	the video is recorded by the host which starts automatically before the podcast. Usually such videos are professionally recorded, they are only inserted into the podcast in the right place
	spoken by the host	an announcement delivered by the podcast host
	an invited expert	the interlocutor of the podcast who is a representative of the company
	native advertising	advertising embedded in the podcast (mentioning a product or service in the context of the podcast topic)
	sponsorship/partnership presentation	mentioning the company as a sponsor/partner of the program. The listeners' loyalty to the brand and its recognition is increased
	branded podcast/ season	podcasts are created under a separate brand, by the brands themselves usually
	advertising modules / thematic episode	these are mini-podcasts on various topics within the main releases, the leitmotif of which is a certain brand. The brand can participate in the recording of the module and/or choose a topic that suits both the brand and the podcast
According to the method of ads in podcast location	pre-roll ad	it appears at the beginning of the podcast and lasts 10-15 seconds
	mid-roll ad	it appears in the middle of the podcast and lasts 30-90 seconds
	post-roll ad	it appears at the end of the podcast and lasts 15-30 seconds
According to the method of insertion ads into podcasts	dynamic insertion	advertising is recorded separately
	embedded advertising	recorded in the main audio file of the episode

Table 3.1: Classification of podcast advertising. Except from Bulakh et al. (2023).

they categorize the placement in pre-, mid- and post-roll ads as well as the method of insertion that can be either dynamic or statically embedded.

In this thesis, we will follow the distinction of inserted and self-voiced advertisements, as they appear also in other research can be found in our dataset (see Chapter 6).

4
Problem Statement

In this chapter, we will summarize the problem of advertisement detection in podcasts and define a formal problem statement.

4.1 Problem Context

As the related studies in the previous chapter have shown, working with long form audio in general and with podcast in particular for classification tasks is difficult due to the medium's heterogeneous characteristics. To recap, podcasts as a media format come in many different flavors, from formal news formats to true-crime stories, comedy and many more. Each genre differs in speaking style, number of participants and structure.

Regarding the advertisements included in the data, there is neither a guaranteed position within the podcast nor is the length of an ad segment fixed. Segments can be inserted or self-voiced, with inserted advertisements often being distributed via DAI. The use of DAI systems makes the creation of datasets and corpora non-deterministic, as it is highly dependent on factors that are relevant to the advertiser, such as the date, the geographical location and other user specific data. Bulakh et al. (2023) show that advertisements can be categorized using different criteria, like how they convey the advertising information, the ad's position in the episode as well as how it is inserted (Table 3.1). They find that inserted segments differ from self-voiced segments in style and production. This diversity in style complicates classification tasks, as there is no clear blueprint for what content and advertisements can look like.

Before one can classify advertisements in podcasts, the input data first has to be segmented into chunks that are large enough to capture needed context, but small enough as to not exceed an algorithm's input size limit. Some studies have proposed approaches to chapterize podcast data (Ghazimatin et al. 2024), which is a difficult task on its own, due to podcast's non-linear structure. If segmentation were to be done by time, segments could be cut mid-sentence or mid-chapter, resulting in decreased coherence. Text-level segmentation on the other hand requires the decision of how

many utterances / words / sentences should be considered as a single input, with the aforementioned tradeoff between large context and small input size in mind.

Lastly, the approaches looked at in Chapter 3 have most often used signal data as model inputs (e.g. Ramires, Cocharro, and Davies (2018)), with only the most recent studies leveraging the Transformer model architecture to efficiently compute and use word embeddings (e.g. Álvarez et al. (2024)). Combining both modalities in an efficient model architecture is desired.

4.2 Research Questions

We have defined the following research questions in Chapter 1:

RQ1: How can advertisements in podcasts be characterized and what differentiates advertisements in audio podcasts from advertisements in video podcasts?

RQ2: How does model performance vary when using audio, text, or multimodal training data?

RQ3: Can training data from video podcasts be used to detect advertisement in audio podcasts?

We will use our research questions to structurally investigate the problem. Answering RQ1 will give us insights into the structure of podcasts, RQ2 will help us understand how much information audio and text features in podcast hold. With RQ3, we investigate how well learned features can be transferred to video podcasts and the space of YouTube videos.

4.3 Formal Problem Statement

This thesis investigates the problem of automatically identifying advertisement segments in audio podcasts using machine learning. The central challenge lies in designing a model that can distinguish between ad and non-ad content across diverse podcast genres, formats, and insertion styles.

5 Methodology

After outlining the problem in the previous chapter, this chapter describes the practical approach taken to tackle the issue. We will describe the employed technologies and considerations for the dataset creation, the model selection, training as well as the evaluation.

5.1 Data Collection

As previously mentioned, a dataset of labeled ad segments in podcasts transcriptions and their audio source is not readily available. The Spotify Podcast Dataset (Clifton et al. 2020) did provide transcriptions of 100,000 episodes of English podcasts, but it is unclear whether they contained advertisements and the dataset has since been pulled from the internet.

To address this issue, we will create our own dataset. Since we want to explore how OOD data from video podcasts can be used to detect advertisements in audio podcasts (RQ3), the dataset will be composed audio and video podcasts. We will leverage the SponsorBlock database (Ajay Ramachandran 2024) to automate the annotation of the video podcasts. The audio podcast dataset will be annotated manually, as there is no database to use, thus having less episodes. From now on, we will speak of either the audio podcast or the video podcast dataset.

5.1.1 Audio Podcasts

The audio podcast dataset represents podcasts as they are listened to by most users, with 60% of Spotify users streaming a new show choosing an audio podcast (Spotify for Creators 2024). The first step in creating the dataset is selecting shows and episodes to download. To eliminate biases in the selection of podcasts, we select first 100 shows in Apple Music's top podcasts charts in the United States as of the 10th October 2024. The list was sourced from chartable.com (Chartable.com 2024), a service that has since been phased out. For reference, the shows as well as their order can be found in

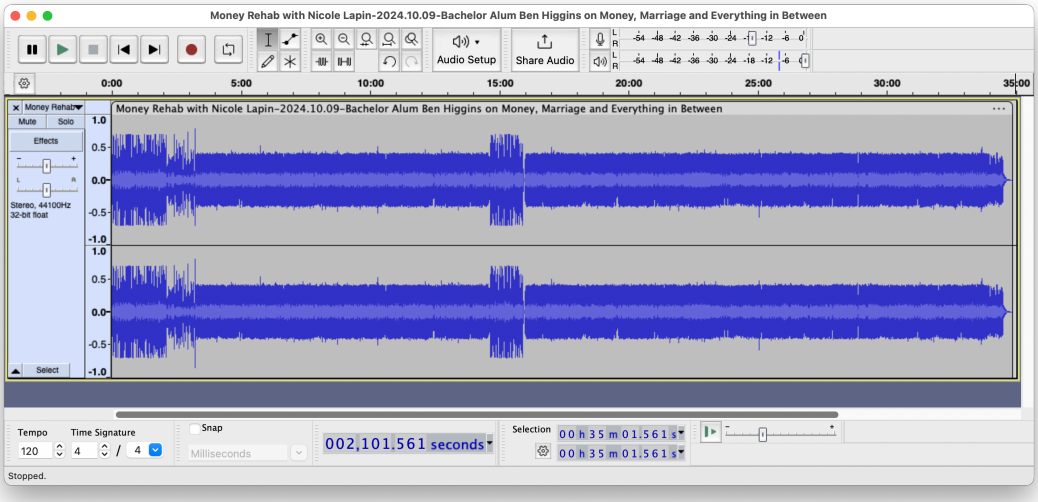


Figure 5.1: A podcast episode loaded into Audacity. From the waveform, we instantly see parts of the audio that are different from others. Placement and length hints that these parts might be advertisements.

Appendix A. For each of the 100 shows, all available, but no more than five episodes were downloaded, sorting by release date and selecting the latest episodes.

The next step is the annotation. Each episode is manually annotated. The process here is to open the episode in Audacity[1] and examining the waveform. In many cases, possible ad breaks are visually distinguishable from regular content, as shown in Figure 5.1. Skipping through the episode, we ensure that no advertisements are missed and also very that possible candidates identified by visual inspection are indeed advertisements. Start and end timecodes of advertisements are retrieved by hand and saved in a JSON file, as well as the type of advertisement (inserted / self-voiced). Audio loading, data entry and saving is streamlined by a custom python script.

The final step is the transcription of the audio files. This is described in Section 5.2. In total, 160 of the 480 downloaded audio podcast episodes were annotated, due to time limitations.

5.1.2 Video Podcasts

Using in-domain audio podcasts for training and evaluation is the canonical way when ultimately seeking inference on audio podcasts. The appeal for creating a dataset from OOD data to train ML models is in the possibility to skip the tedious data annotation step, allowing us to create larger datasets with less effort in less time. Essentially, we are looking for a database with existing advertisement timecodes for podcast-like material. On YouTube, this database is part of the SponsorBlock project.

SponsorBlock is an "[...] open-source crowdsourced browser extension and open API for skipping sponsor segments in YouTube videos" (Ajay Ramachandran 2024).

[1] https://www.audacityteam.org/

Heuristic Attribute	**Condition**	**Description**
Segment Category	= 'sponsor'	Must be labeled as a sponsor segment
Video Duration	1hr $\leq$ duration $\leq$ 3hr	Podcast-like video length
Segment Length	$>$ 10 seconds	Filters out very short segments

Table 5.1: Heuristics used to extract podcast-like video from SponsorBlock's database of YouTube-videos.

Users submit timecodes and appropriate category tags (e.g. 'sponsor', 'unpair / self-promotion', 'exclusive access' etc.) for their currently watched YouTube video using buttons embedded into the YouTube player. If a user watches a video that happens to already have timecodes submitted by another user, the 'sponsor' segment is automatically skipped by the video player. SponsorBlock can be understood as an adblocker for in-video advertisements. The project is available on most browsers and as of April 2025, the SponsorBlock database has data for 10,040,141 videos, although only 2,793,476 have timecodes for 'sponsor' type segments (data taken from downloaded database).

Since the SponsorBlock database does not only contain the advertisement timestamps for each video, but also metadata like the video's duration in seconds, the type of segment and the length of the segment, we can create a simple heuristic to retrieve videos that are most likely podcast-like (Table 5.1). We use these heuristics to query the SponsorBlock database for podcast-like videos. Tools to download YouTube videos programmatically are readily available, we choose YT-DLP (yt-dlp 2025). The downloaded videos are stored as audio files in 16 kHz mono to reduce storage footprint. We download 1,000 videos using our outlined heuristics. Chapter 6 will explore the created dataset.

5.2 Transcription

The transcription process is the same for the downloaded audio and video podcasts. We use OpenAI's Whisper (Radford et al. 2023) to create transcriptions of our audio data, as it is freely available, performant and low in word error rate. Whisper is a Transformer sequence-to-sequence model trained on a large dataset of diverse audio, enabling it to perform multilingual speech recognition, translation and speech identification (Radford et al. 2023).

When creating the final dataset for training, our goal is to generate samples labeled either `AD` or `NO_AD`, with no overlaps (mixed labels). While we can split the audio data along the timecodes to ensure no overlaps, we lack any information about the corresponding text content. We need to align the advertisement timecodes with the audio transcriptions.

For this, we use the whisper-timestamped fork of Whisper (linto-ai 2023), which enhances the original model by adding word-level timestamps and confidence scores to the transcriptions. This enables us to accurately split the transcription based on the advertisement timecodes.

When using Whisper and whisper-timestamped, we can choose between six model sizes, from the 39 million parameter tiny model to the 1.5 billion parameter large model. An increase in model parameter increases accuracy but also required VRAM and processing speed. For comparison, the large model is 10 times slower than the tiny model (OpenAI 2022). Facing time and computing constraints, we choose the base model with 74 million parameters, hoping to achieve a balance between transcription accuracy and resource and time requirements. We will examine obtained transcription quality in the data exploration chapter (Section 6.11).

5.3 Segmentation

As discussed in the problem statement in Chapter 4, segmenting the raw audio stream is a difficult task. We are looking for a tradeoff between small samples being easy to digest by models, but holding little information regarding its context and large samples holding a lot of (not necessarily monothematic) context, but being potentially too large for specific model input limitations.

On the text level, if we assume that the topics of advertisements differ from regular content, we ideally split the audio along changes in topic. But to do so, we would have to employ some kind of chapterization technique as presented by Ghazimatin et al. (2024) (Glavăs and Somasundaran 2020) and Feldstein Jacobs (2022). Traditionally, text segmentation is done using the LDA topic model as presented by Hearst (1997) and later Riedl and Biemann (2012), but it is likely that these methods perform worse given the less linear structure of podcasts compared to traditional texts (see Figure 3.1). Using more advanced methods like the mentioned exceeds the scope of this thesis.

On the audio level, we could use cues and characteristics found at content borders to segment along, as showcased by e.g. Marlow et al. (2001) in television broadcasts using silent frames. However, it is unclear whether these characteristics are similarly present and consistent.

Ultimately, we've decided to employ sentence level segmentation, splitting the transcription along sentence borders. Using the timestamped transcription obtained through whisper-timestamped, we can split the audio accordingly. Since single sentences might hold too little contextual information, we can concatenate sentences to increase the context.

5.4 Feature Selection

While traditional approaches in content classification (as discussed on Chapter 3) would use hand-crafted features extracting specific data characteristics (e.g. MFCCs) as their classifier's input, the emergence of the Transformer architecture (Vaswani et al. 2017) and pre-trained Transformer models (like the generative pre-trained Transformer (GPT) model family) enables us to use pre-trained model embeddings as features for downstream tasks, such as classification. As these pre-trained models have been trained on vast inputs, their embeddings can be expected to be feature-rich. Additionally, where traditional approaches necessitate a pipeline of multiple models each solving a specific

step (e.g. topic modelling and segmentation), modern models are build in an end-to-end architecture, unifying the tasks of multiple models in a single models, decreasing setup and training complexity.

In this thesis, we will use multimodal input embeddings computed from audio and text input produced by the LanguageBind Transformer model as feature representations for our classification layers. The model is further described in the next section.

5.5 LanguageBind

LanguageBind as presented by Zhu et al. (2024) is a multimodal Transformer model, designed to align input representations between language and audio, video, image, depth and thermal modalities into a unified embedding space. The unified embedding space enables zero-shot and few-shot transfer across tasks and modalities.

Architecturally, LanguageBind uses a multi-encoder approach, in which each modality is processed by a dedicated encoder. Before the input is being processed by the encoder, transformation is done. For text, this is done by a BPE tokenizer. For other modalities, they employ patch embedding, transforming the inputs into Transformer-digestable patched vectors. Specifically for audio, the transformation step entails downsampling to 16 kHz.

LanguageBind uses a contrastive learning objective, feeding pairs of text and another modality into the model. Pairs of semantically aligned inputs are encouraged to have similar embeddings, while non-matching pairs are pushed apart. To enable this, they present a novel five-modality dataset (VIDAL-10M), holding three million pairs of video-language, infrared-language and depth-language and one million pairs of audio-language. The resulting learned input embeddings serve as semantic representations of the raw input.

Since we pursue a multimodal approach for our classification, in which we want to capture information in text and audio, using LanguageBind to source aligned embeddings is a logical choice.

5.6 Model Architecture

As outlined in the previous sections, we segment the raw input on the sentence level and feed the aligned samples of both audio and text into the LanguageBind Transformer model to obtain multimodal feature embeddings, which we use in a classification layer.

Figure 5.2 illustrates the model architecture. Each episode consists of an audio file as well as corresponding timestamped transcription. The segmentation step uses the timestamped transcription to split the text into sentences. As LanguageBind expects to receive whole audio files as inputs, we could split the audio into sentence-long chunks and use those as our input, since we want to the model to pay attention only to the part of the episode's audio data that aligns with the current sentence. I/O operations such as loading audio data into memory is computationally expensive, encouraging a different approach than doing thousands of I/O operations for tiny audio pieces per episode.

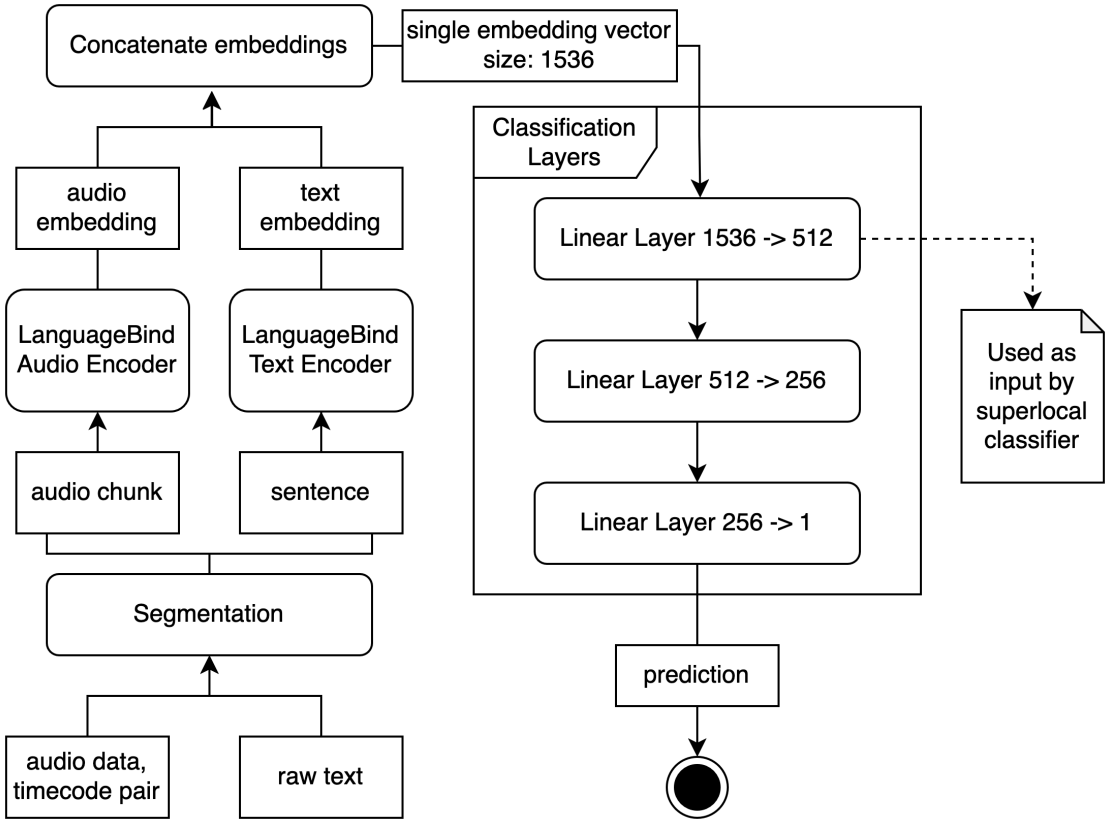


Figure 5.2: Model architecture of *local* classifier. LanguageBind audio and text embeddings are fed through a FNN classifier.

Instead, we do not split the audio into sentence chunks, but rather create timecode pairs of sentence start and end times. During training, we load the entire episode into memory once and use the timecode pairs to splice the audio data for our desired data chunk. Making LanguageBind accept preloaded audio data and timecode pairs as input required an interface change.

Feeding raw text and audio data and timecode pairs to LanguageBind, we receive the feature embeddings from text and audio encoders. Before handing them to the downstream classifying layers, we concatenate the two vectors of size 768 to a single vector of size 1536. The concatenated vector is passed to the classification layers, consisting of two hidden layers and the output layer. The architecture can also be configured to only work on single-modality embeddings from text or audio. In this case, the concatenation step is skipped, and the first linear layer takes in input vector of size 768 and produces an output vector of size 512.

The above presented architecture processes multiple samples at once, in batches, but the samples have no information about surrounding samples. Therefore, we can feed the samples randomly into the model, increasing robustness. We call this architecture *local*, as the context is limited to the local sample. Figure 5.3 presents another architecture employed in the thesis, which uses multiple intermediate sample representations from a *local* classifier as input to introduce contexts. We will call this

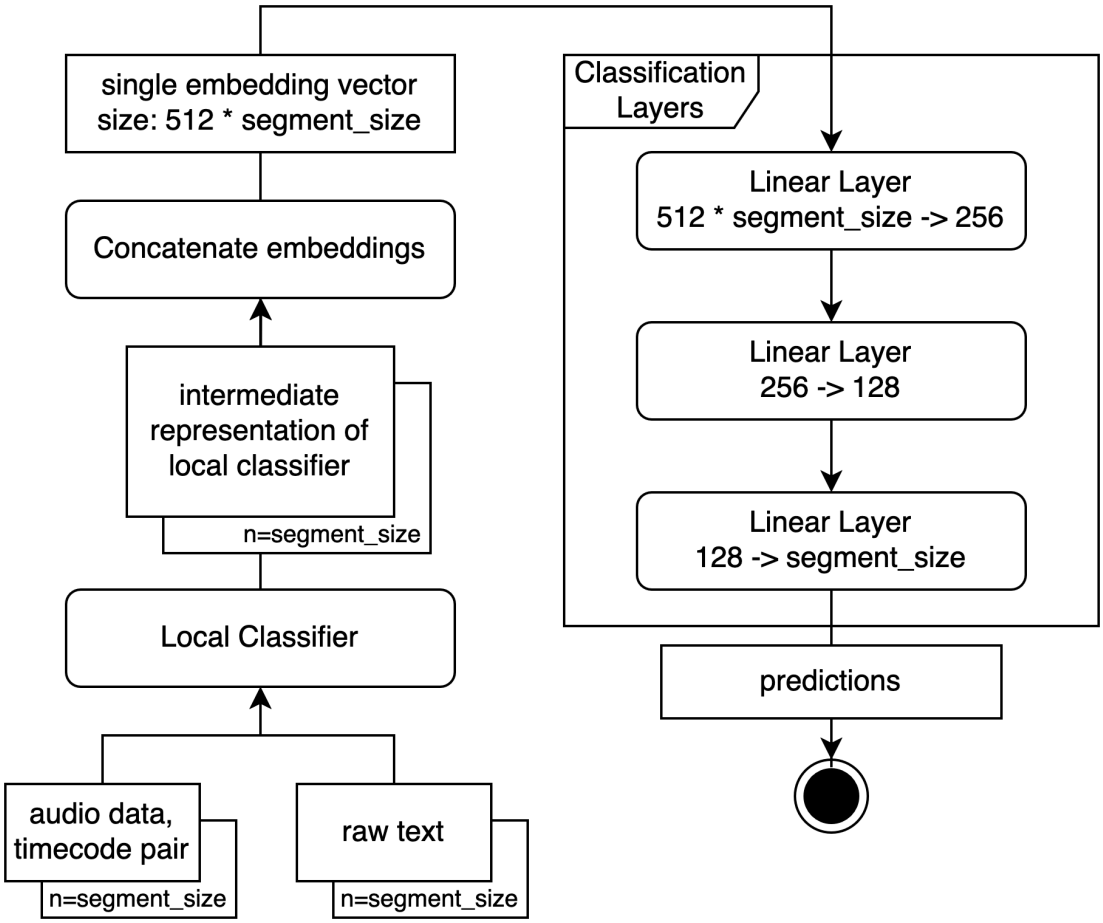


Figure 5.3: Model architecture of *superlocal* classifier. Intermediate representations of *local* classifier are concatenated to create context.

architecture *superlocal*, as it's still limited, but exceeds local context. The intermediate representations produced by the local classifier are concatenated into a single vector and subsequently reduced in dimensionality in the classification layers. The output layer produces a single vector with a real number for each input sample.

5.7 Training Process

We choose binary cross-entropy as our error metrics, as it is well suited for binary classification tasks. We will experiment with undersampling of the majority class to overcome the imbalanced class problem. Here, we test different class splits, as well as training on the original distribution and see their impact. Using backpropagation, we update the weights on our classification layers, but leave the weights in the Language-Bind encoders as-is. In our classification layers, we use a dropout probability of 0.3 and the RELU activation function.

In the local classifier, where samples are not logically connected to each other, we input samples into the model out-of-order. The batch size is set to 32. In the superlocal classifier, samples are fed into the local classifier in order (otherwise, the context

would be meaningless), before being concatenated. Here, we experiment with different segment size values.

To retain comparability between different models when using random sample selection, we use the same random seed throughout all experiments. Additionally, we perform a train test split of 70%/30%, that also uses a fixed seed.

We will train local classifiers on audio and video podcasts separately, in an effort to compare the performance similarities and since we want to answer RQ3, in which we examine how OOD data can be used to train models running inference on in-domain data.

Superlocal classifiers use a local classifier model as for their input transformation. Here, we use the best performing model obtain in the process of local classifier training. Due to scope constraints, we will only train superlocal classifiers on in-domain audio podcast data.

To recap, Table 5.2 summarizes the experiments we will carry out. While the first two are designed to answer our remaining research questions, the third experiment is designed to overcome possible downsides of our chosen architecture, namely the missing context between samples. We will also investigate if different advertisement types have different classification performances. After the experiments are carried out, we will experiment with decision thresholds, look at ROC curves and evaluate the models in a real-world application scenario.

