UNIVERSITY OF
GOTHENBURG

Classifying written and spoken text

A comparison between BERT and Naive Bayes

Master's thesis in Mathematical Statistics

JENS IFVER & CALVIN SMITH

Department of Mathematical Sciences
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

Classifying written and spoken text

A comparison between BERT and Naive Bayes

JENS IFVER & CALVIN SMITH

UNIVERSITY OF GOTHENBURG

Department of Mathematical Sciences
Division of Applied mathematics and statistics
Financial mathematics
University of Gothenburg
Gothenburg, Sweden 2022

Classifying written and spoken text
A comparison between BERT and Naive Bayes
JENS IFVER & CALVIN SMITH

Supervisor: Mattias Wahde, Department of Applied Mechanics
Examiner: Torbjörn Lundh, Department of Mathematical Sciences

Master's Thesis 2022
Department of Mathematical Sciences
Division of Applied mathematics and statistics
University of Gothenburg
SE-412 96 Gothenburg

Classifying written and spoken text
JENS IFVER & CALVIN SMITH
Department of Mathematical Sciences
University of Gothenburg

Abstract

In the field of natural language processing (NLP), written or spoken communication is modeled using machine learning algorithms. However, the lack of data generated from spoken language has led to most pre-trained NLP models being trained on written texts. This could perhaps lead to problems in the performance of the model since there is a significant difference between written and spoken language. The study will investigate how well the machine learning model BERT, in comparison to a naive Bayes classifier, can distinguish between written and spoken language and whether there is a difference in BERT's ability to predict masked words for both classes. The results indicate that both BERT and the naive Bayes classifier are able to separate written and spoken text fairly accurately, achieving a mean accuracy of 80.8% and 75.7% respectively. However, an interesting and quite surprising result is that the naive Bayes classifier outperforms BERT at classifying spoken sentences. The masked word predictions showed that BERT performs poorly at predicting masked words that are common in the spoken data and uncommon in the written data. This could imply that BERT has a lesser understanding of spoken language in general.

Keywords: BERT, natural language processing, naive Bayes classifier

Acknowledgements

We want to express our gratitude to our supervisor Mattias Wahde for his support throughout this master's project.

We also want to thank our study colleagues and in particular Jonatan Hellgren for all the laughs and discussions both in and outside of the university, for the past five years. Despite the fact that we most of the time did not discuss mathematics, we are sure that this was a key component in making these master's degrees possible.

Jens Ifver & Calvin Smith, Gothenburg, Nov 2022

Contents

1 Introduction

In applied machine learning the training environment should resemble what the model eventually will be exposed to once employed. That means using training data similar to the future unseen data of the real-world problem that the model is supposed to solve. The models sensitivity for diverging data depends partly on the machine learning algorithm and choice of hyperparameters, but these choices can only provide support to a certain degree. Therefore, many employed machine learning models need regular updates to handle the ever-changing world of real-life problems. Thus, the question of whether the new unseen data is similar enough to the data used for training needs to be asked continuously. In the field of natural language processing (NLP), verbal or text communication is modeled using machine learning algorithms. However, the lack of transcribed verbal communication has led to most NLP models being trained on written texts. In 2018 Google introduced the language model BERT [1], a transformer-based neural network that can be used for a wide variety of NLP tasks. As with most NLP models, BERT has primarily been trained on written texts. This study investigates how well the language model BERT, in comparison to a naive Bayes classifier, can distinguish between written and spoken data. Further, this study investigates how the performance of BERT changes when using written text compared to spoken text.

1.1 Background

The structural differences between written and spoken language are well established in the scientific literature. In general, spoken language is characterized by being less abstract and containing fewer unique words compared to written language. This means that when speaking, people tend to use a smaller vocabulary and form shorter sentences than when writing [2] [3]. Further, when quantifying the differences between written and spoken language, studies have found that spoken language contains shorter words and shorter sentences compared to written language [4].

In the field of NLP and in particular conversational AI [5], deep neural networks have become the standard. In 2018 Google introduced BERT (Bidirectional Encoder Representations from Transformers), a transformer-based neural network that can be used for a wide variety of NLP related tasks [1]. The introduction of BERT was a breakthrough in the field of NLP because of its impressive results on a number of different popular NLP tasks. Since then other bigger networks have been created but BERT is still considered a benchmark. BERT was trained on every text from the English version of Wikipedia (2,500M words) and a dataset containing books from unpublished authors called BooksCorpus (800M words). While these texts may contain some quotes and dialogue, they are dominated by written language in the form of articles and books. The original BERT model has been used in multiple NLP subfields like chatbots [6], natural language understanding [7] and automatic speech recognition [8]. In these fields spoken language rather than written language is the prevalent form.

1.2 Aim

In this study, two main questions will be investigated. Firstly, can machine learning techniques in general, and BERT in particular, distinguish between written and spoken text? Secondly, is there a difference in BERT's performance on spoken text and written text in terms of its ability to predict masked words?

1.3 Sub-problems

In relation to the background and aim, this study will be looking at the following problems and questions:