5.8 Performance Evaluation

We will evaluate models by computing their precision, recall, F_1 and $F_{0.5}$ scores. We motivate the evaluation of $F_{0.5}$ scores by stating that advertisement detection is a precision-oriented task. When an advertisement detection system is used to automatically remove advertisements from podcasts to improve user experience, removing content

Name	Description
Modality Information Gain	Train models on embeddings of text data, audio data and combined data. Evaluate performances. Answers RQ2.
Out-of-domain Training Data	Train models on in-domain and out-of-domain data, compare performance when evaluated on in-domain data. Answers RQ3.
Superlocal Context	Introduce superlocal context by concatenating embeddings of single sentences. Evaluate how performance is influenced.
Advertisement Type Classification Performance	Train classifier on only self-voiced or inserted ads, evaluate performance differences.

Table 5.2: Experiment overview.

due to false positives (low precision) is far worse than not removing advertisements due to false negatives (low recall). The $F_{0.5}$ scores values precision as twice as important as recall.

Additionally, we employ cross-evaluation, evaluating a model that is trained on audio podcasts both on audio podcasts and video podcasts and vice versa. This will help us answer our third research question:

RQ3: Can training data from video podcasts be used to detect advertisement in audio podcasts?

For selected models, we will examine how the choice of the decision threshold value in inference influences the model's performance. We will use the ROC curve and its true positive and false positive rate to illustrate a model's performance in a real-world scenario with an example. Lastly, we will examine faced errors and try to find hypotheses for their causes.

6 Data Exploration

In this chapter, a comprehensive analysis of the datasets will be provided. Using various plots, we will gain insights on how advertisements are used in podcasts. We will look at the position of ads in an episode, how the number and length of ads correlate with the episode's length as well as general information regarding the entries in the dataset. This will lead us to the answer of the RQ1 (see Section 6.13). Throughout the data exploration, data related to audio podcast is colored blue, while video podcast data is colored orange.

6.1 General Statistics

First, Table 6.1 holds general statistics about the dataset. As discussed in Chapter 5, the dataset consists of 160 annotated audio podcasts and 1000 annotated video podcasts. Since video podcasts were converted to 16 kHz mono, they have only a slightly larger storage footprint than audio podcasts, which we store in 44.1 kHz stereo. Measuring how much of an episode's duration is attributed to advertiser content, we see that audio podcasts have more than double the advertisement percentage compared to video podcasts (8.95% vs. 3.43%).

6.2 Timeline Distribution

Next, we will look at the date of upload of the entries in the dataset, which is shown in Figure 6.1. The x-axis holds the relevant months and years, the y-axis denotes

Source	Number of Items	Storage Size	Total Duration	Ad Duration	Ad Percentage
Audio Podcast	160	13.85 GB	171h	15h18m	8.95%
Video Podcast	1000	15.50 GB	1,481h	50h44m	3.43%
Total	1160	29.35 GB	1,652h	66h03m	4.00%

Table 6.1: General dataset statistics.

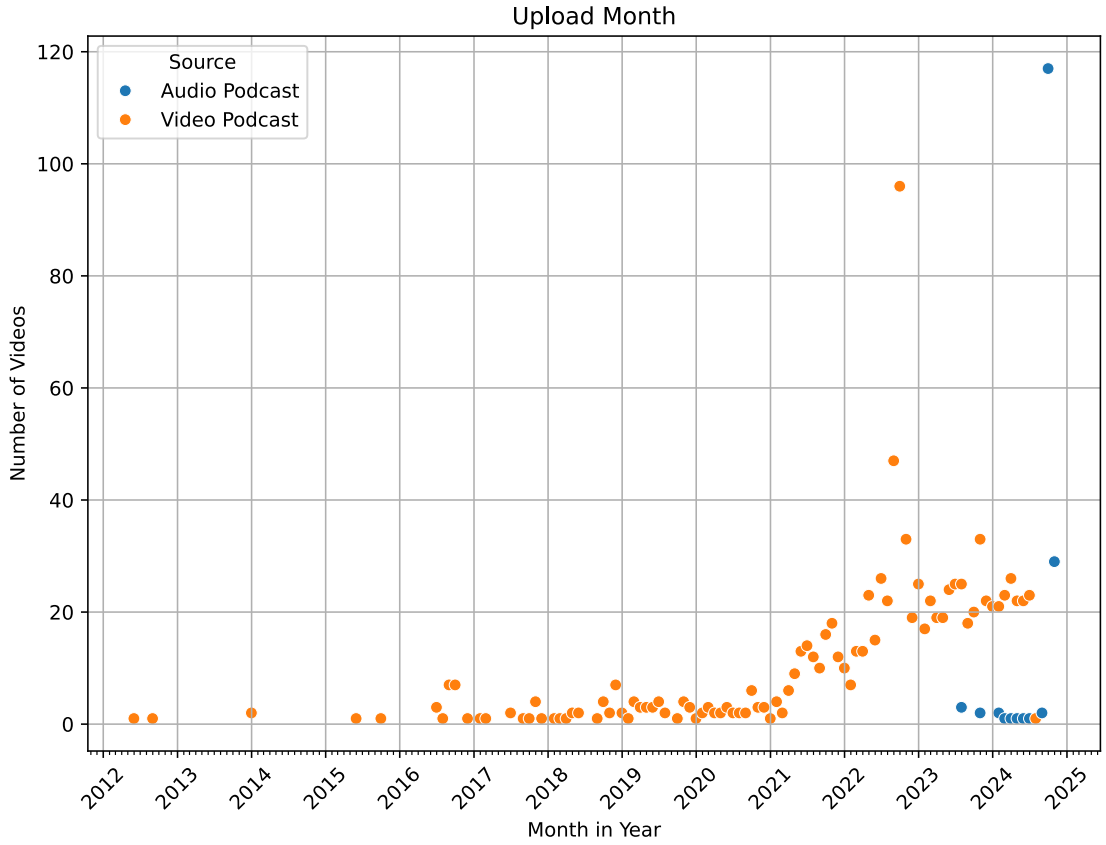


Figure 6.1: Timeline distribution of dataset entries.

the number of videos uploaded in a particular month. In our video podcast dataset, entries are distributed from 2012 until 2024, with the majority of data points from 2021 onwards.

Audio podcast data on the other hand is only available for the years 2023 and 2024. This is caused by the way the audio podcasts were sourced, since we've selected podcasts that were popular in mid of October 2024 and of those podcasts, the 5 most recent episodes were downloaded. That way, a podcast releasing once a week would have no episode older than September 2024. Some podcasts release new episodes less frequently than weekly, which is why we have some data points that are released in 2023. Also, it could be the case that a podcast is popular in late 2024 that stopped airing in late 2023.

Video podcasts were chosen randomly from all entries in the SponsorBlock (Ajay Ramachandran 2024) database. Here, we see a bias towards recent years. The hypothesis here is the project gained users over the years and people might watch and therefore annotate primarily new videos.

6.3 Episode Length

Looking at the episode's duration in the dataset (Figure 6.2), we see that on average, audio podcasts are shorter than video podcasts. The minimum length in video podcasts is 60 minutes, which is a slightly more than the median value of the audio podcasts. Here, it's important to note that the used heuristics in the video podcast dataset generation (see Section 5.1.2) required the videos to be at least 60 minutes in length. This heavily influences the statistic, as one can assume that most YouTube videos are shorter than one hour. Another heuristic limits our query to videos no longer than 3 hours, which is why no video is longer than that.

We see that in the case of video podcasts, most episodes have a length between 60 and 100 minutes, with the majority of those being between 60 and 70 minutes in length. Audio podcasts on the other hand seem to follow a normal distribution, with most of the episodes being between 30 and 70 minutes in length.

6.4 Genres

Next, we examine the episodes' genres. In all cases, the podcast's RSS file contained the `<itunes:category>` tag, holding the specified genre. Some podcast have multiple entries, in that case the first entry is used. Video podcasts sourced from YouTube have no RSS file, but we are able to process the video tags provided by the creator, as they are returned to us by YT-DLP (yt-dlp 2025). We therefore have genre data for both sources, but cannot directly compare them as their categorization differ. Figure 6.3 shows the found genres.

While the two sources share only a few common genres, such as comedy, education and sports, we can also look at what is included in one part of the dataset but absent

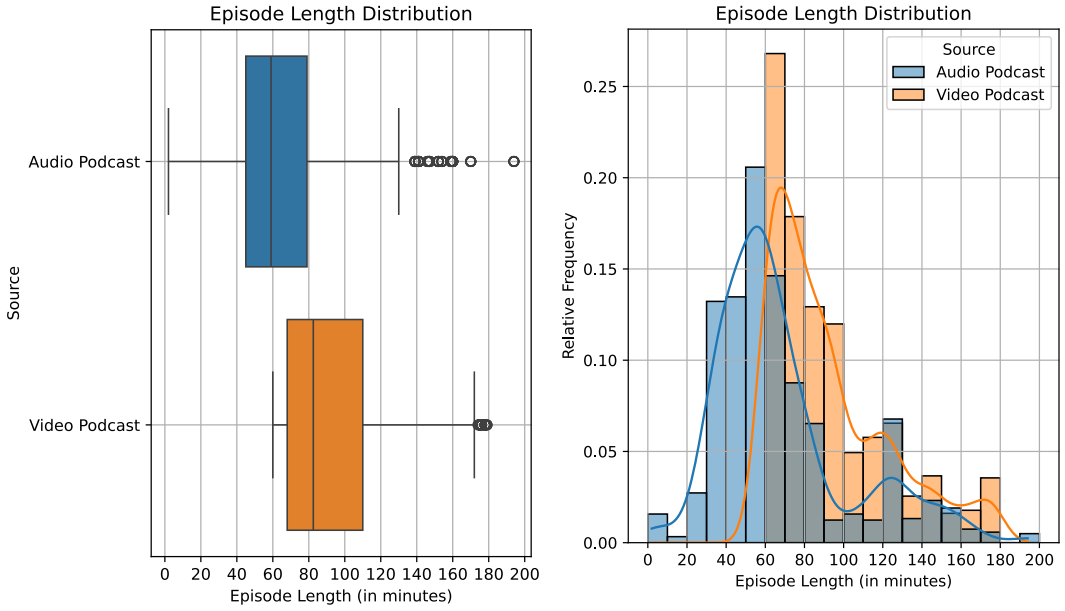
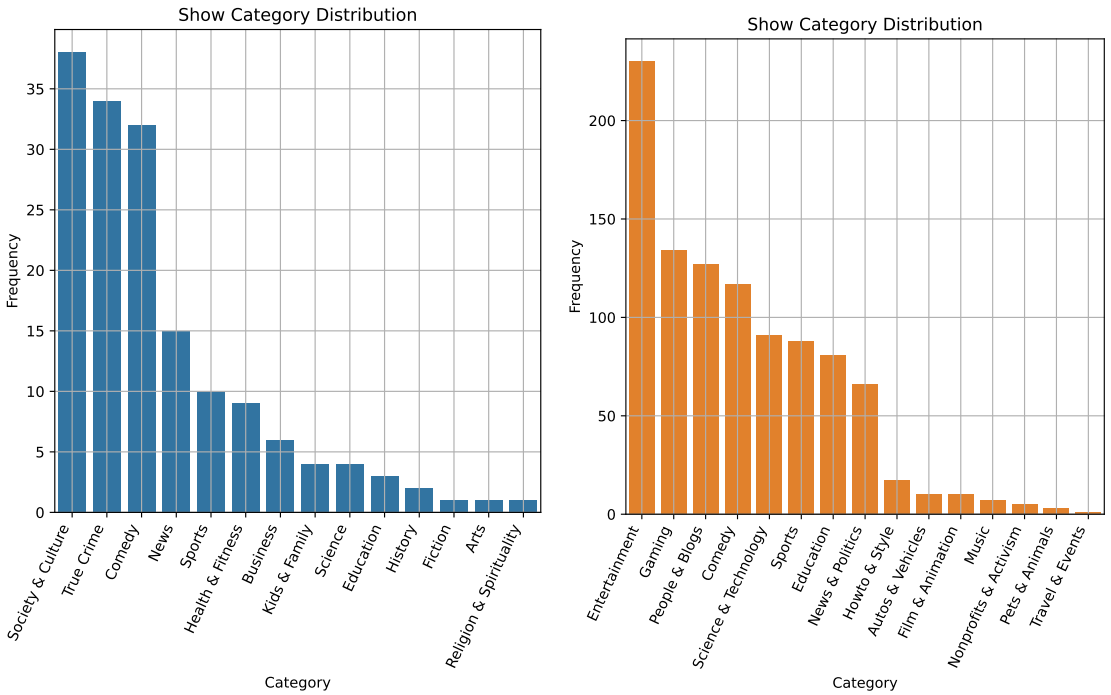


Figure 6.2: Episode lengths in minutes. On average, audio podcasts are shorter.



(a) Audio podcast genre frequencies. (b) Video podcast genre frequencies.

Figure 6.3: Genre frequencies of audio and video podcasts. As they come from different sources, they cannot be directly compared.

in the other. For example, the genre of true-crime has the second-highest frequency among audio podcasts and is not found in video podcasts. This either means that true-crime is not popular on YouTube or that true-crime videos are categorized differently, for example under entertainment. In video podcasts, entertainment is the most-frequent genre, but it's not listed for audio podcasts.

6.5 Number of Ads per Episode

Turning to advertisement-related observations, we first examine the number of advertisements per episode. Figure 6.4 shows that one to twelve individual advertisement can be found in a typical audio podcast, with seven segment being the most common.

In a study published by Westwood one, 600 participants where questioned for the acceptable number of advertisements in a podcast of a specified length (Westwood One 2021). For a 60-minute-long podcast, 3.8 advertisements were deemed to be appropriate. Compared to our findings, we conclude that today's audio podcasts now contain, on average, twice as much advertising as was deemed acceptable in 2021.

Video podcasts usually contain one to three segments, with about 75% of episodes containing one or two individual advertisements. While we can ensure that individual advertisements are annotated as such (as opposed to marking multiple ads in a row as a single annotation) for audio podcast, we cannot say the same for video podcasts, as the annotations are crowdsourced. Users contributing to the SponsorBlock project have little incentive to label multiple, consecutively shown advertisements as single

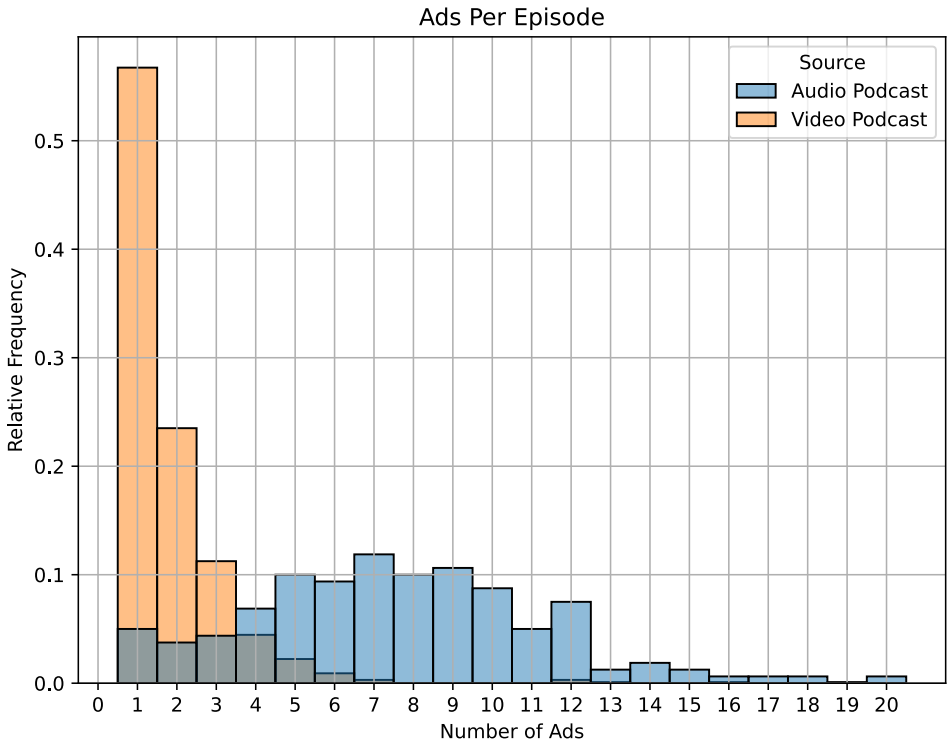


Figure 6.4: Number of ads per episode.

blocks. They are likely to annotate them as one segment, as the goal is to skip the complete block.

In an attempt to answer whether video podcasts contain no consecutive advertisements, we have sampled 10 random advertisements from our dataset. Out of those 10, 7 ad breaks did only contain a single advertisement and three samples contained 2 advertisements. From this experiment, we conclude that video podcasts contain on average 1.3 advertisements per annotation, rounded to 1.

If we now compare the number of advertisement blocks (opposed to individual advertisements) between audio and video podcasts, the two domains are more closely aligned, as shown in Figure 6.5. While the video podcast data did not change (we assume that every annotation is a complete block), audio podcasts now most frequently hold four advertisement blocks. About 80% of audio podcasts have three to five advertisement blocks. This is closer in line with the aforementioned study, stating that participants find 3.8 advertisements per 60-minute podcast appropriate.

6.6 Number of Consecutive Advertisements

Another way to look at whether advertisements come in blocks or individually is to examine the number of ads presented in a row. Figure 6.6 shows that while we assume video advertisements mostly come individually, most advertisements in audio podcasts

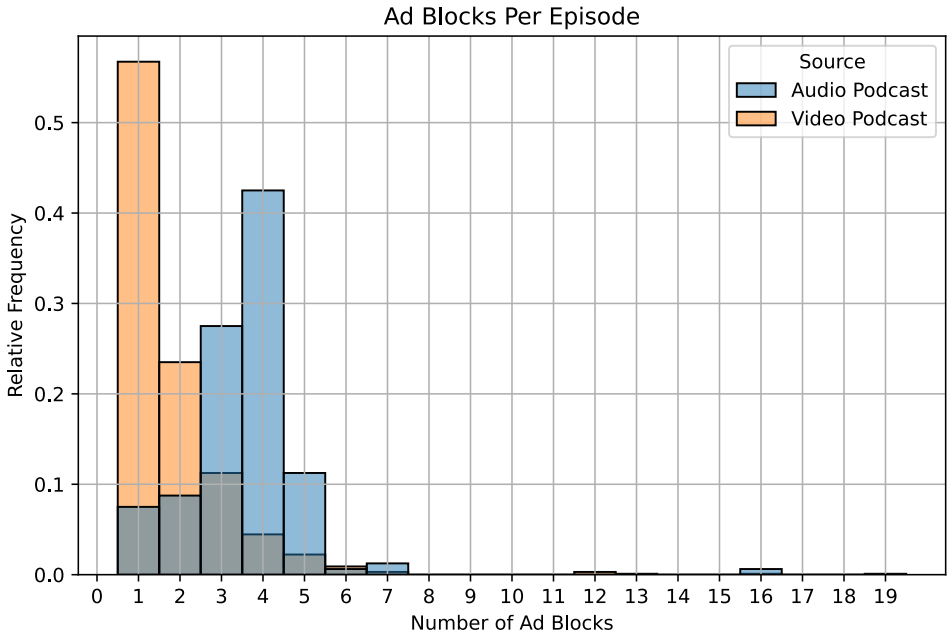


Figure 6.5: Number of ad blocks per episode.

come in blocks of at least two. In nearly 10% of the cases, four consecutive advertisements are played.

Between blocks of consecutive advertisements, regular content is played. Figure 6.7 illustrates that while both sources follow a similar trend, video podcasts have a ten-

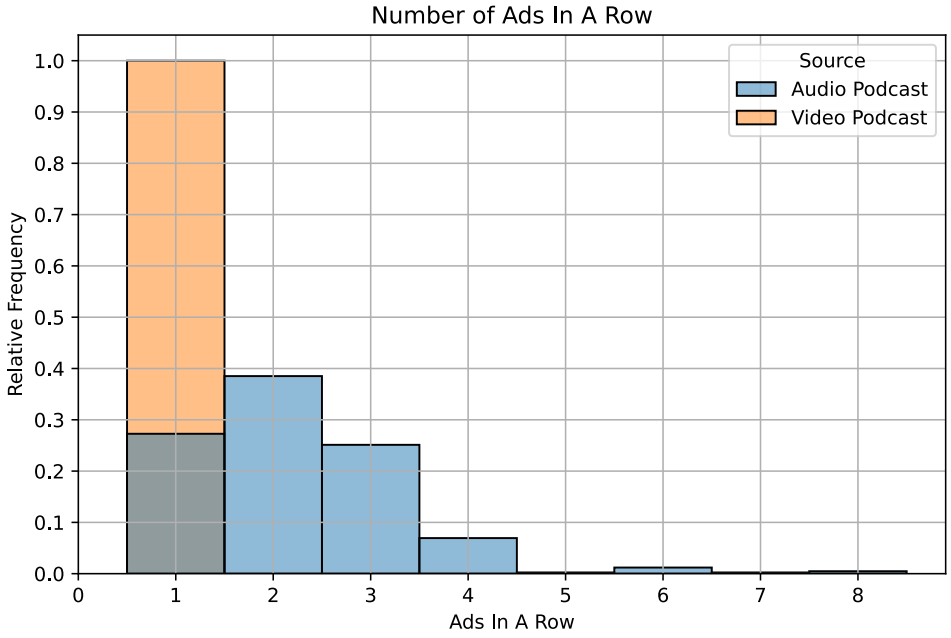


Figure 6.6: Number of ads in a row per episode.

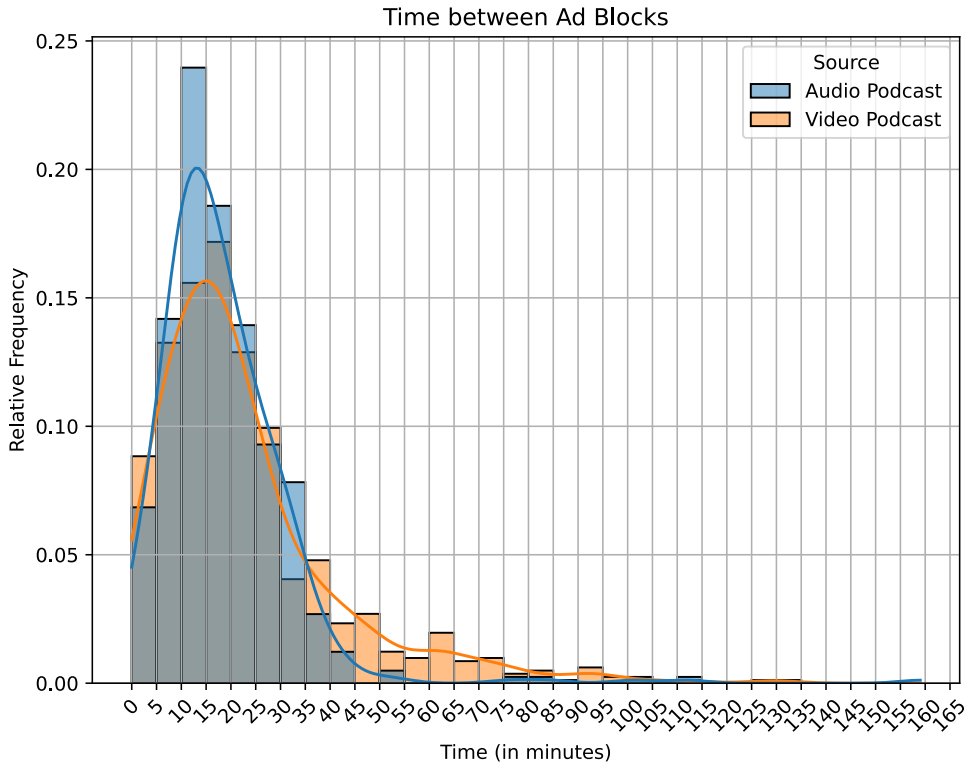


Figure 6.7: Time between ad blocks.

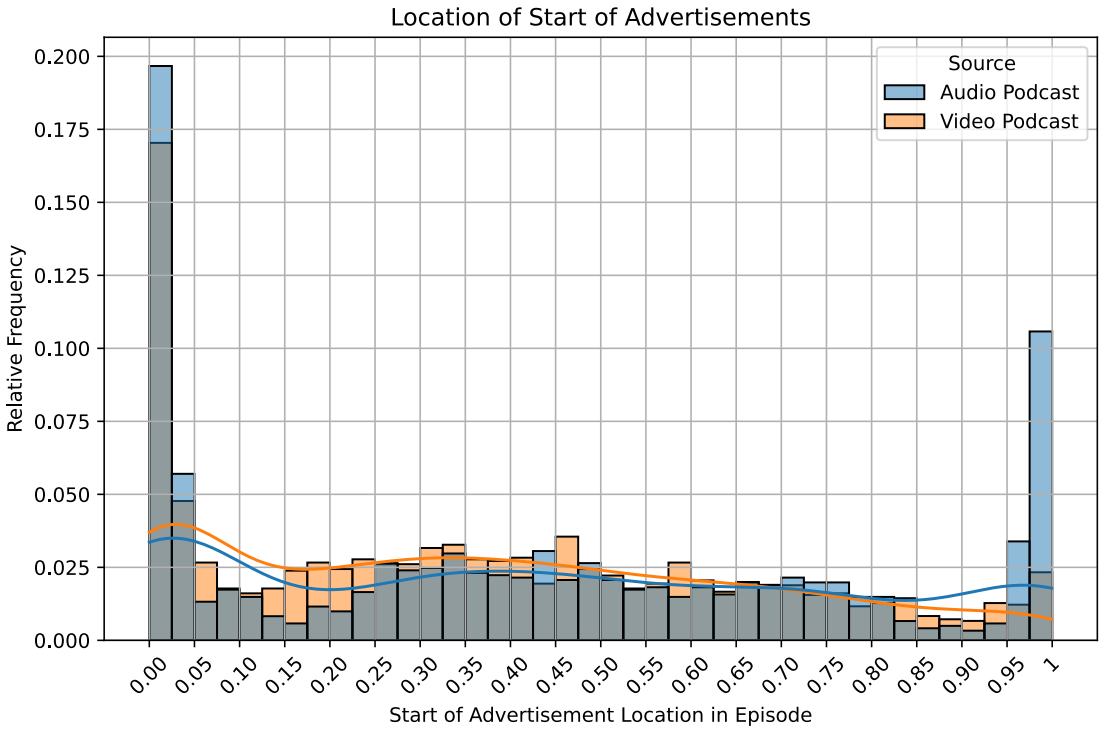
dency to have longer content segments between ad breaks compared to audio podcasts. But as we have learnt in Section 6.5, video podcasts only contain a single ad block in most cases. These episodes do not occur in the data for Figure 6.7, as we need at least two ad breaks to measure the distance.

6.7 Advertisement Location

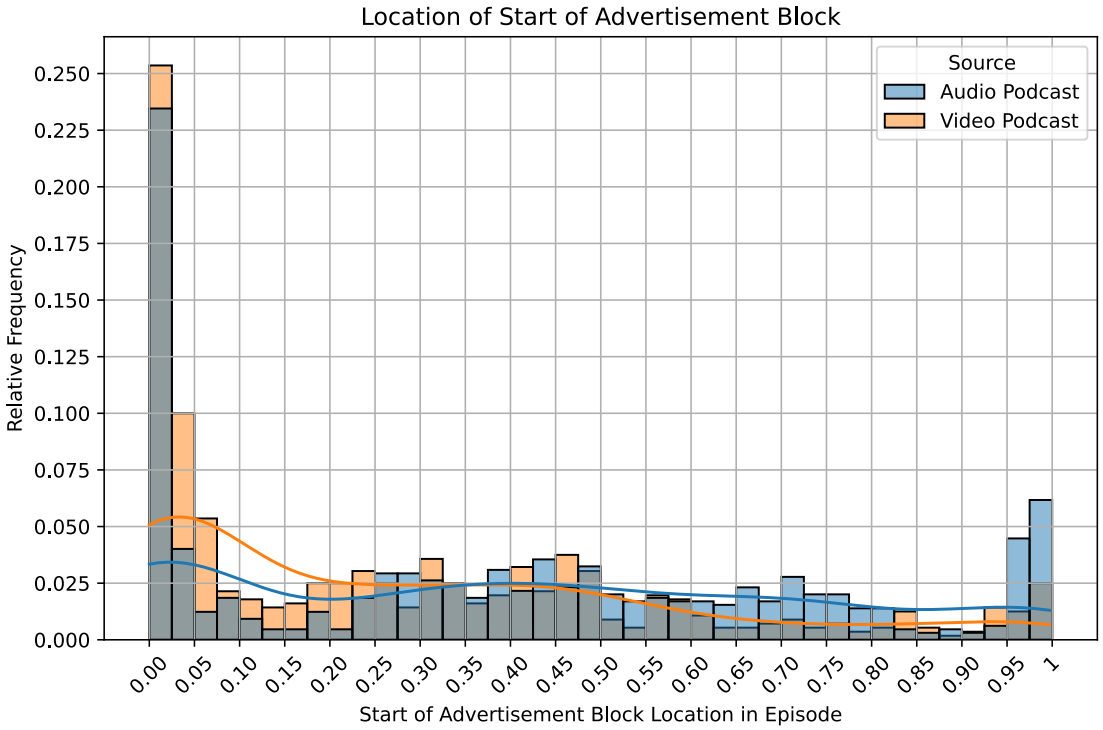
Next, we examine the location of an advertisement in its episode, beginning with individual advertisements. Figure 6.8a shows that both audio and video podcasts have most of their advertisements at the beginning of an episode.

Over the course of the episode, both sources have a similar distribution, with video podcasts having a tendency to have more advertisements in the earlier part of the episode. Noticeable is the fact that in audio podcasts, many episodes have trend to have advertisements at the end. While this can also be observed for the YouTube data, it is not as significant. Both sources share the characteristic that few ads are shown close to the end, around the 90% mark.

If we aggregate individual advertisements into blocks of consecutive advertisements, we see that in both sources, advertisements are typically placed at the beginning, in the middle and at the end of an episode (Figure 6.8b). Right after the beginning or just before the end, few advertisements can be expected.



(a) Location of start of advertisement over the course of an epsisode.



(b) Location of start of advertisement block over the course of an episode.

Figure 6.8: Analysis of beginnings of advertisements. We visualize the beginning of individual advertisement (top) and the beginning of advertisement blocks (bottom).

6.8 Advertisement Type

As stated in Section 5.1.1, where we describe the creation of the audio podcast dataset, we follow the differentiation by Bulakh et al. (2023) and categorize advertisements in audio podcasts into either inserted or self-voiced advertisements. We have also acknowledged that YouTube prohibits the use of pre-produced, 'inserted' advertisements in their terms of service (YouTube Help 2025b). We therefore assume that video podcasts sourced from YouTube contain only self-voiced advertising segments.

Figure 6.9 shows that the two advertisement types considered in this thesis are almost equally distributed in the audio podcast dataset.

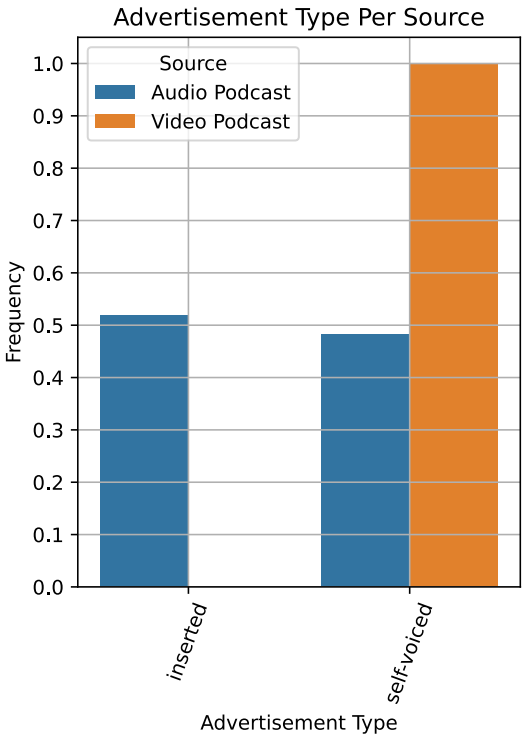


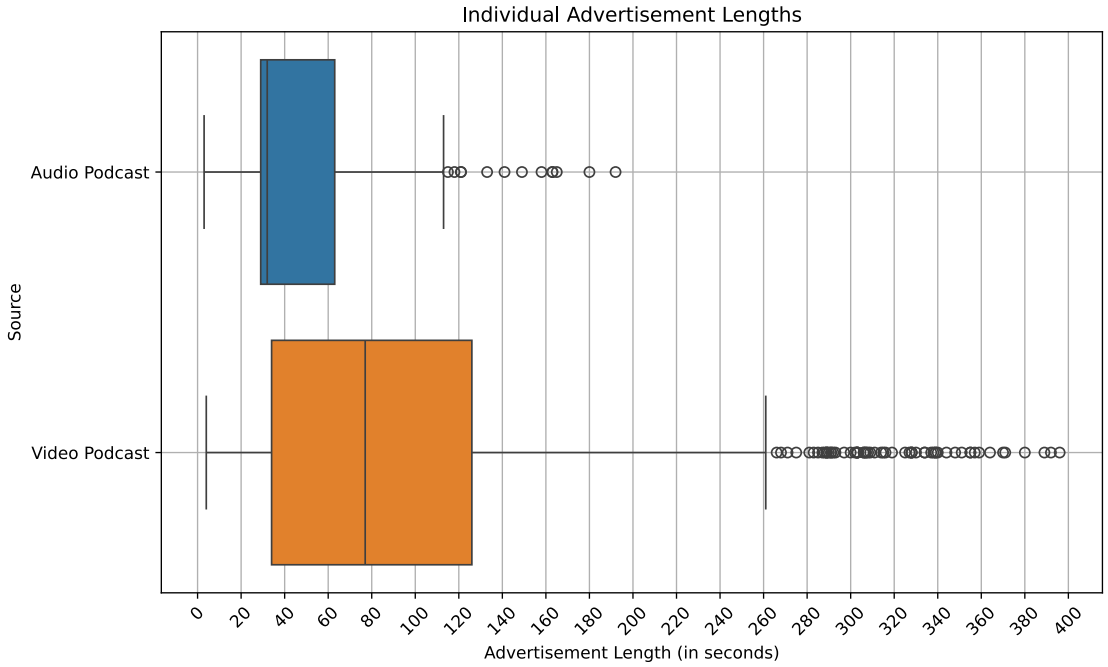
Figure 6.9: Distribution of inserted and self-voiced ads. We assume that video podcasts only contain self-voiced ads as YouTube prohibits the use per their terms of service.

6.9 Advertisement Length

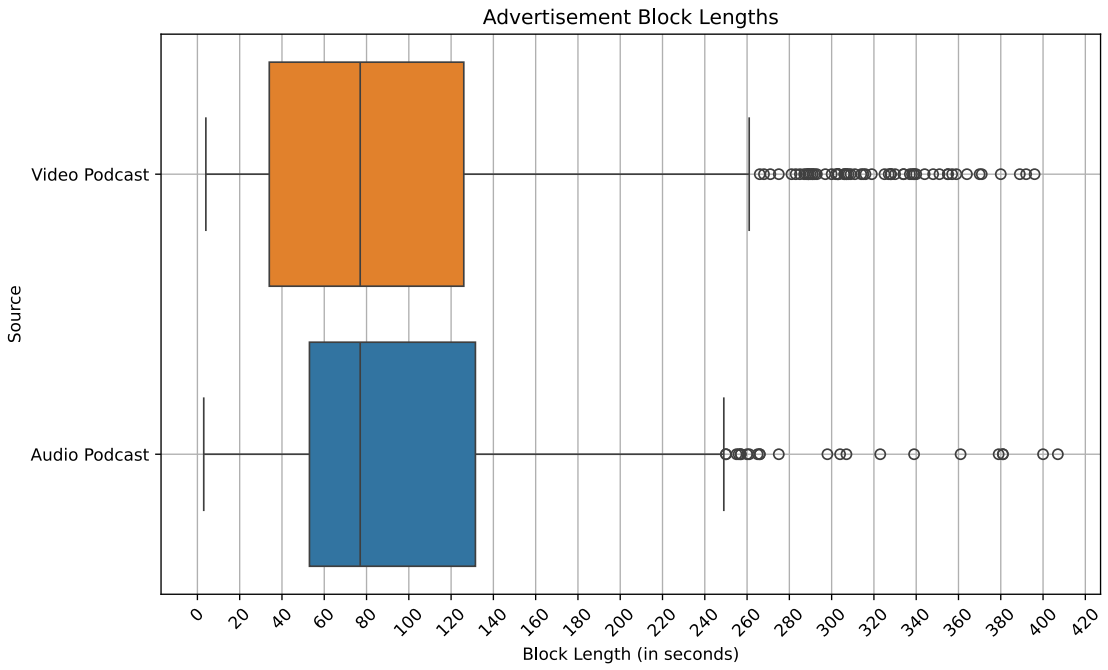
Now that we know the location and type of advertisements, we turn to their durations. Figure 6.10a visualizes that in audio podcasts, individual advertisement segments are on average 30 seconds long, with half of the segments being between 25 and 61 seconds in length. This is in stark contrast to video podcasts, where the average segment is ~75 seconds long. Additionally, advertisement length in video podcasts exhibits greater variability, with some segments being longer than five minutes. In audio podcasts, the longest advertisement has a duration of about three minutes (~190 seconds).

Taking into consideration that advertisements in audio podcasts come in blocks, we can examine the advertisement block length and compare that to the video podcast data. Figure 6.10b shows that now, the data is more closely aligned, with a median duration of ~75 seconds. This can be interpreted as advertising breaks from both sources are of roughly the same length.

As we have further differentiated the audio podcast advertisement segments into self-voice and inserted ads, we are able look at how the two types differ from each other in terms of length, which is shown in Figure 6.11. We observe that inserted ads are mostly grouped around the 30-second mark, with no segments going longer than 90 seconds. Self-voiced segments on the other hand have an average duration from ~60



(a) Lengths of individual advertisements, audio vs. video podcasts.



(b) Lengths of advertisement blocks, audio vs. video podcasts.

Figure 6.10: Comparing lengths of individual advertisements and advertisement blocks.

seconds with a wider spread. The hypothesis here is that inserted advertisements are all around 30 seconds in length as this is a well established duration for advertisements in multiple domains, especially in radio broadcasts (OFM 2024). Therefore, advertisers are used to produce 30-second advertisements and an ad produced for podcasts could also be aired in radio and vice versa. Self-voiced advertisements are almost always

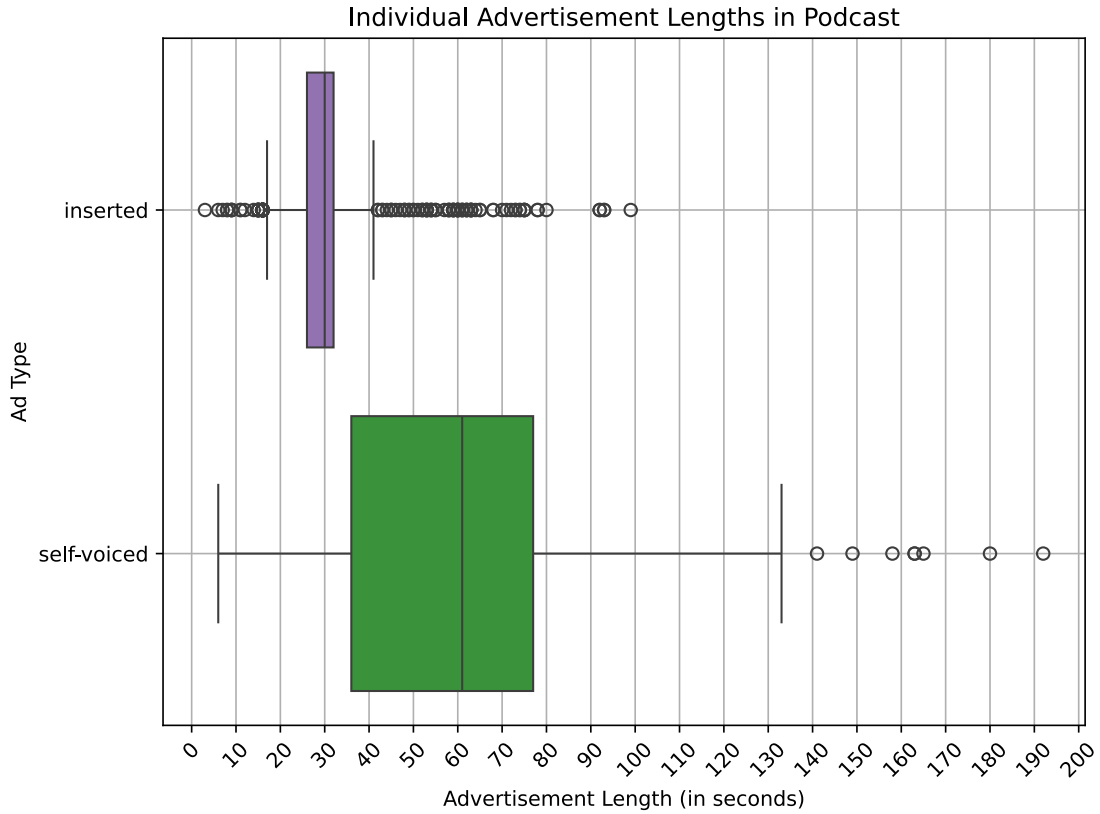


Figure 6.11: Ad Segment Length Distribution for self-voiced and inserted ads in audio podcasts.

produced to be aired in the podcast and therefore don't need to follow advertising industry standards, thus varying in style and length.

6.10 Correlation of Episode Length and Advertisements

Next, we look at how the length of an episode correlates with the total length of advertisements it contains. Figure 6.12 shows that at least for audio podcast, an increase in episode length correlates and increase in combined advertisement length. The same cannot be said about video podcasts, as it does not seem to be unusual for a long video podcast to include comparably short advertisements and for a short video podcast to contain relatively long advertisements.

Overall, the figure shows what became evident in the general statistics at the beginning of the chapter (Table 6.1): Audio podcasts tend to contain a higher ratio of advertisements to content. This is also shown in Figure 6.13: The median ad to episode length value for video podcast is 0.025, and it is 0.08 for audio podcasts.

6.11 Transcription Quality

When working with large artificial intelligence (AI) models trained on extensive volumes of unlabeled data, one has to be wary of model hallucinations (Rawte, Sheth, and Das 2023; Huang et al. 2025). Hallucination "refers to a situation where the model generates content that is not based on factual or accurate information." (Rawte,

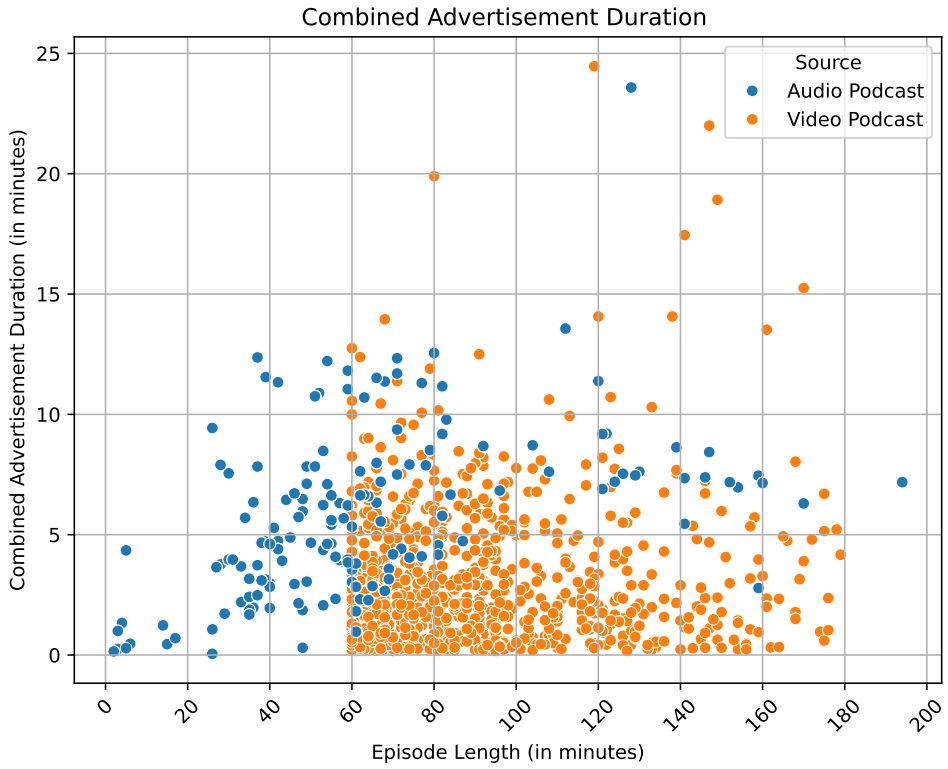


Figure 6.12: Correlation of episode length and total ad duration.

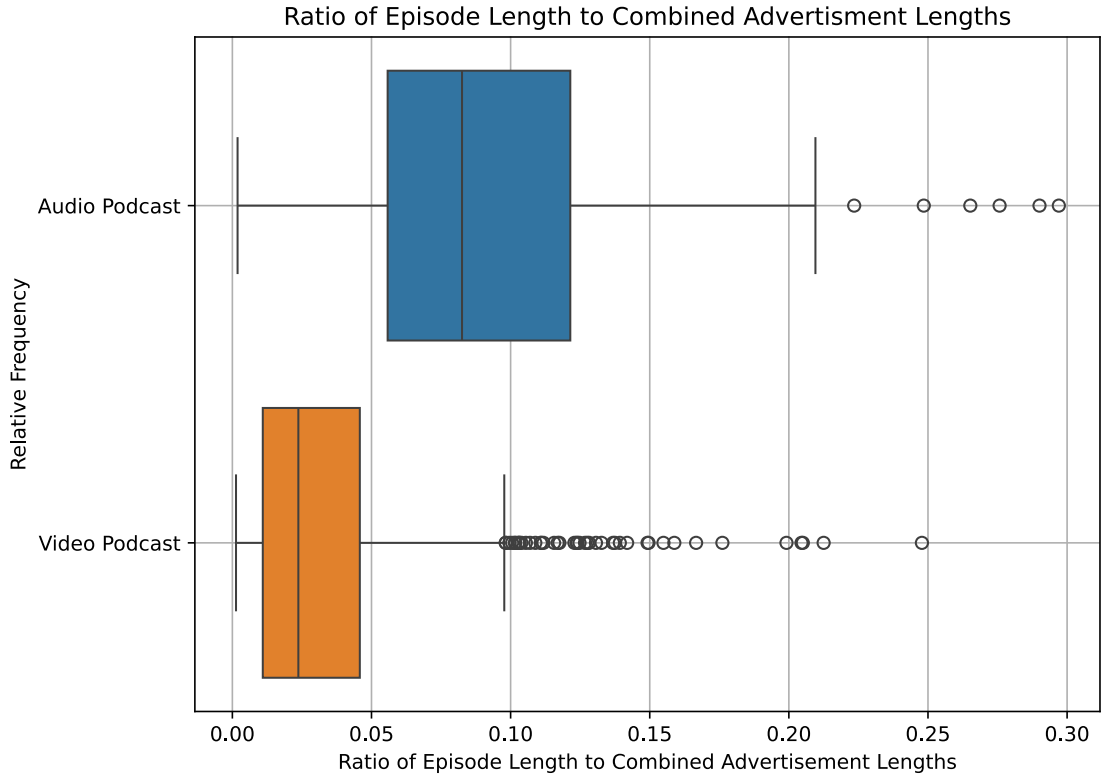


Figure 6.13: Ratio of advertisements to episode duration.

Sheth, and Das 2023). As shown by Rawte, Sheth, and Das (2023), addressing model hallucinations is crucial, since hallucinated content is often hard to distinguish from real content. This is especially problematic in areas where factual accuracy is required, such as journalism, healthcare and legal contexts (Rawte, Sheth, and Das 2023).

In our case, we find that the chosen whisper base model also suffers from hallucination. In most cases, the model repeatedly transcribes the last said sentence over and over, skipping over what was actually said. We observe this to happen when music is played, but also mid-conversation.

Therefore, we want to analyze the dataset in terms of model hallucinations. Having split the transcriptions into single sentences (as discussed in Chapter 5), we assume any sentence that is repeated at least three times in sequence to be a hallucination. This definition is precision-oriented, as transcription model hallucinations might not always result in sentence repetitions. However, it is important to acknowledge that hallucinations can also manifest in other forms, such as semantically inconsistent information, and that normal content may sometimes contain repetitions for emphasis or stylistic reasons. Despite these complexities, we believe this criterion provides a practical and robust starting point for identifying likely hallucinations within the dataset.

After checking all generated sentences in our dataset for hallucinations, we present the results in Figure 6.14. First, Figure 6.14a compares the total number of hallucinated sentences to non-hallucinations for both data sources. We see that for both sources, the number of hallucinated samples is almost equal to non-hallucinated samples. Our audio podcast dataset contains 45% hallucinated sentences, the video podcast datasets 52%.

Although half of all sentences being hallucinates seems severe, diminishing the corpus' quality, it's worth examining the percentage of hallucinations by time, since most hallucinations are on average shorter than non-hallucinations: Hallucinations have a mean duration of 0.45s over both sources and non-hallucinations have a mean duration of 4.82s. Figure 6.14b shows that in both datasets, hallucinated samples make up 6-9% of the total sample duration.

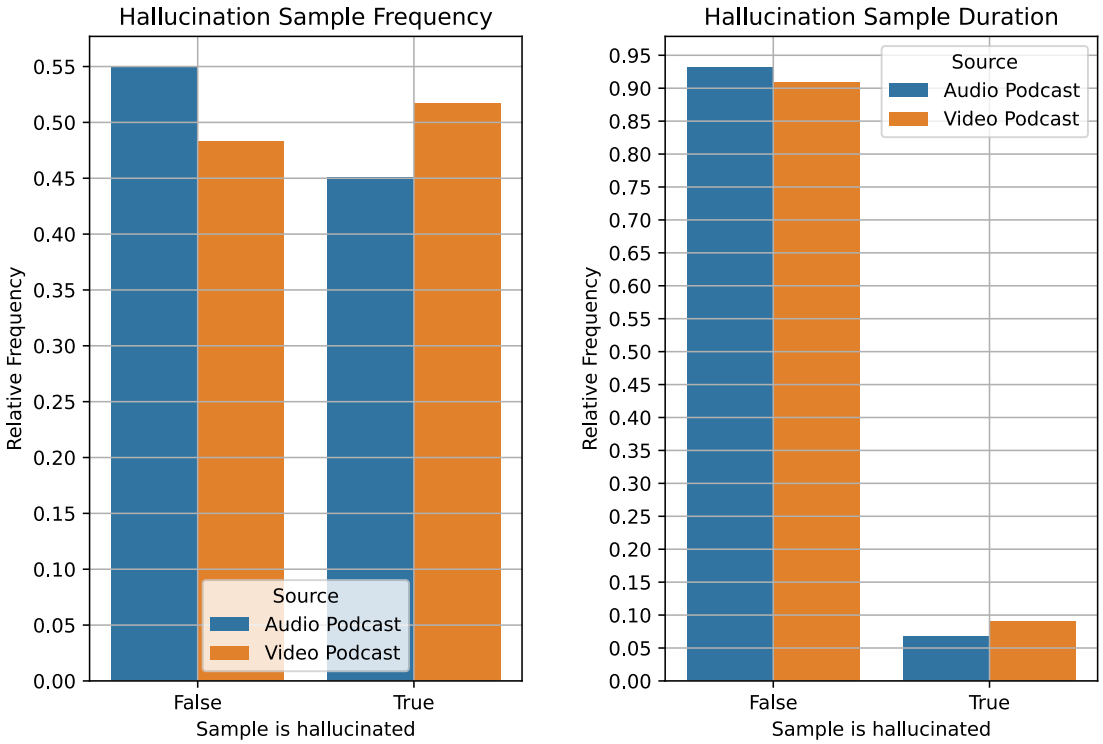
We therefore deem model hallucination as tolerable for this thesis. Hallucinated samples are excluded from all training and evaluation datasets.

6.12 Baseline Classification

Before concluding the chapter, we train two simple classifiers on the data to demonstrate the classification capabilities. We will do this for both audio and video podcasts, training a naive Bayes classifier using the bag-of-words text model and a support vector machine (SVM) using hand-crafted audio features on short segments sampled either from ads or non-ads.

6.12.1 Text Features

A multinomial naive Bayes classifier is a probabilistic machine learning model commonly used for classification tasks. The model is based on the Bayes' Theorem (Equation 6.1) and assumes feature independence.



(a) Relative Frequency of hallucinations by sample count.

(b) Percentage of total sample durations of hallucinations vs. non-hallucinations.

Figure 6.14: Sample hallucinations by count and duration.

$$P(y \mid X) = \frac{P(X) * P(X \mid y)}{P(y)} \tag{6.1}$$

Let y be a class, e.g. `AD`, and X be a feature instance, such as the word `promocode`, we want to compute the probability of class `AD` given the instance `promocode` ($P(y \mid X)$). Bayes' Theorem enables us to invert the conditional probabilities, and since $P(X)$, $P(y)$ and $P(X \mid y)$ can be computed by counting, $P(y \mid X)$ becomes trivial.

When using discrete counts or frequencies as features, we use the multinomial naive Bayes. We preprocess our transcribed corpora by splitting each segment (either `AD` or `NO_AD`) into word lists of length w_l, after removing stopwords. In this X, we chose $w_l = 185$, as this is the average word count in video podcast advertising segments. This method of segmentation requires the knowledge of segment borders, but as this is solely a baseline, this can be assumed. We transform the word lists into bag-of-words text representations. Essentially, we create a vector of the size of our vocabulary filled with zeros. Going through the wordlist, we increment the value by one at the current word's index for each time the word appears in the wordlist, similar to one-hot encodings. With this strategy, we get F_1 scores for audio and video podcasts of 0.91 and 0.72 respectively (Table 6.2).

Source	Precision	Recall	F1
Audio Podcast	0.97	0.85	0.91
Video Podcast	0.73	0.71	0.72
Combined	0.77	0.81	0.79

Table 6.2: Performance of multinomial naive Bayes baseline classifier trained on bag-of-words text features. 185 words per sample, stopwords removed.

As naive Bayes classifiers are probabilistic models, we can examine the empirical log probabilities assigned by the model after we've fit the data. For each feature instance w_i in our data (meaning every word in our vocabulary), the model computes probabilities for $P(y_{\text{AD}} \mid w_i)$ and $P(y_{\text{NO\_AD}} \mid w_i)$. Sorting the vocabulary by their $P(y_{\text{AD}} \mid w_i)$ values, we obtain a list of words order by their likelihood to appear in the 'AD' class. The top ten of those words are listed in Table 6.3a and Table 6.3b for audio and video podcasts respectively.

Both audio and video podcast share words with high likelihoods, such as 'com', 'slash', 'free' and 'get'. In podcasts, advertising messages often contain a call to action to visit the advertiser's website, which explains the use of URL-related terms ('com' and 'slash'). Words such as 'free' and 'get' are also often used in call to action messages, encouraging the user to 'get' a special promotion, or promoting a deal where the first x months of a subscription are 'free'.

Another information we can extract from the probabilities are words that have a high probability delta, meaning a high value of $P(y_{\text{AD}} \mid w_i) - P(y_{\text{NO\_AD}} \mid w_i)$. Put simply, these words appear frequently in `AD` segments and seldom in `NO_AD` segments. The top ten words for audio and video podcasts are listed in Table 6.3c and Table 6.3d. In almost all cases, these are the names of the advertisers (e.g. 'shopify', 'ag1' and 'betterhelp').

In summary, we see that we can perform a baseline classification using a naive Bayes classifier, assuming we know the segment boarder and use a large context size (185 words). Experimenting with fewer words per sample showed that classification performance decreases drastically. In terms of source comparison, the classifier performs better on audio podcasts compared to video podcasts, hinting at either a clearer distinction between `AD` and `NO_AD` features or a better dataset quality.

6.12.2 Audio Features

Similar to studies examined in Chapter 3, we can extract hand-crafted audio features and by that train a classifier on the audio data. As a baseline for audio classification, we will train a SVM classifier on typically used audio features, such as MFCCs, chromagrams, zero-crossing-rate and spectral contrast (Vimal et al. 2021; Logan and others 2000).

We preprocess the audio data by splitting the files into 10 second chunks, respecting the segment borders. Meaning if the first `AD` segment starts at second 5, we generate a `NO_AD` segment from 0 to 5. We fit our SVM for audio, video and combined data.

Word	Empirical log probability
com	-4.782607243482638
get	-5.101060974601173
slash	-5.319245252556032
like	-5.42146974055551
one	-5.507236562312936
free	-5.520112276672981
time	-5.668877913869351
go	-5.777306344358514
new	-5.80563685098474
know	-5.977666721742147

(a) Top ten words by their empirical log probability in the audio podcast dataset.

Word	Empirical log probability
get	-4.470392984553348
like	-4.498942362660067
com	-4.790379346504375
know	-5.021190155154074
go	-5.088297548359778
one	-5.132888923088265
right	-5.162216538182785
slash	-5.255742596193609
free	-5.304859611158563
going	-5.362152641677722

(b) Top ten words by their empirical log probability in the video podcast dataset.

Word	$P_{\text{AD}} - P_{\text{NO\_AD}}$
shopify	6.453521931261331
ag1	5.338780260663337
mint	5.236126106603253
5g	5.236126106603253
checkout	5.236126106603253
cashback	5.15156871857519
peloton	4.957412704134233
airbnb	4.945436513087518
cadillac	4.883304731980511
betterhelp	4.84408401882723

(c) Top ten words by their log probability delta in the audio podcast dataset.

Word	$P_{\text{AD}} - P_{\text{NO\_AD}}$
squarespace	7.9860231773099635
ag1	6.559988487540558
expressvpn	6.460616013727355
draftkings	6.414806477696061
userway	6.35026795655849
moxfield	6.127124405244279
undies	5.969495461040696
babble	5.944802848450324
betterhelp	5.903980853930069
fitbod	5.893509554062774

(d) Top ten words by their log probability delta in the video podcast dataset.

Table 6.3: Log probability analysis for multinomail naive Bayes vocabulary.

Table 6.4 illustrates baseline performance on audio features. For audio podcast, we have good precision, but bad recall, resulting in a classification score below those achieved using text features. For video podcasts, results are worse. Using more than 10% of the video podcast dataset results in an unreasonable fitting time (using scikit-learn's SVC classifier). Additionally, the class distribution in the YouTube data by nature is highly imbalanced, with the only 3% of samples being of class `AD`. Without resampling (see Section 2.4.1), the classifier refuses to predict `AD`. After resampling the data to 25% of ad class presence using undersampling, we get a F_1 score of merely 0.16.

Source	Precision	Recall	F1
Audio Podcast	0.94	0.42	0.58
Video Podcast[1]	0.8	0.09	0.16
Combined[1]	0.81	0.18	0.29

Table 6.4: Performance of SVM baseline classifier trained on hand-crafted audio features of samples of 10 seconds. Classifier for video and combined datasets use resamples class distribution due to naturally low-occuring minority class (see[1]).

6.13 Conclusion

The extensive data exploration performed in this chapter surfaced many relevant findings comparing advertisements in audio podcasts to those found in video podcasts. While showing similarities, both audio and video podcasts seemed to differ in ad-related characteristics. This directly answer the first research question RQ1:

RQ1: How can advertisements in podcasts be characterized and what differentiates advertisements in audio podcasts from advertisements in video podcasts?

Advertisements in audio podcast usually come in blocks of one to three individual advertisements, where video podcasts on average show advertisements individually (Figure 6.6). Furthermore, both sources often contain more than one advertising block, with audio podcasts most often having four blocks per episode and video podcasts one (Figure 6.5). If an episode contains more than one advertising block, the time between blocks is shorter in audio podcasts than in video podcasts (Figure 6.7). This is because audio podcasts have more advertising blocks, while also being shorter (Figure 6.2). This ratio between an episode's length and the total ad duration is also illustrated in Figure 6.13. Here, the trend continues, with the audio podcast's ratio being more than double that of video podcasts. Additionally, we see that for audio podcasts, the duration of advertisements seems to linearly increase with the episode's length. This does not seem to hold true for video podcasts.

We have also looked at where in an episode an advertisement is likely to occur (Figure 6.8). Both audio and video podcasts have a strong trend do include advertisements at the beginning of the episode. While audio podcasts also have a high relative frequency of advertisements at the end of the content, it's not as pronounced for video podcasts. If we aggregate individual advertisements to advertisement blocks, we observe that video podcasts have a tendency to start a block after a short introduction to the episode. This seems to occur less in audio podcasts. Overall, both sources have few advertisements just after the start of an episode or just before its end.

Looking at the duration of individual advertisements, those in audio podcasts are shorter and less diversified in length compared to those in video podcasts (Figure 6.10a). Aggregated to advertising blocks, both are similar in length (Figure 6.10). We can further differentiate between self-voiced and inserted advertisements in audio podcasts,

[1]Performed on subset of dataset of 50,000 samples with an artificial minority class presence of 25%.

following findings by Bulakh et al. (2023) and others. Here, we find that inserted advertisements are grouped around the 30-second mark, while self-voiced ads are usually around 60 seconds long and have a higher spread. We theorize that this can be attributed to inserted advertisements being produced to known industry norms, with 30 seconds being a well established advertisement duration in radio broadcast (OFM 2024). Using Figure 6.9, we conclude that both ad types are of equally distributed in audio podcasts, where video podcasts are assumed to only feature self-voiced advertisements, due to YouTube's ad policy restrictions.

In addition to ad-related metrics, we also examined non-ad-related metrics. We see in Figure 6.1 that the episodes in the videos podcast dataset have been uploaded over the last few years, with a tendency for 2021 to mid 2024. The earliest videos go back to 2012. Audio podcasts on the other hand are mostly from 2023 and 2024, with the majority being uploaded in October 2024. As discussed in Section 6.2, this is due to the data collection strategy, which differs for both sources. We have also examined the genre and tags of episodes (Figure 6.3). Unfortunately, as they come from different sources (audio podcasts: iTunes; video podcasts: YouTube), they cannot be directly compared.

Next, we have examined the transcription quality, regarding model hallucination (Figure 6.14). We find that the chosen base whisper model does hallucinate, repeating previous transcribed sentences numerous times. But, the repeated sentences are short, accounting for only a combined 8.8% of total dataset sample duration. We choose to exclude hallucinated samples as part of our data preprocessing but acknowledge the risk of other undiscovered dataset deficiencies.

Lastly, we trained simple classifiers on text and audio features extracted from the dataset in Section 6.12. For both experiments, we assume we know segment boundaries. We find that we achieve reasonable results using a naive Bayes classifiers on bag-of-words text representations, given we increase the length of each sample to 185 words (without stopwords). Audio podcasts work better than video podcasts, across both precision and recall scores (Table 6.2). We examined assigned feature instance probabilities in Table 6.3, surfacing words with high log probabilities for the `AD` class and words with high probability deltas between classes.

Using audio features on the other hand, we face difficulties getting the classifier to predict samples as `AD` without altering the class balance using undersampling. After doing so, performance for video podcasts is far below that for audio podcasts, hinting at differences in audio features across sources (Table 6.4).

7 Results

In this chapter, we present our results. We structure our findings according to the experiments carried out. Chapter 8 discusses the results and puts them into context.

Important: We have trained and evaluated many different models, each using various combinations of training data, evaluation data, data modality, and ad class percentage. All results are summarized in the tables Table 7.1 and Table 7.4. Throughout this chapter and in Chapter 8, focused subsets of these results are presented to support specific comparisons and discussions. However, all subsets are derived from the same full results in the main tables. Each model is labeled with an identifier (A-P), and these identifiers always refer to the same model across all tables. In performance tables, the highest score in each column is highlighted in green, and the lowest in red, to make comparisons easier.

7.1 Modality Information Gain

In our first experiments, we analyze how a model's performance varies when trained on embeddings computed from text, embeddings computed from audio or a combination of both. For clarity, Table 7.2 presents an excerpt of Table 7.1.

First, we will look at models trained on audio podcasts. Here, models A, E and G are all trained on audio podcasts. Using undersampling of the majority class (`NO_AD`), we set the minority class presence to 10%, which is close to the natural presence of 9.5% in our audio podcast training dataset. In this experiment, we train and evaluate models on the same source of podcasts (audio / video).

Model E is trained on embeddings computed from texts and achieves F_1 and $F_{0.5}$ scores of 0.71 and 0.79 respectively. Model G is trained on embeddings computed from audio and scores similarly to Model E, with F_1 and $F_{0.5}$ scores of 0.73 and 0.82. Lastly, model A is trained on a combination of audio and video data by concatenating the

Id	Train	Eval	Modality	Percent Ad Class	Samples	Pre	Rec	F_1	$F_{0.5}$
A	Audio Podcast	Audio Podcast	text + audio	10%	63,833	0.94	0.69	0.80	0.87
		Video Podcast				0.30	0.04	0.07	0.13
B	Video Podcast	Audio Podcast			180,005	0.70	0.11	0.19	0.34
		Video Podcast				0.50	0.25	0.33	0.42
C	Audio Podcast	Audio Podcast		20%	31,916	0.74	0.86	0.80	0.76
		Video Podcast				0.22	0.09	0.13	0.17
D	Video Podcast	Audio Podcast			90,002	0.54	0.47	0.50	0.53
		Video Podcast				0.38	0.34	0.36	0.38
E	Audio Podcast	Audio Podcast	text	10%	63,833	0.88	0.59	0.71	0.79
		Video Podcast				0.29	0.27	0.28	0.29
F	Video Podcast	Audio Podcast			180,005	0.77	0.33	0.46	0.61
		Video Podcast				0.56	0.32	0.40	0.49
G	Audio Podcast	Audio Podcast	audio	10%	63,833	0.90	0.61	0.73	0.82
		Video Podcast				0.04	0.00	0.00	0.00
H	Video Podcast	Audio Podcast			180,005	0.59	0.05	0.10	0.20
		Video Podcast				0.47	0.04	0.07	0.14

Table 7.1: Performance evaluation of local classifiers, samples chosen at random, 10 epochs, batch size of 32.

computed LanguageBind embeddings. It outperforms both previous models, with F_1 and $F_{0.5}$ scores of 0.80 and 0.87.

Id	Train + Eval	Modality	Percent Ad Class	Samples	Pre	Rec	F_1	$F_{0.5}$
A	Audio Podcast	text + audio	10%	63,833	0.94	0.69	0.80	0.87
E		text			0.88	0.59	0.71	0.79
G		audio			0.90	0.61	0.73	0.82
B	Video Podcast	text + audio		180,005	0.50	0.25	0.33	0.42
F		text			0.56	0.32	0.40	0.49
H		audio			0.47	0.04	0.07	0.14
C	Audio Podcast	text + audio	20%	31,916	0.74	0.86	0.80	0.76
D	Video Podcast			90,002	0.38	0.34	0.36	0.38

Table 7.2: Evaluation of modality information gain. Models are trained and evaluated on either audio or video podcasts and on text, audio or combined embeddings. Excerpt from Table 7.1.

Next, we turn to models trained on video podcasts, models B, F and H. Using undersampling of the majority class, we again set the minority class presence to 10%. We hypothesize that this helps us compare model performance over source choices. Initially, we trained a model on video podcasts without addressing the imbalanced data problem, which resulted in the model never predicting the minority class, as the `AD` class has a natural presence of 3.5% in the video podcast training data set (compared to the 9.5% for audio podcast).

Analyzing the results, we observe that all three models trained on video podcasts perform far worse than those trained on audio podcasts. Training on texts, audios and both modalities (models F, H and B), we achieve F_1 scores of 0.40, 0.07 and 0.33 respectively. $F_{0.5}$ score are slightly higher with 0.49, 0.14 and 0.42. Interesting to note is that here, the model trained on solely texts outperforms the model trained on text and audio, which was not the case for audio podcasts. Furthermore, the model trained on solely audio demonstrates poor performance, due to a notably low recall score. This indicates that the models fails to detect a significant number of advertisements, resulting in a high rate of false negatives. Recall that in audio podcasts, the addition of audio features improves the F_1 score by nearly 0.1. However in video podcasts, adding audio features decreases performance.

Regarding the imbalanced data problem, we've experimented with different class distributions. Again using undersampling of the majority class, we train model C and D on datasets where the minority `AD` class has a presence of 20%. As it is evident from our results, this hurts precision but boosts recall. While F_1 scores stay at a similar level, as they value precision and recall equally, we see a decrease in $F_{0.5}$ scores compared to models trained with 10% minority class presence. An increase in the minority class presence results in a decrease of overall number of samples. This is because once

we already use all of our minority class samples in the training dataset, we can only increase its relative frequency by sampling less majority class samples.

Overall, we can now formulate an answer to RQ2:

RQ2: How does model performance vary when using audio, text, or multimodal training data?

We state that for audio podcast data, we can increase model performance by using both audio and text modalities as input, outperforming models trained on single-modalities. These findings cannot be transferred to video podcasts, where a multimodal model performs worse than a text-modality model. For video podcasts, it's also necessary to create an artificial minority class presence of 10%, with a presence of 20% decreasing precision but increasing recall.

7.2 Out-of-domain Training data

In our second experiment, we examine how well out-of-domain podcast data sourced from YouTube can be used to train models that will be used for inference on in-domain audio podcasts. To reiterate, this can be motivated by the fact that there does not exist a dataset of advertisements in audio podcasts, requiring us to manually annotate a self-built dataset. For video podcasts, we can use existing advertisement timestamp databases for YouTube videos as found in the SponsorBlock project (Ajay Ramachandran 2024) to automate the annotation process, allowing us to build a larger corpus in less time.

We present our results in Table 7.3. Models B, F, H and D are trained on video podcasts and evaluated on audio podcasts. Again using undersampling, the minority class presence is set to 10% for models B, F and H and to 20% for model D. No matter what minority class presence models are trained on, they are always evaluated on the original dataset with its original class distribution. Models B and D are multimodal, while model F is trained on text embeddings and model H on audio embeddings. All models are trained for 10 epochs.

Id	Train	Eval	Modality	Percent Ad Class	Samples	Pre	Rec	F_1	$F_{0.5}$
B	Video Podcast	Audio Podcast	text + audio	10%	180,005	0.70	0.11	0.19	0.34
F			text			0.77	0.33	0.46	0.61
H			audio			0.59	0.05	0.10	0.20
D			text + audio	20%	90,002	0.54	0.47	0.50	0.53

Table 7.3: Performance results of OOD training data. Models are trained on video podcasts and evaluated on audio podcasts. Different modalities are compared. Excerpt from Table 7.1.

In our results, we see that the model trained and evaluated on text embeddings performs best with OOD data. Trained on audio data however, model H's performance is the lowest among all. This is again due to a poor recall score. Combining text and audio modalities does increase all performance metrics, but model B is still outperformed by model F. Increasing the minority class presence to 20% using multimodal embeddings (model D), we see a familiar trend: Recall increases while precision decreases (compared to model B). Model D thereby performs the second best, while model H performs the worst.

With these findings, we can answer RQ3:

RQ3: Can training data from video podcasts be used to detect advertisement in audio podcasts?

Overall, feature characteristics learned from OOD training data do not transfer over to in-domain audio podcasts. While the model trained on video podcast text embeddings is able achieve a $F_{0.5}$ score of 0.61 when evaluated on audio podcasts (Table 7.3, model F), training directly on in-domain audio podcast text embeddings (Table 7.2, model E) scores a higher $F_{0.5}$ score of 0.79.

The model trained on video podcast audio embeddings performs worse, achieving a $F_{0.5}$ score of only 0.20, caused by a particularly low recall score. This hints at stark differences in audio characteristics between audio and video podcasts. Combining audio and text features improves overall performs over audio only models, but still lacks behind text only models.

7.3 Superlocal Context

In our last experiment, we analyze how increasing the context of the model architecture impacts classification performance. To increase the context, we concatenate intermediate sample representations before feeding them into a classifier, as described in Section 5.6. We have tried contexts size of 2 to 128. The results are shown in Table 7.4.

Id	Samples	Context Size	Training Steps	Precision	Recall	F_1	$F_{0.5}$
I	64,896	128	507	0.74	0.11	0.19	0.35
J	64,896	64	1014	0.84	0.32	0.46	0.63
K	67,264	32	2102	0.77	0.73	0.75	0.76
L	67,280	16	4205	0.80	0.75	0.77	0.79
M	67,288	8	8411	0.87	0.79	0.82	0.85
N	67,292	4	16823	0.80	0.86	0.83	0.82
O	67,294	2	33647	0.82	0.85	0.83	0.82
P	4,204	2	2102	0.85	0.68	0.76	0.81

Table 7.4: Performance results of 'superlocal' classifiers. Base model is model A in Table 7.1. All models trained and evaluated on in-order audio podcast samples for 10 epochs.

As the superlocal architecture requires a 'local classifier' model to compute the intermediate sample representations, we choose model A (Table 7.1), which is our best performing model, trained and evaluated on audio podcasts. It scores F_1 and $F_{0.5}$ scores of 0.80 and 0.87, which is the baseline for this experiment. Due to an implementation detail, the number of training steps correlates with the chosen context size (see Section 8.2.3).

As we can see, no significant improvement can be made at any of the tried context sizes. Models with context sizes 2, 4 and 8 (model O, N, M) outperform our baseline model in F_1, but show worse $F_{0.5}$ scores. Overall, model performance decreases with increasing context sizes. The recall in particular sees a drastic decline from context size 32 to 64. We will hypothesize about the reasons in Section 8.2.3. Model P is in its hyperparameters equal to model O, but is trained on $\frac{1}{16}$ of its data. The reasoning will also follow in the discussion.

7.4 Advertisement Type Classification Performance

Since we have looked into how self-voiced ads differ from inserted ads in greater details over the previous chapters, especially in the data exploration, it is worth to examine if one ad type is easier to classify than the other. For this reason, we have split the audio podcast dataset into two subsets, each containing all `NO_AD` samples but only `AD` samples of either self-voiced or inserted advertisements. We train models on both subsets and on all three modalities (text, audio and multimodal) for 10 epochs with a minority class presence of 10%.

Table 7.5 holds the results. For inserted advertisements, the audio modality performs the best for both F_1 and $F_{0.5}$ scores. The text modality presents a balanced, but lower result over the two scores. Interestingly, the model trained on multimodal input has the highest recall score but lowest precision score, achieving the lowest $F_{0.5}$ score among the three models trained on inserted advertisements.

Models trained on self-voiced advertisements indicate a difference in the datasets. Here, the model trained on audio embeddings performs the worst in $F_{0.5}$. This is in contrast to inserted advertisements, where audio embeddings performed best. The

Train + Eval	Modality	Samples	Pre	Rec	F_1	$F_{0.5}$
Audio Podcast Inserted Ads	text + audio	25,683	0.69	0.90	0.78	0.72
	text		0.74	0.72	0.73	0.73
	audio		0.79	0.88	0.83	0.81
Audio Podcast Self-Voiced Ads	text + audio	38,899	0.81	0.70	0.76	0.79
	text		0.81	0.50	0.62	0.72
	audio		0.70	0.70	0.70	0.70

Table 7.5: Performance results of audio podcast ad types. Different modalities are compared.

model trained on multimodal input outperforms the other two in all four scores. Overall, scores are comparable between both advertisement types, with a mean F_1 score of 0.74 and a standard deviation of 0.44.

7.5 Optimal Decision Thresholds

When evaluating our models, we set a fixed value as the decision threshold in our classifiers. Local classifier output layers emit a single real number between zero and one, that it is mapped to either 1 (predicting `AD`) or 0 (prediciting `NO_AD`) depending on the decision threshold. Up until now, we always chose 0.5 as our threshold.

In Figure 7.1, we examine if we can improve classification performance by optimizing the threshold value. In the figures, the F_1 (solid line) and $F_{0.5}$ (dashed line) scores over all decision thresholds from zero to one for a particular model are visualized. The red dot marks the optimal decision threshold, maximizing the respective score. For clarity, we mark the previous chosen threshold of 0.5 in gray. We have model performance evaluated on audio podcasts in blue and evalauted on video podcasts in orange.

First, we look at Figure 7.1a, which holds results for model A, that was trained on combined embeddings of audio podcasts. When evaluated on audio data, we see that $F_{0.5}$ performance cannot be improved any further, as a decision threshold of 0.5 is already close to optimal. If we were interested in F_1 score however, we could improve the score by 0.03 choosing a decision threshold of ~0.32. Evaluated in video podcasts, both metrics can be improved by choosing a lower threshold, in the case of F_1 we see an improvement by 0.1. Overall, choosing a lower decision threshold seems to be effective in this case.

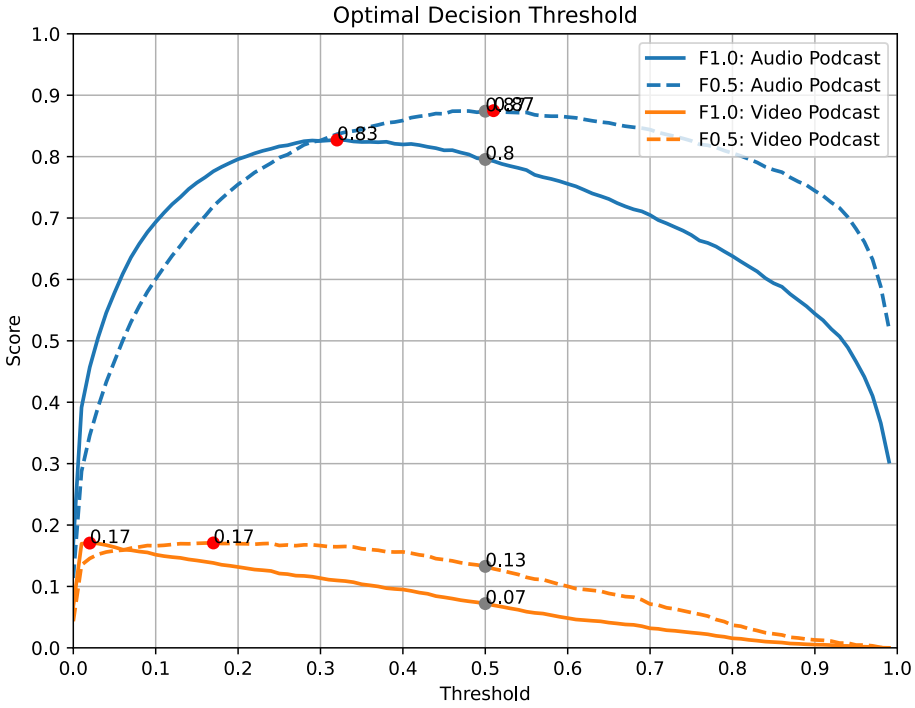
The same holds for model B, which was trained on video podcasts (Figure 7.1b). Here we observe that in three of the four cases, decreasing the decision threshold improves the performance metric. Interestingly, after choosing optimal thresholds, the model trained on video podcasts now performs better when evaluate on audio podcasts. Before, the opposite was the case. Table 7.6 compares performances between old and new decision thresholds.

7.6 ROC Curves

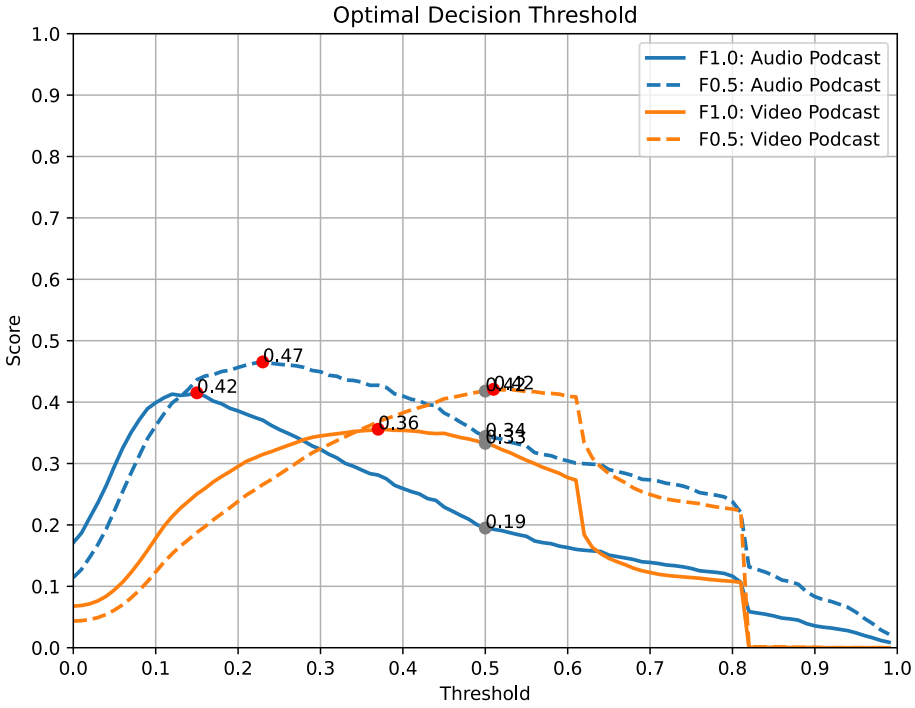
In this section, we will briefly look at receiver operating characteristics (ROC) curves of our multimodal models A and B. Figure 7.2a and Figure 7.2b display the ROC curves for model A and B respectively.

In Figure 7.2a, model A achieves a very high performance on the audio podcast dataset, with an Area Under the Curve (AUC) of 0.98, indicating near-perfect discrimination between advertisements and non-advertisements. Conversely, the performance on the video podcast dataset is substantially lower, suggesting that the model does not generalize well from one source to the other.

In Figure 7.2b, model B shows a more balanced but overall lower performance across the datasets. On the audio podcast dataset, it achieves an AUC of 0.72, which is



(a) Optimal decision threshold of model A when evaluated on audio vs. video podcasts.

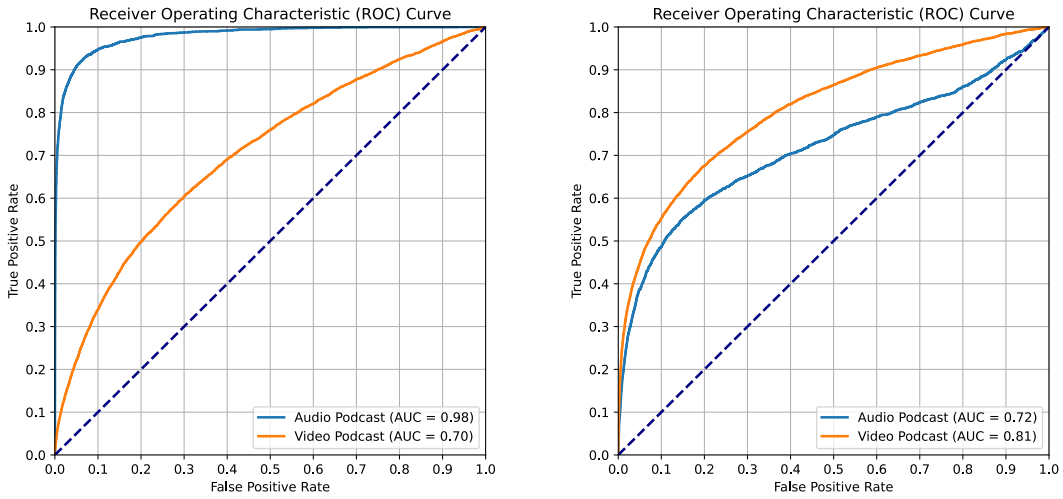


(b) Optimal decision threshold of model B when evaluated on audio vs. video podcasts.

Figure 7.1: Evaluation of decision threshold performance for model A and B from Table 7.1. Models are evaluated on either audio podcast (blue) or video podcast dataset (orange). Performance evaluated with F_1 (solid) and $F_{0.5}$ score (dashed).

Id	Train	Eval	F_1old	$F_{0.5}$old	F_1opt	$F_{0.5}$opt
A	Audio Podcast	Audio Podcast	0.80	0.87	0.83	0.87
		Video Podcast	0.07	0.13	0.17	0.17
B	Video Podcast	Audio Podcast	0.19	0.34	0.42	0.47
		Video Podcast	0.33	0.42	0.36	0.42

Table 7.6: Findings summary of decision threshold experiments, comparing model performance at decision threshold 0.5 to performance at optimal threshold. Optimal threshold is model dependent.



(a) ROC curve of model A.

(b) ROC curve of model B.

Figure 7.2: ROC curve comparison of model A and B from Table 7.1. Models are evaluated on either audio (blue) or video podcasts (orange).

significantly worse than model A's performance. On the video podcast dataset, model B achieves an AUC of 0.81, outperforming model A on the same dataset. This is the expected result, as model B was trained on the video podcast dataset.

Overall, model A is highly effective when evaluated on audio podcast data, but model B shows a more balanced performance between the datasets. The true positive rate, the false positive rate and the associated decision thresholds computed by the ROC can be used to assess a model's practical effectiveness given a maximum error rate, as will be shown in the next section.

7.7 Practical Performance Evaluation

Before concluding the chapter, we want to assess a model's performance in a real-word example. Consider an advertisement detection system intended to remove unwanted advertisements from a user's requested podcast audio file. As previously noted, such a task is precision oriented, as falsly removing content (due to false positive) is consid-

ered worse than not removing advertisements (due to false negatives). Consequently, we have measured models $F_{0.5}$ score in prior sections.

7.7.1 True Positive Rate Calculation

For our example, we assume a hypothetical user requires that the system produces, on average, no more than one false positive every 30 minutes. In other words, the system should have a mean time between false positives (MTBFP) of at least 30 minutes. To translate this requirement into a model's acceptable FPR, we consider the average sample length in our dataset. In the audio podcast dataset, each sample (i.e., each sentence) has an average duration of 4.52 seconds (after excluding model hallucinations). Knowing that, we can say that a MTBFP of 30 minutes equals a FPR of $\frac{1}{398}$, as there are on average 398 samples in 30 minutes (Equation 7.1).

$$\frac{1 \text{ false positive}}{\frac{30*60 \text{ seconds}}{4.52}} \approx \frac{1}{398} \tag{7.1}$$

Using the desired FPR, we find the next higher FPR in our retrieved FPRs, true positive rates (TPRs) and thresholds from the ROC calculation. The associated TPR tells us the percentage of advertisements the model will detect without producing more than one false positive per 30 minutes. This data is presented in Table 7.7. Using a decision threshold value of 0.6272, the model correctly identifies and removes 61% of all advertisement sentences.

Assuming the user has stricter performance requirements, we can calculate the effective TPR for a MTBFP of e.g. 60 or 120 minutes as well. At least for the considered FPRs, the effective TPR seems to decrease linearly to 48% and 39% of detected advertisement samples.

7.7.2 Model Prediction Behavior

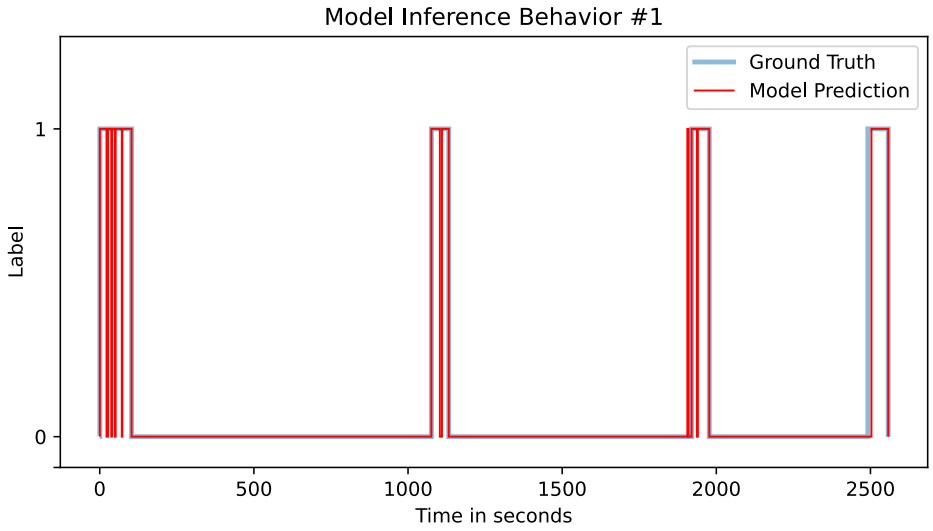
In the previous section, we've seen that the best performing model (A) is capable of identifying 61% of all positive samples, given MTBFP of 30 minutes. In this section, we will visualize the model predictions for a randomly selected podcast episode in our dataset and briefly discuss its deficiencies. Figure 7.3 shows the predictions of model A for two separate podcast episodes using the decision threshold for a MTBFP of 30 minutes as calculated in Table 7.7. The ground truth values are marked in blue, the model predictions in red. Model hallucinations in the transcription are marked in orange.

Id	MTBFP	FPR	TPR	Threshold
A	30min	$\frac{1}{398} \approx 0.00251$	0.6072	0.6272
	60min	$\frac{1}{796} \approx 0.00126$	0.4814	0.7906
	120min	$\frac{1}{1592} \approx 0.00063$	0.3920	0.8852

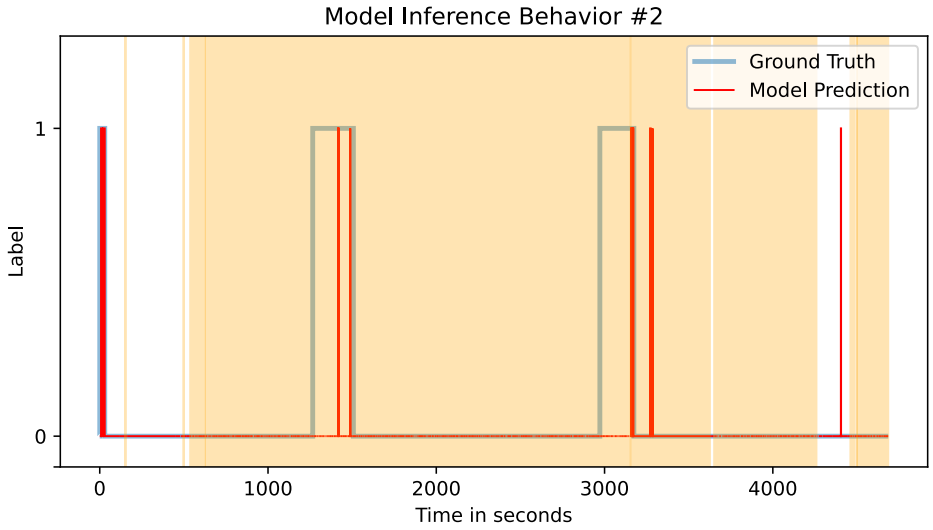
Table 7.7: Practical performance evaluation. Shows TPR and associated decision threshold given a mean time between false positives (MTBFP) of 30 / 60 / 120 minutes.

Figure 7.3a shows a well working example. The transcription does not contain any hallucinations and the model identifies each advertisement block, although not completely: In the first three blocks, the model flips from `AD` to `NO_AD` during the block. The last advertisement block is detected too late.

In Figure 7.3b, the performance is worse. Here, the transcription contains many hallucinations. In those areas, the model has to rely on the audio data, as the text data does not hold much value. As we can see, the most of the advertisement blocks are missed by the model. Additionally, the model flags a couple sentences at the end falsely for advertisement, producing false positives.



(a) Predictions for episode “Noble - The Missing Right Arm Chapter 7”



(b) Predictions for episode “Bad Friends - Keep Sketch Away From Loop Loop”

Figure 7.3: Inference behavior of model A using a decision threshold of 0.6272. Hallucinations in transcription are marked in orange.

7.8 Conclusion

In this chapter we have performed various performance evaluations and experiments. We presented results comparing different podcast sources, input modalities and class balances. Decision threshold values were evaluated and ROC curves analyzed. In summary, we have found that advertisements in audio podcasts were best identified when models are trained on real, in-domain audio podcast data. Using OOD video podcast data sourced from YouTube does not work as well, with models trained on the video podcast dataset failing to achieve decent performance scores when evaluated on audio podcasts.

Interestingly, evaluating said models on same source video podcasts does yield higher scores, but results are still below audio podcast performances. Using multimodal input features in audio podcasts scored the best, followed by only audio and only text. In contrast, video podcast models perform worse when using multimodal features, likely due to a particularly bad performance of the audio feature space.

Comparing overall performance trends to those of the baseline classifiers in Table 6.2 and Table 6.4, we observe similarities. The naive Bayes classifier trained on audio data scored high, with the audio podcast dataset outperforming the video podcast dataset. The same is true for our Transformer-based *local* classifier. The Bayes classifier achieved a F_1 score of 0.91, using a context size of 185 words and relying on proper class segmentation within the samples. As models in this chapter classify on sentence level, and as sentences in the audio podcast dataset have on average 16 words, the two approaches cannot be directly compared. The SVM trained on audio data performed worse for video podcast compared to audio podcast, which we also find in this chapter. It especially struggles when trained on video podcast data, as do our Transformer-based models. In summary, performance of Transformer-based models is similar to baseline models while using much less input context and without requiring perfect segmentation.

We have also compared self-voiced to inserted ads in audio podcasts. Here, we conclude that the audio modality is more feature-rich in regard to advertisement detection for inserted ads compared to self-voiced advertisements. With respect to precision performance, self-voiced advertisements have an edge over inserted advertisements due to a high precision score for text embedding models.

Lastly, we've evaluated our best performing model in a real-world scenario. Assuming an arbitrarily set mean time between false positives (MTBFP) of 30 minutes, we find that the model achieves a TPR of ~0.61, effectively identifying more than half of all advertisements. Over the course of two exemplary episodes, we've seen that a model targeting a MTBFP of 30 minutes is able to pick up on most advertisement blocks, but hallucinations in the transcription worsen the performance.

To recap, our best performing model is model A of Table 7.1. It is trained on audio and text embeddings from audio podcasts, with an ad class percentage of 10% and a decision threshold of 0.5. It scores 0.8 and 0.87 in F_1 and $F_{0.5}$.

8 Discussion

This chapter provides a critical analysis of the results. Where possible, we will provide an interpretation of our findings. In an OOD setup, evaluating on video podcasts exhibits worse classification performance compared to evaluating on audio podcasts. We will perform an investigation for a possible cause. The chapter is concluded by a general discussion of the limitations that the system faces.

8.1 Error Analysis

As illustrated in the previous section, models trained on the video podcast dataset do not perform well when evaluated on audio podcasts. In fact, they also do not perform well evaluated on the test part of the video podcast dataset. This section will investigate why that is. If a model does not perform well for a given classification task, it is either because of low annotation quality or the model design is unable to capture the intricacies of the problem. But since we are able to train well performing models on audio podcast data, model design is deemed to be an unlikely cause.

8.1.1 Confusion Matrix

In Figure 8.1, the confusion matrix of model B from Table 7.1 is shown.

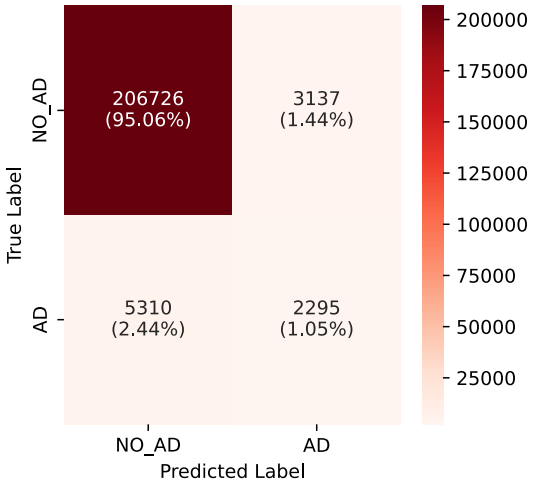


Figure 8.1: Confusion matrix of model B from Table 7.1, trained and evaluated on video podcast text + audio embeddings. Most predictions are TNs. The model produces more FNs and FPs than TPs.

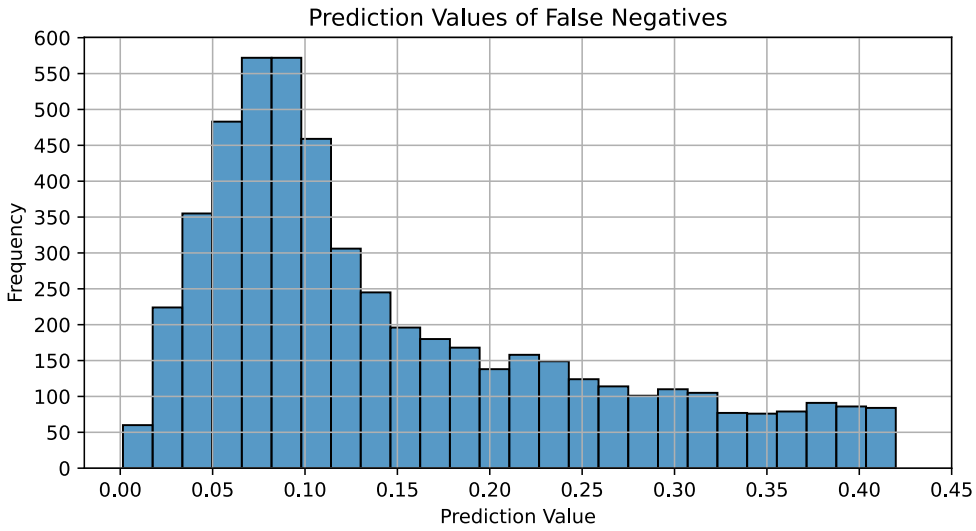


Figure 8.2: Prediction values of false negatives from model B from Table 7.1, trained and evaluated on the video podcast dataset, using a decision threshold of 0.42.

To recap, the model is trained using text and audio embeddings from video podcasts as features. It is then evaluated on the same dataset, using a decision threshold of 0.42, as the $F_{0.5}$ score is maximized at this threshold (as per Figure 7.1b).

Once again, we see that the data is highly imbalanced, with 7605 samples (3.5%) being of class `AD`. Of those samples, only 2295 (30%) were correctly identified as advertisements. In other words, the model misses 70% of all relevant instances. It also produces false positives, but only to a negligible degree, with 3137 of 209863 (1.44%) `NO_AD` samples being predicted as positive. From this model behavior, we conclude that a low TPR and therefore a low recall score is the reason for the model's poor performance.

When examining the model's prediction scores in Figure 8.2, we see that most values fall in the range of 0.05 to 0.13, with a mean value of 0.15. Therefore, we argue that in this case, false negatives are not a case of model indecisiveness, as the model seems to be certain about the samples being of class `NO_AD`. In the next section, we will examine samples of false negatives in more details.

8.1.2 Sample Investigation

To reiterate, the video podcast dataset uses community-sourced advertisement annotations found in the SponsorBlock project (Ajay Ramachandran 2024). There is neither a barrier of entry for anyone to submit annotations, nor is there any instance ensuring high annotation quality. It is therefore worth to investigate whether bad annotation quality could be the reason for low classification performance. In an effort to investigate data annotation quality, we manually examine 100 randomly chosen instances (meaning individual sentences) of false negatives, assessing the annotation quality. The result is presented in Table 8.1, the raw data is listed under Appendix B.

Metric	Yes	No	Total	Percentage
Transcription accurate	90	10	100	0.90%
Advertisement present	78	22	100	0.78%

Table 8.1: Investigating 100 randomly chose false negative for transcription accuracy and label correctness.

First, we observe that in 22 out of 100 cases, the video did not contain an advertisement in the specified timeframe, meaning a sample that is a false-negative under our evaluation data is actually a true-negative. In some instances, the ad had just ended, meaning the annotation's timecodes are inaccurate. In other instances, we found no ad in the vicinity of the timecodes. With 78% of valid annotations in our samples, the video podcasts dataset seems to contain annotations of suboptimal quality.

As the literature shows, NNs are adversely impacted by noise in training data: Rolnick et al. (2017) argue that depending on the dataset and the model architecture, having a clean label to noisy label ratio of 1:1 can cause a decrease in prediction accuracy of 0.1. While they show that a convolutional neural network with four hidden layers is robust to massive noise when trained on the MNIST dataset (Deng 2012), the model architecture and the dataset characteristics are hardly transferrable to this thesis. Kaplan, Handelman, and Handelman (2021) come to a similar conclusion, having trained a convolutional model on two image datasets. They find that corrupted labels decrease classification accuracy, depending on corruption percentage and class type.

Even though we conclude from the literature that noise in training data for simpler neural networks can reduce accuracy, we would expect the decrease to be present for both text embeddings and audio embeddings. But as Table 7.2 shows, model performance is particularly bad for audio embeddings, while text embeddings show higher performance (albeit worse compared to models trained on audio podcasts).

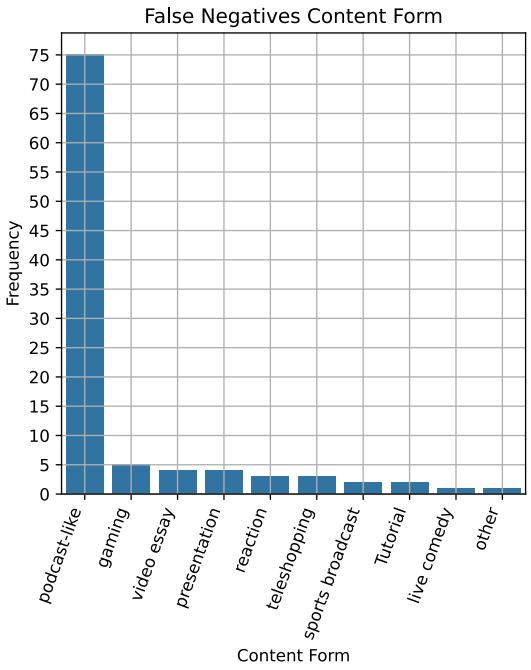


Figure 8.3: Content forms of false negatives. Categories are self-defined.

We have also investigated transcription accuracy. We deemed the transcription accurate if the words and punctuations at least broadly matched what was actually said. In 90 of the 100 cases, we find the transcription to be accurate.

Another reason for poor classification performance could be the fact that the chosen YouTube videos were too

podcast-unlike compared to episodes in the audio podcast dataset. But as the data in Figure 8.3 suggests, this is not the case, as 75 of the 100 samples were found to be podcast-like videos.

Given the above results, we conclude from the investigation of 100 randomly sampled false negatives that while we see some deficiencies in data annotation, especially in label noise, these findings do not explain the stark performance difference in model performance between text and audio embeddings. In the next section, we will examine the video podcast dataset on its audio characteristics.

8.1.3 Audio Data Characteristics

As we've seen in Table 7.1, the same model architecture exhibits vastly different classification performance when trained on audio embeddings compared to using text embeddings taken from the same audio snippets. This indicates that text embeddings from different classes are more easily distinguishable than audio embeddings. Both text and audio embeddings are of high dimensionality, with embeddings vectors of length 768, and thus difficult to visualize. Using the t-distributed stochastic neighbor embedding (t-SNE) dimension reduction technique (Maaten and Hinton 2008), we assign each embedding a location in a two-dimensional space. Although the technique is non-deterministic and the resulting visualization is highly dependent on the choice of parameters, it gives us a glimpse into the embedding space.

Before generating the visualizations, we preprocess our datasets. We combine audio and video podcasts and group the unified dataset by class and source, thus obtaining four groups. From each group, we sample 500 entries at random. Our visualizations therefore have 2000 data points. In each visualization, the colors correspond to the source, while marker type corresponds to classes. We visualize audio and text embeddings separately.

Figure 8.4 holds the t-SNE visualization for text embeddings. Here, we see that data points of both sources overlap, indicating no present domain shift. While we intuitively assumed classes to be more separated, this is not the case. Figure 8.5 on the other hand visualizes audio embeddings. We show t-SNE for video podcasts in 16 kHz (Figure 8.5a) and in 48 kHz (Figure 8.5b). Interestingly, we observe clustered audio data similar to the t-SNE of text embeddings when using video podcast data in 48 kHz. Using video podcast data in 16 kHz, we see that embeddings of audio and video podcasts are separated, indicating are clear distinction. In the next paragraph, we examine if and how fidelity of input data impacts model performances.

On a technical level, one clear difference in our dataset is that audio podcasts are stored in 44.1 kHz stereo, while video podcasts are stored in 16 kHz mono. While this might seem relevant at first glance, LanguageBind internally resamples all audio data to 16 kHz. Additionally, we have rerun some experiments with video podcast data in 16 kHz mono, 16 kHz stereo and 48 kHz stereo (Table 8.2).

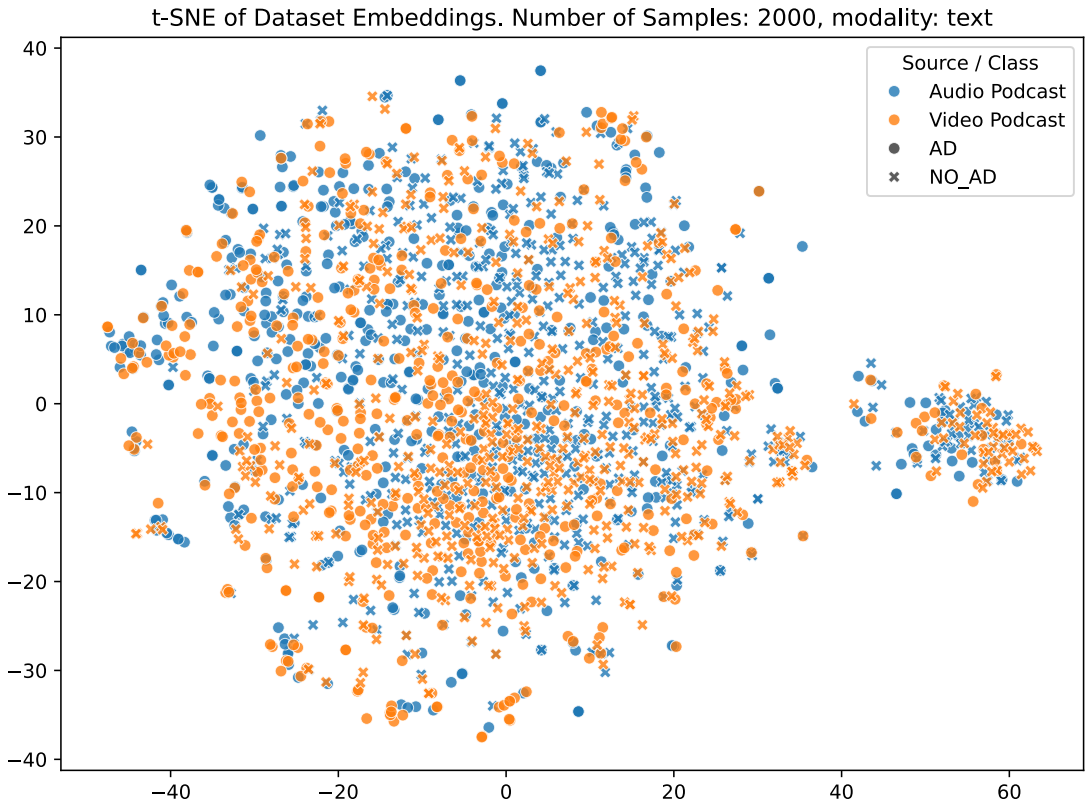
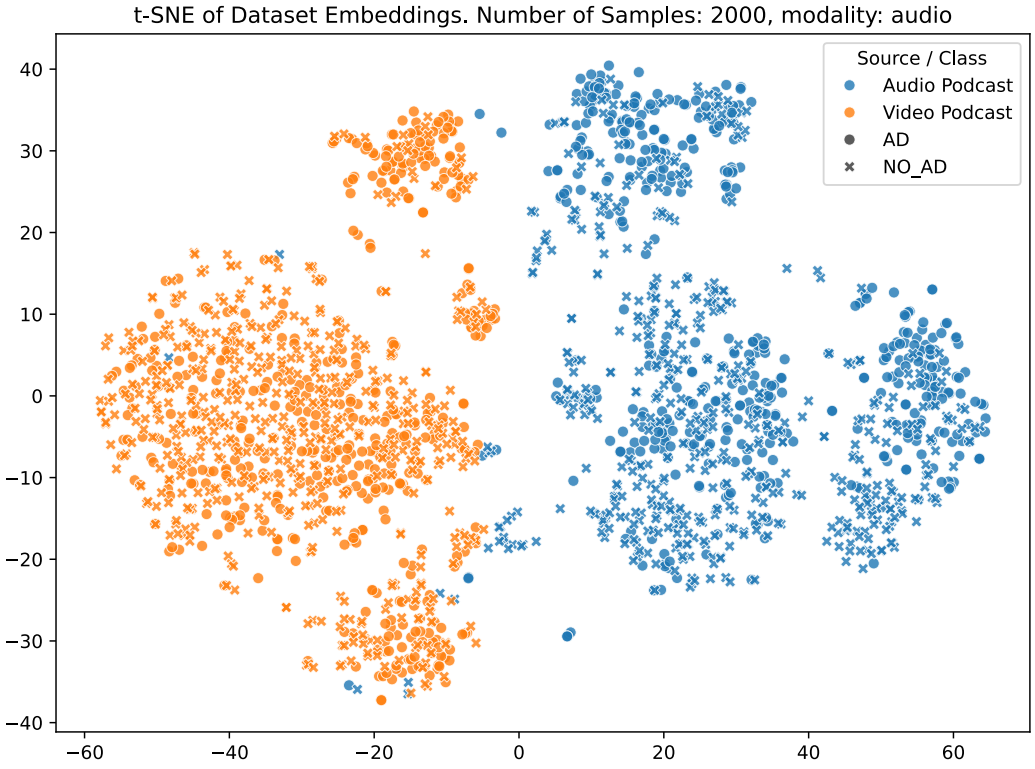


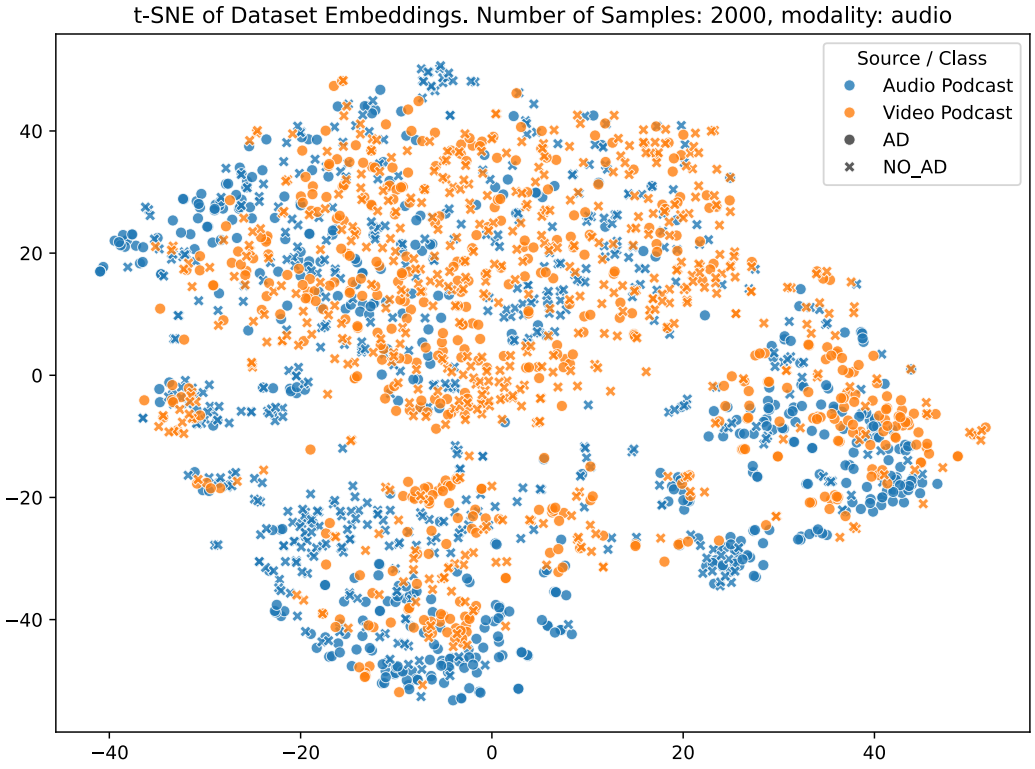
Figure 8.4: t-SNE of training dataset. 500 samples chosen at random per class and source combination.

In terms of F_1 and $F_{0.5}$, using stereo input data of 48 kHz seems to produce the best results when evaluating on same-source data. Although F_1 scores are doubled going from mono to stereo data and doubled again going from 16 kHz to 48 kHz, the gain in absolute numbers is only from 0.07 to 0.21, which is far below expectations. Additionally, we are especially interested in how models trained on video podcasts perform when evaluated on audio podcasts, as we are trying to use efficient to source video podcast data as training material for audio podcast inference. Here, we're also unable to improve from previous subpar results. Going from mono to stereo data, we observe a performance decrease. Using high-fidelity 48 kHz data drastically increases F_1 and $F_{0.5}$ to 0.24 and 0.33, but absolute numbers are still behind expectations.

In short, while using higher fidelity and stereo channels in input data improves model performance, the overall performance trend is unchanged, with generally low transferability of models across domains. Changes in fidelity resulting in different evaluation results is unexpected, as LanguageBind's internal resampling transforms all input data to 16 kHz low fidelity. The expectation here is that performance is unchanged regardless of input fidelity. Noticeable differences in t-SNE clustering in Figure 8.5 further hint at differences in audio embeddings between fidelity settings.



(a) Video podcast dataset in 16 kHz stereo.



(b) Video podcast dataset in 48 kHz stereo.

Figure 8.5: t-SNE of training dataset. 500 samples chosen at random per class and source combination. a) video podcast in 16 kHz low fidelity, b) video podcasts in 48 kHz high fidelity.

Train	Channels	Fidelity	Samples	Eval	P	R	F_1	$F_{0.5}$
Video Podcast	Mono	16 kHz	180,005	Video Podcast	0.47	0.04	0.07	0.14
				Audio Podcast	0.59	0.05	0.10	0.20
	Stereo	16 kHz	67,438	Video Podcast	0.54	0.09	0.16	0.28
				Audio Podcast	0.66	0.04	0.07	0.15
	Stereo	48 kHz	47,866	Video Podcast	0.49	0.14	0.21	0.32
				Audio Podcast	0.43	0.17	0.24	0.33

Table 8.2: Model performance when training on audio embeddings of video podcast dataset with different channels and fidelities. Models are evaluated on same-source data and audio podcast data.

8.1.4 Conclusion

After having investigated false negative samples on content form, annotation quality and label correctness, as well as experimenting with changes in fidelity and mono vs. stereo channels and plotting dimension reductions, we cannot give a definitive, single answer as to why our models perform worse on video podcasts compared to audio podcast.

It is likely that performance is low due to a combination of factors: Imperfect annotation quality, high advertisement variance due to only self-voiced advertisements being present in YouTube videos, presence of non-podcast-like content in the dataset as well as missing second audio channel in stored YouTube videos. We can assume YouTube to process videos and audios post-upload, possible smoothing audio characteristic differences between advertisements and content. As we have seen in Section 7.4, self-voiced advertisements are harder to classify than inserted advertisements, as they are generally more diverse and produced in-house, using the same processing as the video's content. This could also play a role here, as YouTube only permit self-voiced advertisements in its videos (see Listing 2.1).

8.2 Results Interpretation

In this section, we will provide hypotheses and possible explanations for the observed results. For clarity, the section is structured according to the experiments.

8.2.1 Modality Information Gain

In the first experiment in Section 7.1, we've examined how using embeddings from either text, audio or the combination of both impacted the performance of our classi-

fication models. We've observed that the audio features perform particularly well for audio podcasts. We attribute this the presence of inserted advertisements in the audio podcast dataset, as these types of advertisements are produced at the advertisers end, thus having different audio characteristics compared to regular content. This hypothesis is supported by visual inspection performed while manually annotating the audio podcast dataset (see Section 5.1.1 and Figure 5.1).

For video podcast, we observed that audio features perform worse, to the point that the combination of modalities perform worse than text embeddings alone. We have investigated possible reasons in the previous section, concluding that various factors contribute to the issue, namely mislabeled data, self-voiced advertisement and possible technical differences in terms of audio fidelity and channel count.

8.2.2 Out-of-domain Training data

We initially motivated using podcast-like videos sourced from YouTube as OOD data to train models for in-domain audio podcast inference in an effort to circumvent the tedious annotation process of podcasts, as there is no publicly available dataset of podcasts with advertisement annotations known to us. As we've seen in Section 7.2, learned feature characteristics learnt from OOD data do no transfer over well to in-domain data. This again is due to bad performance of the audio features, as text features at least achieve a F_1 score of 0.46 (see Table 7.3). We again attribute this to the difference in advertisement between domains, with video podcasts lacking the important inserted advertisement type.

8.2.3 Superlocal Context

To bypass the segmentation decision, which is a problem worth exploring on its own (Hearst 1997; Koshorek et al. 2018; Ghazimatin et al. 2024), we've decided on sentence-level segmentation treating each sentence uttered in a podcast episode as a single sample. The classifiers are then trained on sentence level, thus working with limited context. In order to introduce more context into the model architecture, we've created the *superlocal* architecture variant, which receives intermediate embeddings from a *local* type base model and concatenates up to context_length representations together to perform classification. In Section 7.3, we've shown how these changes to the model architecture resulted in a negligible performance gain. The reason for that is likely twofold: Datasets limitations and inner workings of the training loop.

First, we've shown in Section 6.11 that the audio podcast dataset's transcription contains hallucinations in the form of short sentences being repeated over and over. Additionally, we've encountered issues in LanguageBind' data preprocessing pipeline when feeding audio data that is too short and therefore require each sample to be at least 0.5s long. By excluding hallucinations and short sentences, we introduce gaps in the dataset. Thus, two samples chosen in order might not appear directly after each other on the original data. Testing how many samples are affected by this might be a good first step in a future work.

Id	Samples	Context Size	Training Steps	Precision	Recall	F_1	$F_{0.5}$
K	67,264	32	2102	0.77	0.73	0.75	0.76
O	67,294	2	33647	0.82	0.85	0.83	0.82
P	4,204	2	2102	0.85	0.68	0.76	0.81

Table 8.3: Excerpt of Table 7.4.

Second, the windowing logic in the training loop does only full window-length steps, meaning given a content_length of 2, samples [1, 2, 4, 5, 6] will be loaded for training in pairs of (1, 2), (3, 4), (5, 6). Therefore, the number of steps in the training loop directly correlates with the size of the context. Alternatively, using overlapping windows, samples could be processed as (1, 2), (2, 3), (3, 4) and so on. This way, context length is decoupled from training steps.

To confirm that lower training steps lead to worse performance, we've trained a model using a context size of 2 (model P) but reduced its training samples to match the number of samples (and thus, steps) a model with context size 32 would have (model K, Table 8.3). We see that the reduction in training samples for model P leads to a performance decrease compared to model O, thereby confirming our hypothesis that models with higher context length might not only suffer from possibly too large contexts, but also from reduced training data. It'd be interesting to see how using a bigger transcription model in conjunction with overlapping window logic in training affects superlocal model performance, as it could be assumed that the model would generally benefit from capturing more context.

8.2.4 Advertisement Type Classification Performance

In our last experiment, we focused on performance differences between inserted and self-voiced advertisements. We created two subsets of our audio podcast dataset, each containing only one of the two ad types. These subsets are almost equal in length, as both advertisement types are equally frequent in the original dataset (see Figure 6.9). Having trained and evaluated models on the subsets (Table 7.5) on text, audio and combined embeddings, we conclude that inserted advertisements yield the best classification performance in F_1 and $F_{0.5}$ scores, especially on audio embeddings.

Again, we attribute this to the fact that inserted advertisements are produced at the advertiser's end and thus differ from content in audio characteristics. In addition, we've observed instances of inserted advertisements being present over multiple episodes in our dataset, even across podcast shows. Self-voiced advertisements seem to be reused less often. From this observation, we conclude that inserted advertisements can be learned as a repeating pattern, which is less the case for self-voiced advertisements. Examining individual advertisement on their content and how often it's repeated across episodes and shows is another well suited task for a future work.

8.3 Limitations

While this thesis presents a systematic approach to identifying advertisements in podcasts using machine learning, several limitations must be acknowledged. These limitations stem from both practical constraints and methodological choices made throughout the process. It is crucial to consider these limitations when drawing the final conclusion.

8.3.1 Data Limitations

In the context of this thesis, data limitations arise from multiple sources, each having an impact on the data's quality and quantity.

First off, to our knowledge, a labeled dataset of advertisements in podcasts is not publicly available. In our literature review, we've encountered the once available dataset of 100,000 annotated podcast episodes published by Spotify (Clifton et al. 2020), but public access has since been discontinued (Spotify 2023a). To overcome this issue, we created our own dataset, consisting of 160 manually annotated audio podcasts and 1000 community annotated podcast-like YouTube videos.

Starting with the audio podcast dataset, we state that most episode were published and scraped in October 2024 (Figure 6.1). Since advertising agencies often run advertisements in multiple podcasts simultaneously, we noticed several advertisements being shared over episodes and shows, decreasing the overall advertisement diversity, running the risk of overfitting. As our possibly overfitted models are evaluated using the testing part of the same dataset, we cannot definitely say whether overfitting is present.

To account for overfitting, the corpus should consist of more episodes published and sourced over a longer period of time. Simply sourcing more episodes all at once cannot be the solution, due to dynamic ad insertion (DAI). Using DAI, the user is served advertisements based on region, time and ad profile, making the dataset creation non-deterministic. We've experienced this first hand when downloaded English podcast shows contained advertisements in German. Setting an American geolocation using a VPN not only resulted in English advertisement, but also in an overall higher advertisement frequency. For that reason, creating a dataset of podcasts released over a longer period of time (e.g. the last year) requires to periodically download recently released episodes, as downloading them all at once would result in shared advertisements due to DAI. While a dataset 160 episodes is a good start, the models trained in this thesis are likely to benefit from a larger, better balanced dataset.

The video podcast dataset sourced from YouTube and annotated by users in the SponsorBlock Project (Ajay Ramachandran 2024) comes with its own list of issues. As mentioned previously, the SponsorBlock annotations are of unknown quality, with no mechanisms such as inter-annotator agreement (IAA) in place to ensure annotation quality. We have discovered in our error analysis (Section 8.1) that for the chosen model, 20% of false negatives were mislabeled samples and thus in fact true negatives, thus confirming subpar annotation quality.

Then, not all sourced videos are often podcast-like in content (Figure 8.3), due to the broadly chosen heuristics (Table 5.1). While it would have been possible to pre-select known-good video podcasts to download, using heuristics for video discovery was motivated to create a broader, more diverse dataset.

During the error analysis in Section 8.1, we've encountered slight differences in model performance depending on mono / stereo and fidelity of stored data. Although LanguageBind internally resamples all incoming audio to 16 kHz, and mono data performed comparable to stereo data in selected experiments, the t-SNEs visualized in Figure 8.5 hint at embedding differences between high fidelity and low fidelity. It is therefore possible that inconsistencies in data formats could have caused the datasets to perform this differently, although multiple tests were performed to reject this hypothesis.

Class imbalance is present in both datasets, with an averaged 4% of all samples being advertisements (Table 6.1). This leads to models overfitting to the majority (non-ad) class. We have experimented with different class distributions by undersampling the majority class, without being able to drastically improve model performance, especially when trained on video podcasts. It is possible that using different class splits or different techniques to overcome the imbalance data problem would lead to better results, such as using a weighted loss function as discussed in Section 2.4.2.

8.3.2 Model Limitations

Next, we'll go over limitations in models used and trained throughout the thesis.

To begin, we have used the Whisper variant whisper-timestamped (linto-ai 2023) to transcribe podcast episodes and YouTube videos. The quality of the transcriptions is critical, as its accuracy affects performance of downstream tasks. While measuring exact word error rates is beyond the scope of this thesis, we've shown in Section 6.11 that the deployed base whisper model is prone to hallucinations, repeating short sentences. In raw numbers, about half of all our samples are marked as hallucinations (Figure 6.14a), due to the fact that once hallucination is present, the number of repetitions is generally high. Looking at the total sample duration of clean sentences compared to hallucinations, we see that hallucinations only account for 10% of the total sample duration, rendering them less severe. Even if present hallucinations are less critical in terms of time, a cleaner dataset transcribed using a more accurate (but also more resource heavy) whisper model might increase classification performance, especially when joining samples for superlocal classifier training.

As the data preprocessing pipeline in LanguageBind throws an error for short samples when decomposing the audio into its frequency components using bandpass filters, the dataset was stripped of samples shorter than 0.5s. Ideally, all parts of an episode are part of the training dataset. In a future work, one could combine short samples with its neighbors until a threshold is reached to ensure a minimal duration over all samples.

One major limitation for all models in the thesis is the restriction to the English language. Both source material of podcasts and YouTube videos were limited to English to decrease complexity in transcription, preprocessing and classification pipelines. A future work could investigate how the chosen model architecture performs over multiple languages, as form and content of advertisements is likely to differ from one language and culture to another.

Speaking of model architecture, faults in model architecture design cannot be ruled out. For example, the number of hidden layers and their hidden neurons in both local and superlocal classifiers is chosen according to general advice (Heaton 2008), but as there is no definitive answer as to which layer design works best for a task, a different setup could have result in better results. It is possible that adding context in superlocal classifiers would have worked better if technique decisions were made differently.

8.3.3 Conclusion

Drawing a conclusion about the laid out limitations in the previous sections, we have to acknowledge that the transferability of the results presented in this thesis is fairly limited. This is due to severe limitation in both data and model design.

In terms of data limitations, the well performing audio podcast dataset is limited in the sense that most episodes were published and scraped over the course of a single month. All podcasts are English-speaking and popular in the United States, further decreasing generalizability. While the video podcast dataset is larger, more diverse in time of upload and has a higher number of unique shows, it's performance in classification tasks on audio embeddings is low. Models trained in the thesis were also never evaluated on datasets from other domains containing advertisements, such as TV and radio broadcasts. Model limitations include hallucinations, difficulties in sample preprocessing, English language focus and unproven architecture design regarding e.g. layer and neuron count.

Although presented limitations are non-negligible, we were able to show that using the created dataset and chosen model design and decision thresholds, we were able to achieve F_1 and $F_{0.5}$ scores of 0.83 and 0.87 classifying advertisement in audio podcasts.

9 Conclusion

In the final chapter of the thesis, we will recap our main findings, reference the research questions and sum up the main learnings. We'll end by providing an outlook for possible future research directions.

9.1 Summary

In this thesis, we created a two-part dataset of advertisement in podcasts. Sourcing over 1150 podcast episodes from RSS feeds and from YouTube videos, we aimed to build a large, diverse dataset of podcast-like audio content for advertisement classification. The audio podcast part was annotated manually, thus significantly smaller (160 episodes). YouTube videos were matched with existing ad annotation data from the SponsorBlock project. To our knowledge, it is the first dataset of podcasts with advertisement annotations.

Exploring the dataset, we differentiate two main types of advertisements: Inserted and self-voiced advertisements. The former are produced by the advertiser and usually have an industry standard length of 30 seconds, self-voiced advertisements are read by the podcast's host and their team, being more varied in duration, style and content. This directly answers

RQ1: How can advertisements in podcasts be characterized and what differentiates advertisements in audio podcasts from advertisements in video podcasts?

While traditional audio podcasts from RSS feeds feature both advertisement types equally, YouTube does not allow inserted advertisements on their platform. Generally, we have seen major differences in advertisement characteristics across sources, like audio podcasts often feature multiple single ads in a row, something that's less common in video podcasts.

As part of the preprocessing, we transcribe podcast episodes using the base model of the Whisper model family. Using its timestamp-enhanced fork whisper-timestamped,

we align transcription and advertisement annotation, splitting the dataset into sentences of `AD` and `NO_AD` samples.

We introduced two types of classifiers. The *local* classifier uses the Transformer-based LanguageBind model to compute multimodal embeddings from text and audio data of samples on sentence-level. Training on audio podcasts, we are able to outperform our baseline naive Bayes and SVM classifiers trained on long-word sequences. Then, we examined how modality information affected model performance in

RQ2: How does model performance vary when using audio, text, or multimodal training data?

We found that while both the text and audio modality produced suitable results with high F_1 scores, concatenating embeddings from both modalities boosts the model performance further. We state that for audio podcasts, combining modalities is beneficial for classification performance.

The same cannot be said for models trained on video podcasts. In an effort to skip the manual annotation process, we've tried to use publicly available annotations of OOD data in the form of YouTube videos in

RQ3: Can training data from video podcasts be used to detect advertisement in audio podcasts?

Overall, model performance when trained on OOD YouTube data is far below that of in-domain models, evaluated on either dataset. We attribute this to multiple factors. Different advertisement characteristics across datasets in terms of class distribution and advertising type, fluctuating annotation quality in the video podcast dataset and lastly possible post-upload processing performed by YouTube. Although text embeddings perform somewhat tolerable, audio embeddings performance is particularly low. Trying to transfer learned characteristics from video podcasts by evaluating on audio podcast did also not produce suitable results. We therefore conclude that at least for the presented dataset and model architecture, manually annotation of in-domain audio podcasts yields far higher results than using OOD data.

We've also tried augmenting the model architecture with more context in the form of the *superlocal* architecture variant, effectively joining intermediate input representations from a *local* classifier to classifier multiple samples at once. Models performed on par or slightly above *local* classifiers, possibly as increase context windows decrease total number of steps in training.

We conclude by stating that modern Transformer-based approaches are perfectly capable of identifying advertisements in podcasts. In a real-world scenario, our best performing model is capable of detecting 60% of all advertising sentences while only producing a single false positive in 30 minutes.

9.2 Outlook

Due to limited time and resources, interesting methods and aspects of advertisement identification in podcasts were left untouched. As seen in related works, content detection on audio data is a vast field in which a problem can be approach from multiple angles.

Beginning with segmentation, we have decided to use sentence-level segmentation in this thesis to keep segmentation effort to a manageable minimum. In turn, our *local* classifier architecture relies solely on the context of a single sentence. We've tried to introduce more context by processing multiple sentences at once in our *superlocal* classifier architecture, but where unable to significantly improve on single-sentence results. Another approach could have been employing chapterization as part of the preprocessing as presented by Glavăs and Somasundaran (2020) and Ghazimatin et al. (2024), as topic differences can be expected between classes.

Due to details in the training loop, training steps of *Superlocal* classifiers decreased with increased context sizes, possibly resulting in models with longer contexts showing particularly low performance. To overcome this, adequate training steps could be ensured by either using overlapping windows as discussed in Section 8.2.3 or by using a larger training dataset.

During training of either classifier architecture, we've only adjusted the weights of the classification layers via backpropagation. Also adjusting LanguageBind's first few layers could result in input embeddings that are better suited for the classification task at hand.

In terms of data, the audio podcast dataset is limited in number of episodes and time distribution, resulting of shared advertisements over episodes and shows. Sourcing more shows over a longer period of time could lead to higher diversity in advertisements. We've also not looked at how approaches presented in this thesis can be transferred to non-English content or other domains like radio broadcasts.

With the emergence of the generative pre-trained Transformer (GPT) architecture, LLMs became freely available and performant, even on consumer-grade hardware. Álvarez et al. (2024) use GPT-4 for results validation, concluding that while achieving good results, "small models trained on a specific task can beat state-of-the-art generalist models" (Álvarez et al. 2024). In our case, we've tried using the Llama (Touvron et al. 2023) 3.1 8 billion parameter LLM for advertisement classification with mixed results. While requiring little to no setup and corpus preprocessing, inference on available hardware is slow and requires proper prompting. Additionally, we ran into situations where the model refused to process a given sample, as it violated safety guard rails established in training, for example when the sample included sexually explicit content or violence. Here, using a newer model from the Llama 4 model family with a higher parameter count could improve classification and segmentation performance in few-shot prompting, although computing requirements will still exceed those of smaller, single-task models like the ones presented in this thesis.

Bibliography - Science

Brinson, Nancy H., and Laura L. Lemon and. 2023. "Investigating the Effects of Host Trust, Credibility, And Authenticity in Podcast Advertising". *Journal of Marketing Communications* 29 (6): 558–76. https://doi.org/10.1080/13527266.2022.2054017

Bulakh, Tetiana, Olena Kulykova, Kateryna Martiukhyna, Olena Karpenko, and Iryna Putsiata. 2023. "Main Types of Podcast Advertising: Foreign and Ukrainian Experience". *Amazonia Investiga* 12 (67): 162–72

Cardinal, Patrick, Vishwa Gupta, and Gilles Boulianne. 2010. "Content-Based Advertisement Detection". In *11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010, Chiba, Japan, September 26-30, 2010*, 2214–17. Chiba, Japan. https://doi.org/10.21437/INTERSPEECH.2010-609

Chen, Chao, Andy Liaw, and Leo Breiman. 2004. "Using Random Forest to Learn Imbalanced Data". Vol. 110. University of California, Berkeley

Chung, Joon Son, Arsha Nagrani, and Andrew Zisserman. 2018. "Voxceleb2: Deep Speaker Recognition". In *Interspeech 2018*. Hyderabad, India. https://doi.org/10.21437/interspeech.2018-1929

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". In *Proceedings of the 28th International Conference on Computational Linguistics*, 5903–17. Barcelona, Spain (Online). https://doi.org/10.18653/v1/2020.coling-main.519

Covell, Michele, Shumeet Baluja, and Michael Fink. 2006. "Advertisement Detection and Replacement Using Acoustic and Visual Repetition". In *2006 IEEE Workshop on Multimedia Signal Processing*, 0:461–66. Victoria, Canada. https://doi.org/10.1109/MMSP.2006.285351

Dahlin, Christian Ryddheim. 2024. "Adblock for Podcasts". The Norwegian University of Science and Technology, Norway. https://hdl.handle.net/11250/3137499

Deng, Li. 2012. "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]". *IEEE Signal Processing Magazine* 29 (6): 141–42. https://doi.org/10.1109/MSP.2012.2211477

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding". In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–86. Minnesota, USA

Drumond, A, AL Rodrigues, JFCL Costa, FG Niquini, and MG Lemos. 2024. "Models for Analysing the Economic Impact of Ore Sorting, Using ROC Curves". *Journal of the Southern African Institute of Mining and Metallurgy* 124 (7): 397–406

Estabrooks, Andrew, Taeho Jo, and Nathalie Japkowicz. 2004. "A Multiple Resampling Method for Learning from Imbalanced Data Sets". *Computational Intelligence* 20 (1): 18–36. https://doi.org/https://doi.org/10.1111/j.0824-7935.2004.t01-1-00228.x

Feldstein Jacobs, Adam. 2022. "Automatic Podcast Chapter Segmentation: A Framework for Implementing and Evaluating Chapter Boundary Models for Transcribed Audio Documents", no. 2022:559 , 63

Gage, Philip. 1994. "A New Algorithm for Data Compression". *C Users J.* 12 (2): 23–38

Gales, Mark, and Steve Young. 2008. Vol. 0. https://doi.org/10.1561/2000000004

Ghazimatin, Azin, Ekaterina Garmash, Gustavo Penha, Kristen Sheets, Martin Achenbach, Oguz Semerci, Remi Galvez, et al. 2024. "PODTILE: Facilitating Podcast Episode Browsing with Auto-Generated Chapters". In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 4487–95. CIKM '24. Idaho, USA. https://doi.org/10.1145/3627673.3680081

Glavăs, Goran, and Swapna Somasundaran. 2020. "Two-Level Transformer and Auxiliary Coherence Modeling for Improved Text Segmentation". In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence : New York, NY, February 7-12, 2020*, 5:7797–7804. New York, USA. https://doi.org/10.1609/aaai.v34i05.6284

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press

Hatzel, Hans Ole. 2020. "Using Neural Language Models to Detect Spoilers". University of Hamburg, Germany

He, Haibo, and Edwardo A. Garcia. 2009. "Learning from Imbalanced Data". *IEEE Transactions on Knowledge and Data Engineering* 21 (9): 1263–84. https://doi.org/10.1109/TKDE.2008.239

Hearst, Marti A. 1997. "Texttiling: Segmenting Text into Multi-Paragraph Subtopic Passages". *Comput. Linguist.* 23 (1): 33–64

Heaton, Jeff. 2008. *Introduction to Neural Networks for Java, 2nd Edition*. 2nd ed. Heaton Research, Inc.

Hirschberg, Julia, and Christopher D. Manning. 2015. "Advances in Natural Language Processing". *Science* 349 (6245): 261–66. https://doi.org/10.1126/science.aaa8685

Huang, Chen, Yining Li, Chen Change Loy, and Xiaoou Tang. 2016. "Learning Deep Representation for Imbalanced Classification". In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 0:5375–84. Las Vegas, USA. https://doi.org/10.1109/CVPR.2016.580

Huang, Lei, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, et al. 2025. "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, And Open Questions". *ACM Trans. Inf. Syst.* 43 (2). https://doi.org/10.1145/3703155

Hutchins, W. John. 2004. "The Georgetown-IBM Experiment Demonstrated in January 1954". In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas: Technical Papers*, 102–14. Washington, USA: Springer. https://aclanthology.org/2004.amta-papers.12/

Iqbal, Umar, Zubair Shafiq, and Zhiyun Qian. 2017. "The Ad Wars: Retrospective Measurement and Analysis of Anti-Adblock Filter Lists". In *Proceedings of the 2017 Internet Measurement Conference*, 171–83. IMC '17. London, United Kingdom. https://doi.org/10.1145/3131365.3131387

Japkowicz, Nathalie, and Shaju Stephen. 2002. "The Class Imbalance Problem: A Systematic Study". *Intelligent Data Analysis* 6 (5): 429–49

Jin, Liuyi, Tian Liu, Amran Haroon, Radu Stoleru, Michael Middleton, Ziwei Zhu, and Theodora Chaspari. 2023. "Emsassist: An End-to-End Mobile Voice Assistant at the Edge for Emergency Medical Services". In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*, 275–88. Mobisys '23. Helsinki, Finland. https://doi.org/10.1145/3581791.3596853

Johri, Prashant, Sunil K. Khatri, Ahmad T. Al-Taani, Munish Sabharwal, Shakhzod Suvanov, and Avneesh Kumar. 2021. "Natural Language Processing: History, Evolution, Application, And Future Work". In *Proceedings of 3rd International Conference on Computing Informatics and Networks*, 365–75. Athens, Greece (Online)

Juang, B. H., and L. R. Rabiner and. 1991. "Hidden Markov Models for Speech Recognition". *Technometrics* 33 (3): 251–72. https://doi.org/10.1080/00401706.1991.10484833

Kaplan, Shimon, Doron Handelman, and Amir Handelman. 2021. "Sensitivity of Neural Networks to Corruption of Image Classification". *AI and Ethics*, 1–10. https://doi.org/10.1007/s43681-021-00049-0

Koshorek, Omri, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Berant. 2018. "Text Segmentation as a Supervised Learning Task", June, 469–73. https://doi.org/10.18653/v1/N18-2075

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep Learning". *Nature* 521 (7553): 436–44. https://doi.org/10.1038/nature14539

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. "Roberta: A Robustly Optimized BERT Pretraining Approach". https://arxiv.org/abs/1907.11692

Logan, Beth, and others. 2000. "Mel Frequency Cepstral Coefficients for Music Modeling.". In *ISMIR 2000, 1st International Symposium on Music Information Retrieval*, 270:11. Massachusetts, USA

Lu, Lie, Hong-Jiang Zhang, and Hao Jiang. 2002. "Content Analysis for Audio Classification and Segmentation". *IEEE Transactions on Speech and Audio Processing* 10 (7): 504–16. https://doi.org/10.1109/TSA.2002.804546

Maaten, Laurens van der, and Geoffrey Hinton. 2008. "Visualizing Data Using t-Sne". *Journal of Machine Learning Research* 9 (86): 2579–2605. https://jmlr.org/papers/v9/vandermaaten08a.html

Marlow, Seán, David A Sadlier, Karen McGeough, Noel E O'Connor, and Noel Murphy. 2001. "Audio and Video Processing for Automatic TV Advertisement Detection". In *ISSC 2001 - Irish Signals and Systems Conference*. Maynooth, Ireland

Meenaghan, Tony. 2001. "Sponsorship and Advertising: A Comparison of Consumer Perceptions". *Psychology & Marketing* 18 (2): 191–215. https://doi.org/https://doi.org/10.1002/1520-6793(200102)18:2<191::AID-MAR1005>3.0.CO;2-C

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. "Efficient Estimation of Word Representations in Vector Space". https://arxiv.org/abs/1301.3781

Moe, Margaret. 2023. "Podvertising: Podcast Listeners' Advertising Attitudes, Consumer Actions and Preference for Host-Read Ads". *Journal of Economics and Behavioral Studies* 14 (4(J): 50–66. https://doi.org/10.22610/jebs.v14i4(J).3278

Nguyen, Minh Nhut, Qi Tian, and Ping Xue. 2010. "Efficient Advertisement Discovery for Audio Podcast Content Using Candidate Segmentation". *EURASIP Journal on Audio, Speech, And Music Processing* 2010:1–12

Rabiner, L.R. 1989. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". *Proceedings of the IEEE* 77 (2): 257–86. https://doi.org/10.1109/5.18626

Radford, Alec, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever. 2023. "Robust Speech Recognition via Large-Scale Weak Supervision". In *Proceedings of the 40th International Conference on Machine Learning*, 202:28492–518. Proceedings of Machine Learning Research. Hawaii, USA. https://proceedings.mlr.press/v202/radford23a.html

Ramires, António, Diogo Cocharro, and Matthew EP Davies. 2018. "An Audio-Only Method for Advertisement Detection in Broadcast Television Content". https://arxiv.org/abs/1811.02411

Rawte, Vipula, Amit Sheth, and Amitava Das. 2023. "A Survey of Hallucination in Large Foundation Models". https://arxiv.org/abs/2309.05922

Reddy, Sravana, Mariya Lazarova, Yongze Yu, and Rosie Jones. 2021b. "Modeling Language Usage and Listener Engagement in Podcasts". In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 632–43. Online. https://doi.org/10.18653/v1/2021.acl-long.52

Reddy, Sravana, Yongze Yu, Aasish Pappu, Aswin Sivaraman, Rezvaneh Rezapour, and Rosie Jones. 2021a. "Detecting Extraneous Content in Podcasts". In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 1166–73. Online. https://doi.org/10.18653/v1/2021.eacl-main.99

Riedl, Martin, and Chris Biemann. 2012. "TopicTiling: A Text Segmentation Algorithm Based on LDA". In *Proceedings of ACL 2012 Student Research Workshop*, 37–42. Jeju Island, Korea. https://aclanthology.org/W12-3307/

Ritter, Eric A., and Chang Hoan Cho. 2009. "Effects of Ad Placement and Type on Consumer Responses to Podcast Ads". *Cyberpsychology and Behavior* 12 (5): 533–37. https://doi.org/10.1089/cpb.2009.0074

Rolnick, David, Andreas Veit, Serge J. Belongie, and Nir Shavit. 2017. "Deep Learning Is Robust to Massive Label Noise". *Corr*. https://arxiv.org/abs/1705.10694

Sammut, Claude, and Geoffrey I Webb. 2011. *Encyclopedia of Machine Learning*. Springer Science & Business Media. https://doi.org/10.1007/978-0-387-30164-8

Schmidt, Robin M. 2019. "Recurrent Neural Networks (Rnns): A Gentle Introduction and Overview". https://arxiv.org/abs/1912.05911

Schuster, Mike, and Kaisuke Nakajima. 2012. “Japanese and Korean Voice Search”. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 0:5149–52. Kyoto, Japan. https://doi.org/10.1109/ICASSP.2012.6289079

Sutor, Peter, Yiannis Aloimonos, Cornelia Fermuller, and Douglas Summers-Stay. 2019. “Metaconcepts: Isolating Context in Word Embeddings”. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, 0:544–49. California, USA. https://doi.org/10.1109/MIPR.2019.00110

Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, et al. 2023. “Llama: Open and Efficient Foundation Language Models”. https://arxiv.org/abs/2302.13971

Vaiani, Lorenzo, Moreno La Quatra, Luca Cagliero, and Paolo Garza. 2022. “Leveraging Multimodal Content for Podcast Summarization”. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, 863–70. SAC '22. Online. https://doi.org/10.1145/3477314.3507106

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. “Attention Is All You Need”. In *Advances in Neural Information Processing Systems*, edited by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 30:. Curran Associates, Inc.. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Vimal, B., Muthyam Surya, Darshan, V.S. Sridhar, and Asha Ashok. 2021. “MFCC Based Audio Classification Using Machine Learning”. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 0:1–4. Kharagpur, India. https://doi.org/10.1109/ICCCNT51525.2021.9579881

Werbos, Paul J. 1988. “Generalization of Backpropagation with Application to a Recurrent Gas Market Model”. *Neural Networks* 1 (4): 339–56. https://doi.org/https://doi.org/10.1016/0893-6080(88)90007-X

Yacim, Joseph Awoamim, and Douw Gert Brand Boshoff. 2018. “Impact of Artificial Neural Networks Training Algorithms on Accurate Prediction of Property Values”. *Journal of Real Estate Research* 40 (3): 375–418. https://doi.org/10.1080/10835547.2018.12091505

Yu, Shi, Yuxin Chen, and Hussain Zaidi. 2021. “AVA: A Financial Service Chatbot Based on Deep Bidirectional Transformers”. *Frontiers in Applied Mathematics and Statistics*. https://doi.org/10.3389/fams.2021.604842

Zaman, Khalid, Kai Li, Melike Sah, Cem Direkoglu, Shogo Okada, and Masashi Unoki. 2025. “Transformers and Audio Detection Tasks: An Overview”. *Digital Signal Processing* 158:104956. https://doi.org/https://doi.org/10.1016/j.dsp.2024.104956

Zhu, Bin, Bin Lin, Munan Ning, Yang Yan, Jiaxi Cui, HongFa Wang, Yatian Pang, et al. 2024. “Languagebind: Extending Video-Language Pretraining to n-Modality by Language-Based Semantic Alignment”. https://arxiv.org/abs/2310.01852

Zhu, Winstead, Md Iftekhar Tanveer, Yang Janet Liu, Seye Ojumu, and Rosie Jones. 2023. “Lightweight and Efficient Spoken Language Identification of Long-Form Audio”. In *Interspeech 2023*, 496–500. Dublin, Ireland. https://doi.org/10.21437/Interspeech.2023-304

Zibert, Janez, Bostjan Vesnicer, and France Mihelic. 2007. “Novel Approaches to Speech Detection in the Processing of Continuous Audio Streams”. In *Robust Speech*. Rijeka: IntechOpen. https://doi.org/10.5772/4741

Álvarez, Jorge, Juan Carlos Armenteros, Camilo Torrón, Miguel Ortega-Martín, Alfonso Ardoiz, Oscar Garcia, Ignacio Arranz, et al. 2024. “RADIA - Radio Advertisement Detection with Intelligent Analytics”. https://arxiv.org/abs/2403.03538

Bibliography - Websites

Acast. 2023. “The Benefits of Dynamic Ad Insertion in Podcast Advertising”. 2023. https://advertise.acast.com/news-and-insights/the-benefits-of-dynamic-ad-insertion-in-podcast-advertising

Ajay Ramachandran, Jeremy Plsek. 2024. “Sponsorblock”. 2024. https://sponsor.ajay.app/

American Marketing Association. 2025. “Advertising”. 2025. https://www.ama.org/topics/advertising/

Anu Adegbola. 2024. “Youtube Embeds Ads into Videos to Beat Ad Blockers”. 2024. https://searchengineland.com/youtube-embeds-ads-videos-ad-blockers-443187

Chartable.com. 2024. “Apple Podcast Charts”. 2024. https://chartable.com/charts/itunes/us

Coats, Cameron. 2024. “From Audio to Visual, Gen Z Drives the Youtube Podcast Boom”. 2024. https://radioink.com/2024/10/23/from-audio-to-visual-gen-z-drives-youtubes-podcast-boom/

Depoix, Jonas. 2024. “Pypi.org - Youtube Transcript Api”. 2024. https://pypi.org/project/youtube-transcript-api/

Dictionary.com. 2025. “Podcast”. 2025. https://www.dictionary.com/browse/podcast

Engle-Eshleman, Micah. 2025. “Adblock Podcast”. 2025. https://www.adblockpodcast.com/

Escandon, Rosa. 2024. “‘Video Podcasting’ Growing in Popularity”. 2024. https://www.forbes.com/sites/rosaescandon/2024/04/29/video-podcasting-growing-in-popularity/

Lee, Andrew. 2023. “Neuralblock”. 2023. https://github.com/andrewzlee/NeuralBlock

linto-ai. 2023. “Whisper-Timestamped”. 2023. https://github.com/linto-ai/whisper-timestamped

Mayer, Elizabeth, and Paul Riismandel. 2023. “Podcast Download - Fall 2023 Report”. 2023. https://www.westwoodone.com/blog/2023/12/04/youtubes-growth-as-a-podcast-power-player-revealed-in-cumulus-media-and-signal-hill-insights-podcast-download-fall-2023-report/

Mickle, Tripp. 2022. “Farewell to the Ipod”. 2022. https://www.nytimes.com/2022/05/10/technology/apple-ipod-phasing-out.html

OFM. 2024. “Radio Advertising and the 30-Second Rule”. 2024. https://www.ofm.co.za/article/press-office/335164/radio-advertising-and-the-30-second-rule

OpenAI. 2022. “Whisper”. 2022. https://github.com/openai/whisper

Pot, Justin. 2013. “The Evolution of the Podcast – How a Medium Was Born [Geek History]”. 2013. https://www.makeuseof.com/tag/the-evolution-of-the-podcast-how-a-medium-was-born-geek-history/

SiriusXM Media. 2024. “Host-Read Ads Vs Prerecorded Ads: Which Is Better for Your Campaign”. 2024. https://www.siriusxmmedia.com/insights/host-read-ads-vs-prerecorded-ads-which-is-better-for-your-campaign

Spangler, Todd. 2024. “Joe Rogan's Spotify Deal Renewal Worth up to $250 Million, Podcast Will No Longer Be Exclusive to the Platform”. 2024. https://variety.com/2024/digital/news/joe-rogan-renews-spotify-deal-not-exclusive-1235895424/

Spotify. 2023b. “How Do Podcasts Make Money, And When to Start Monetizing Yours”. 2023. https://creators.spotify.com/resources/grow/how-do-podcasts-make-money

Spotify. 2023a. “Spotify Podcast Dataset”. 2023. https://podcastsdataset.byspotify.com/

Spotify for Creators. 2024. “Video Insights to Elevate Your Content Strategy”. 2024. https://creators.spotify.com/resources/podcast-fan-study

Steele, Anne. 2024. “Spotify Takes Aim at Youtube in Battle for Podcasts”. 2024. https://www.wsj.com/business/media/spotify-takes-aim-at-youtube-in-battle-for-podcasts-ad6db6a7

YouTube Help. 2025a. “About Video Ad Formats”. 2025. https://support.google.com/youtube/answer/2375464

YouTube Help. 2025b. “Add Paid Product Placements, Sponsorships & Endorsements”. 2025. https://support.google.com/youtube/answer/154235

yt-dlp. 2025. “Yt-Dlp”. 2025. https://github.com/yt-dlp/yt-dlp

Bibliography - Statistics

Cumulus Media, Signal Hill Insights. 2024. “Preferred Podcast Ad Types in the United States as of October 2023”. 2024. https://www.statista.com/statistics/1238793/podcast-ad-types-usa/

IAB (U.S.). 2022. “Distribution of Podcast Advertising Revenue in the United States from 2019 to 2021, By Ad Type”. 2022. https://www.statista.com/statistics/1238595/us-podcast-advertising-revenue-type/

PwC; IAB (U.S.). 2024. “Podcast Advertising Spending in the United States from 2020 to 2026 (In Billion U.S. Dollars) [Graph]”. May 9, 2024. https://www.statista.com/statistics/610071/podcast-ad-spending-us/

Statista. 2024. “Number of Users of Podcasts in the United States from 2020 to 2029 (In Millions) [Graph]”. 2024. https://www.statista.com/forecasts/1123105/statista-amo-podcast-reach-us

Westwood One. 2021. “Average Number of Ads Per Podcast Considered Appropriate by Podcast Listeners in the United States as of March 2021, By Podcast Length”. 2021. https://www.statista.com/statistics/1205728/number-accepted-ads-per-podcast-usa/

Glossary

AI **– artificial intelligence** 46

ANN **– artificial neural network** 6, 6

ASR **– automatic speech recognition** 21

AUC **– Area Under the Curve** 60, 60, 62

BPE **– byte-pair encoding** 9, 31

BPTT **– backpropagation through time** 8

DAI **– dynamic ad insertion** 2, 15, 15, 15, 16, 21, 25, 25, 75, 75, 75

EEB **– energy-entropy block** 20

FN **– false negative** 10, 66

FNN **– feedforward neural network** 7, 7, 8, 8, 22, 32

FP **– false positive** 10, 66, 79

FPR **– false positive rate** 10, 63, 63, 63, 63, 63, 63

GPT **– generative pre-trained Transformer** 30, 80

HMM **– hidden Markov model** 18, 18, 18, 18, 18, 18

HZCRR **– high-zero-crossing-rate ratio** 20

IAA **– inter-annotator agreement** 75

LDA **– latent Dirichlet allocation** 22, 30

LLM **– large language model** 23, 80, 80

LSTM **– long short-term memory** 18, 18, 21, 21

MFCCs **– mel-frequency cepstral coefficients** 17, 30, 50

ML **– machine learning** 6, 6, 6, 13, 13, 14, 28

MTBFP **– mean time between false positives** 63, 63, 63, 63, 63, 63, 65, 65

NER **– named-entity recognition** 5

NLP **– natural language processing** iv, 2, 3, 5, 5, 5, 5, 5, 6, 6, 6, 6, 18, 18, 21

NN **– neural network** 6, 6, 7, 11, 68

OOD **– out-of-domain** v, 3, 3, 20, 21, 21, 27, 28, 34, 57, 57, 57, 58, 65, 66, 73, 73, 73, 79, 79, 79

OOV **– out-of-vocabulary** 10

PoS **– part-of-speech** 5

RNN **– recurrent neural network** 6, 7, 7, 8, 8, 8, 8

ROC **– receiver operating characteristics** 11, 11, 12, 12, 13, 34, 35, 60, 60, 62, 62, 62, 62, 63, 65

RSS **– really simple syndication** 1, 1, 1, 1, 2, 3, 4, 4, 15, 38, 38, 78

SLI **– spoken language identification** 22, 22

***SNS* – speech / non-speech** 17, 17, 18

***SVM* – support vector machine** 48, 50, 50, 52, 65, 79

***TF-IDF* – term frequency–inverse document frequency** 22

***TN* – true negative** 10, 66

***TP* – true positive** 10, 66

***TPR* – true positive rate** 63, 63, 63, 63, 63, 65, 67

***t-SNE* – t-distributed stochastic neighbor embedding** 69, 69, 69, 69, 70, 70, 71, 76

A. Audio Podcast Charts

Position	Title	Producer	RSS Feed
1	Call Her Daddy	Alex Cooper	https://feeds.simplecast.com/mKn_QmLS
2	The Daily	The New York Times	https://feeds.simplecast.com/Sl5CSM3S
3	Kill List	Wondery | Novel	https://rss.art19.com/kill-list
4	The Joe Rogan Experience	Joe Rogan	https://feeds.megaphone.fm/GLT1412515089
5	Dateline NBC	NBC News	https://podcastfeeds.nbcnews.com/HL4TzgYC
6	Crime Junkie	audiochuck	https://feeds.simplecast.com/qm_9xx0g
7	The Deck Investigates	audiochuk	https://feeds.simplecast.com/sxk7RTyt
8	SmartLess	Jason Bateman, Sean Hayes, Will Arnett	https://feeds.simplecast.com/hNaFxXpO
9	Pod Save America	Crooked Media	https://feeds.simplecast.com/dxZsm5kX
10	The Megyn Kelly Show	SiriusXM	https://feeds.simplecast.com/RV1USAfC
11	Up First from NPR	NPR	https://feeds.npr.org/510318/podcast.xml
12	The Ben Shapiro Show	The Daily Wire	https://feeds.megaphone.fm/WWO8086402096
13	Pardon My Take	Barstool Sports	https://mcsorleys.barstoolsports.com/feed/pardon-my-take
14	Candyman	CBS News	https://rss.art19.com/candyman
15	Morbid	Morbid Network | Wondery	https://rss.art19.com/morbid-a-true-crime-podcast
16	Wiser Than Me with Julia Louis-Dreyfus	Lemonada Media	https://www.omnycontent.com/d/playlist/796469f9-ea34-46a2-8776-ad0f015d6beb/6bf95617-5ffc-4269-a957-afcc015a4d66/ca180b09-a8bc-4612-92fe-afcc015ae720/podcast.rss
17	Shawn Ryan Show	Shawn Ryan	https://rss.pdrl.fm/55dc8e/feeds.megaphone.fm/WWO7410387571
18	All There Is with Anderson Cooper	CNN	https://feeds.megaphone.fm/WMHY6497875466
19	Stuff You Should Know	iHeartPodcasts	https://www.omnycontent.com/d/playlist/e73c998e-6e60-432f-8610-ae210140c5b1/a91018a4-ea4f-4130-bf55-ae270180c327/44710ecc-10bb-48d1-93c7-ae270180c33e/podcast.rss
20	The Tucker Carlson Show	Tucker Carlson Network	https://feeds.megaphone.fm/RSV1597324942
21	Ghost Story	Wondery | Pineapple Street Studios	https://rss.art19.com/ghost-story
22	The Ramsey Show	Ramsey Network	https://feeds.megaphone.fm/RM4031649020
23	The Dan Bongino Show	Cumulus Podcast Network | Dan Bongino	https://feeds.megaphone.fm/WWO3519750118

Position	Title	Producer	RSS Feed
24	The Man in the Black Mask	NBC News	https://podcastfeeds.nbcnews.com/the-man-in-the-black-mask
25	Criminal Attorney	Wondery	https://rss.art19.com/criminal-attorney
26	Money Crimes with Nicole Lapin	Crime House	https://feeds.megaphone.fm/moneycrimes
27	20/20	ABC News	https://feeds.megaphone.fm/ESP3456903052
28	The Ezra Klein Show	New York Times Opinion	https://feeds.simplecast.com/kEKXbjuJ
29	My Favorite Murder with Karen Kilgariff and Georgia Hardstark	Exactly Right Media – the original true crime comedy network	https://www.omnycontent.com/d/playlist/e73c998e-6e60-432f-8610-ae210140c5b1/bdde8bb3-169d-43b1-91d3-b24c0047969c/f450d41f-16bc-4ecd-8f6c-b24c004796e2/podcast.rss
30	The Dan Le Batard Show with Stugotz	Dan Le Batard, Stugotz	https://feeds.megaphone.fm/ESP2298543312
31	The Mel Robbins Podcast	Mel Robbins	https://feeds.simplecast.com/UCwaTX1J
32	Huberman Lab	Scicomm Media	https://feeds.megaphone.fm/hubermanlab
33	Before We Go	Podcast Nation	https://feeds.megaphone.fm/BOOJE7773696316
34	The Toast	Dear Media	https://rss.art19.com/the-toast
35	The Deck	audiochuck	https://feeds.simplecast.com/ZH32N6UM
36	This Past Weekend w/ Theo Von	Theo Von	https://feeds.megaphone.fm/thispastweekend
37	Armchair Expert with Dax Shepard	Armchair Umbrella	https://rss.art19.com/armchair-expert
38	Fantasy Footballers - Fantasy Football Podcast	Fantasy Football	https://feeds.megaphone.fm/fantasy-football
39	The Binge Cases: Denise Didn't Come Home	Sony Music Entertainment	https://rss.pdrl.fm/c6e873/feeds.megaphone.fm/fakepriest
40	What Now? with Trevor Noah	Spotify Studios	https://feeds.megaphone.fm/GLT9681566788
41	Calm Parenting Podcast	Kirk Martin	https://feeds.megaphone.fm/CELBL1853551588
42	New Heights with Jason & Travis Kelce	Wave Sports + Entertainment	https://rss.art19.com/new-heights
43	MrBallen Podcast: Strange, Dark & Mysterious Stories	Ballen Studios	https://rss.art19.com/MrBallen-Podcast
44	The Charlie Kirk Show	Charlie Kirk	https://www.omnycontent.com/d/playlist/5e27a451-e6e6-4c51-aa03-a7370003783c/c865b590-c84f-4f7e-a43e-ac64014b61d9/8978e846-cacd-4d65-b085-ac64014cd49f/podcast.rss
45	Money Rehab with Nicole Lapin	Money News Network	https://feeds.megaphone.fm/TPG1669083856
46	This American Life	This American Life	https://www.thisamericanlife.org/podcast/rss.xml
47	The Bulwark Podcast	The Bulwark	https://audioboom.com/channels/5114286.rss
48	Morning Wire	The Daily Wire	https://feeds.simplecast.com/WCb5SgYj

Position	Title	Producer	RSS Feed
49	The Jamie Kern Lima Show	Jamie Kern Lima	https://feeds.megaphone.fm/LIFEISLOVELLC4342849321
50	The Bible in a Year (with Fr. Mike Schmitz)	Ascension	https://feeds.fireside.fm/bibleinayear/rss
51	The School of Greatness	Lewis Howes	https://feeds.simplecast.com/AAvup9Zz
52	48 Hours	CBS News	https://rss.art19.com/48-hours
53	Hidden Brain	Hidden Brain, Shankar Vedantam	https://feeds.simplecast.com/kwWc0lhf
54	The Commercial Break	Commercial Break LLC	https://feeds.megaphone.fm/tcb
55	Horoscope Weekly with Aliza Kelly	OpenMind	https://feeds.megaphone.fm/horoscopeweekly
56	Candace	Candace Owens	https://feeds.megaphone.fm/candace
57	The NPR Politics Podcast	NPR	https://feeds.npr.org/510310/podcast.xml
58	Conan O'Brien Needs A Friend	Team Coco & Earwolf	https://feeds.simplecast.com/dHoohVNH
59	How To Destroy Everything	HTDE | QCODE	https://feeds.megaphone.fm/QCD2353438940
60	Giggly Squad	Hannah Berner & Paige DeSorbo	https://feeds.acast.com/public/shows/780a3042-864b-40fa-ae09-4557aec621c3
61	The Matt Walsh Show	The Daily Wire	https://feeds.simplecast.com/pp_b9xO6
62	I am Charles Schwartz Show	Charles Schwartz	https://feeds.libsyn.com/526583/rss
63	Bad Friends	Andrew Santino and Bobby Lee	https://feeds.megaphone.fm/TPC1602991613
64	The Diary Of A CEO with Steven Bartlett	DOAC	https://feeds.megaphone.fm/thediaryofaceo
65	The Rest Is History	Goalhanger	https://feeds.megaphone.fm/GLT4787413333
66	Raising Parents with Emily Oster	The Free Press	https://feeds.megaphone.fm/raisingparents
67	So Supernatural	Parcast Network	https://feeds.simplecast.com/kC8PV5p6
68	Noble	Wavland	https://feeds.simplecast.com/vmQnidN1
69	Fresh Air	NPR	https://feeds.npr.org/381444908/podcast.xml
70	Leap Academy with Ilana Golan	Ilana Golan	https://feeds.megaphone.fm/YAP7305356886
71	Young and Profiting with Hala Taha	Hala Taha | YAP Media Network	https://feeds.megaphone.fm/yap
72	Office Ladies	Earwolf & Jenna Fischer and Angela Kinsey	https://feeds.megaphone.fm/office-ladies
73	Las Culturistas with Matt Rogers and Bowen Yang	Big Money Players Network and iHeartPodcasts	https://www.omnycontent.com/d/playlist/e73c998e-6e60-432f-8610-ae210140c5b1/f6816727-c503-47ac-a7ac-ae2700391b1e/935c500f-8bb0-436b-ba7f-ae2700391b49/podcast.rss
74	All-In with Chamath, Jason, Sacks & Friedberg	All-In Podcast, LLC	https://allinchamathjason.libsyn.com/rss
75	Up and Vanished	Tenderfoot TV	https://feeds.megaphone.fm/up-and-vanished

Position	Title	Producer	RSS Feed
76	The Weekly Show with Jon Stewart	Comedy Central	https://feeds.megaphone.fm/BVLLC2163264914
77	The Binge Crimes: Night Shift	Sony Music Entertainment / Campside Media	https://feeds.megaphone.fm/crimesnightshift
78	Scamanda	Lionsgate Sound	https://feeds.acast.com/public/shows/66d9aa05d740de0852b686a8
79	REAL AF with Andy Frisella	Andy Frisella #100to0	https://mfceoproject.libsyn.com/rss2
80	Against the Rules with Michael Lewis	Pushkin Industries	https://www.omnycontent.com/d/playlist/e73c998e-6e60-432f-8610-ae210140c5b1/f4e26994-3ddb-4ec7-b855-ae32006cd5de/ea4fae0c-3282-4cfd-b315-ae32006cd5ec/podcast.rss
81	Serialously with Annie Elise	10 to LIFE & Audioboom Studios	https://audioboom.com/channels/5100770.rss
82	Full Body Chills	audiochuck	https://feeds.simplecast.com/6E74sk6q
83	NPR News Now	NPR	https://feeds.npr.org/500005/podcast.xml
84	The Bible Recap	Tara-Leigh Cobble	https://feed.podbean.com/thebiblerecap/feed.xml
85	The Joe Budden Podcast	The Joe Budden Network	https://jbpod.libsyn.com/applepodcast
86	Scoop City: A show about the NFL	The Athletic	https://feeds.megaphone.fm/TAMC6272189068
87	The Wonder of Stevie	Higher Ground, Pineapple Street Studios	https://rss.art19.com/the-wonder-of-stevie
88	KILL TONY	DEATHSQUAD.TV & Studio71	https://feeds.megaphone.fm/killtony
89	Unlocking Us with Brené Brown	Vox Media Podcast Network	https://feeds.megaphone.fm/GLT4889391284
90	Passion Struck with John R. Miles	John R. Miles	https://feeds.simplecast.com/_BDqCTvj
91	Last Podcast On The Left	The Last Podcast Network	https://feeds.simplecast.com/dCXMIpJz
92	We Can Do Hard Things	Glennon Doyle and Audacy	https://feeds.megaphone.fm/wcdht
93	On Purpose with Jay Shetty	iHeartPodcasts	https://www.omnycontent.com/d/playlist/e73c998e-6e60-432f-8610-ae210140c5b1/32f1779e-bc01-4d36-89e6-afcb01070c82/e0c8382f-48d4-42bb-89d5-afcb01075cb4/podcast.rss
94	The Kevin O'Connor Show	Yahoo Sports	https://rss.art19.com/the-kevin-o-connor-show
95	The Daily Show: Ears Edition	Comedy Central and iHeartPodcasts	https://www.omnycontent.com/d/playlist/e73c998e-6e60-432f-8610-ae210140c5b1/e5e49f91-be9b-42f1-a426-ae3c00026e8b/04b51c34-8028-49a3-b42f-ae3c00026e95/podcast.rss
96	Pivot	New York Magazine	https://feeds.megaphone.fm/pivot
97	Andrew Schulz's Flagrant with Akaash Singh	Andrew Schulz's Flagrant with Akaash Singh	https://feeds.megaphone.fm/APPI6857213837
98	Throwbacks with Matt Leinart & Jerry Ferrara	Matt Leinart, Jerry Ferrara	https://feeds.megaphone.fm/throwbacks
99	The Glenn Beck Program	Blaze Podcast Network	https://feeds.megaphone.fm/BMDC3567910388
100	Freakonomics Radio	Freakonomics Radio + Stitcher	https://feeds.simplecast.com/Y8lFbOT4

B. False Negatives

YouTube-Id	Start	End	Text	Prediction	Transcription Accurate	Ad Present
RwjuE2eUo78	6.12	9.88	You might have seen my interview with wealthy expats who does exactly this for you.	0.23323	1	1
QRoKCAN5mmo	22.84	98.44	This episode of two bears one cave is brought to you by NASCAR Don't miss out and invite over some friends and family to watch the yellowwood 500 on Sunday October 1st at 2 p.m. Eastern on NBC We are supported by Freeze pipe for the smoothest cannabis smoking experience you got to try a freeze pipe freeze pipe mix a unique line of Freezable pipes bugglers and bongs that cool smoke by hundreds of degrees for icy smooth toaks without the throwburn chest pain or coughing attacks they even have freezeable devices built specifically for cooling joints blunts and vapes The secret is the detachable glycerin chambers that come on every Piece pop one of these chambers in the freezer for one hour and a smoke passes through it It's instantly chilled by over three hundred degrees proven to outperform traditional pipes and bongs Simply inhale and relax as freeze pipes icy glycerin chambers do all the heavy lifting this thing makes the smoking Experience so much smoother so much cooler literally so much more enjoyable Shop the smoothest pipes bugglers bongs and dab rigs at everyday great prices by visiting the freeze pipe calm and use the code Bears for 10% off your entire order.	0.17362	1	1
WgyFeTnJ6KE	612.7	616.08	So they make it easy and free to change counselors if needed.	0.22717	1	1
9DJfIKNs32w	107.21	108.82	All you got to do is follow the directions.	0.01373	1	1
ukr7aTfl868	4977.72	4979.08	You already see the lighting, okay?	0.06909	1	1
85LE-8LUaKA	1437.33	1438.18	Hello fresh.	0.29192	1	1
id08k63Mki8	434.56	436.16	Rukions are clearly labeled on the packaging.	0.07414	1	1
Gj5uEr0Lh-k	181.03	182.34	Grammily helps you revise.	0.13376	1	1
A8ZXGOlHACY	268.66	280.88	And I urged everybody to go and pre-order my latest book, The Shrouded Lighthouse.	0.03192	1	1
0fpC8CB22YI	368.78	372.74	Oh Man, I mean I was curious about that.	0.1831	1	0
2qYRuxQWcKU	2292.84	2297.22	For years, I searched me opponents measly hands and found only scurvy dogs.	0.12559	1	1
lee7wJMMt68	2599.38	2600.32	Now back to this episode.	0.11159	1	1
5V_6rh1moKM	2188.32	2189.98	No questions asked.	0.16261	1	1
bPiAvNjh-FY	1782.06	1782.72	Oh, yeah, no.	0.00897	1	1
rJgBPZV7wTM	1744.2	1746.28	Adam Freeman hits his 15th anniversary.	0.1917	1	1
bB5x3C3pFwY	1990.18	1994.96	The most important thing is to enjoy yourself and have a great time with friends and family.	0.13208	1	0

YouTube-Id	Start	End	Text	Prediction	Transcription Accurate	Ad Present
m8_Ou2GQ2vA	1626.8	1632.5	If you're looking to run a CEDH or 1v1 tournament, eminent's gaming has your back with command tower.	0.03176	1	1
LxtZ5shc19I	62.52	64.74	When this stuff's coming in, you're like, no, this isn't mine.	0.24751	1	1
yR6aF26I3eI	1433.18	1434.14	What are the best odds?	0.09238	1	1
av9o2KPKDn0	3531.3	3534.22	Cracks, spills and mechanical failures, you're covered.	0.3222	1	1
NhtxI-ftyUs	3141.98	3144.9	Or should we spend millions of dollars to have the guy sing it?	0.26717	1	0
kAgrE6Gyluk	4310.74	4594.22	Music Music Music Music Music Music Music Music Music Music Music Music Music Music Music Music Music	0.10426	0	0
pOxSbG384Sw	1613.22	1616.21	When the aliens ask you, wow, how are you so prepared?	0.11507	1	1
0p16Kd27Nak	4594.28	4595.56	Be it a machine.	0.08645	1	0
bY71hjQt20o	3324.92	3327.8	Usually I lose and only get one or zero.	0.07459	1	1
TCRn_GUdkEw	1128.68	1129.56	And then here's how it works.	0.13776	1	1
3F1JDaSI24Y	1864.34	1864.99	Do you sleep on your side?	0.30451	1	1
3x8FLngZ1h0	5003.42	5008.7	We're going to take a quick break to hear from the sponsor of this entire show Ryan Reynolds is folks over at Mint Mobile.	0.23923	1	1
1TDYfcH-dVU	1891.08	1893.6	And it ships right to your door in a discrete package.	0.24945	1	1
jU_fuCZ0erE	2474.18	2485.66	I didn't feed suck and coordinating with friends is a nightmare I always end up running a bunch of money and chasing down my friends to get reimbursed and if they flake I'm stuck with the whole bill.	0.0665	0	1
iz5Uf9VeJGE	4493.4	4517.62	You You You You You You You You You You You You You You You You You You You You You You You You You You You	0.06481	0	0
id08k63Mki8	420.0	422.08	Cook, you need gives me quality and variety.	0.2096	1	1
k7DtQnG0mZA	3319.04	3324.14	On their show, you'll hear a chorus of perspectives from the diverse communities behind the code.	0.0562	1	1
segwKd_qwm4	1960.34	1962.82	Can we play with him he'll be so funny.	0.03504	1	0
E0mIntdxUAs	2071.46	2078.48	There's definitely like a different level of class to it that I've been enjoying a lot and everybody that I shown it to is super impressive as well.	0.12024	1	1
Fszw5-qc8yA	1514.09	1515.4	You just go on their website.	0.36951	1	1
VdWbSAtFAsU	163.84	164.76	You don't want to hear this.	0.07516	1	1
qOFepgVRbI8	1455.02	1484.98	house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the	0.18624	0	0

YouTube-Id	Start	End	Text	Prediction	Transcription Accurate	Ad Present
			house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting hotter and the house because it's getting			
Jhfyn5F39FM	3563.62	3567.1	And it's not just the same lack of luster gifts that you find anywhere else.	0.09338	1	1
8oB75VIUR7c	157.85	161.13	So get your degree, put your glasses on.	0.33464	1	1
CaD9nTRMaeU	899.88	904.88	Tom Geico asks, how would you love a chance to save some money on insurance?	0.21632	1	1
i26iBddxvd0	174.44	175.8	No, we've well, we have.	0.05158	1	0
fc4ZxyXSPN0	141.06	143.04	So, thank y'all for continuing to rock with me.	0.07287	1	1
ZVkjfqLqU2k	455.46	458.46	It says book walker on the book walker site.	0.11252	1	1
HlC5lKxEf6k	62.42	64.32	And these are my words completely.	0.17053	1	1
h-aaedXe61o	514.28	522.32	Their cases are optimized for protection up to 11.5 feet which is 5 times the military standard with standing drops up to 130 times.	0.29239	1	1
MlhFdhn3dXY	2272.45	2276.5	Inside of yours brought to you by Neurohacker, Qualia Cinematic.	0.40092	1	1
z6atNBhItBs	2963.96	2993.84	that I think that I	0.02158	0	0
1iLzhNyQoXU	1264.06	1266.6	It was my second failure.	0.0632	1	0
segwKd_qwm4	32.56	33.32	Use these templates.	0.30729	1	1

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe. Sofern im Zuge der Erstellung der vorliegenden Abschlussarbeit generative Künstliche Intelligenz (gKI) basierte elektronische Hilfsmittel verwendet wurden, versichere ich, dass meine eigene Leistung im Vordergrund stand und dass eine vollständige Dokumentation aller verwendeten Hilfsmittel gemäß der Guten Wissenschaftlichen Praxis vorliegt. Ich trage die Verantwortung für eventuell durch die gKI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate.

Ort, Datum	Felix V. J. Wolf

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Ort, Datum	Felix V. J. Wolf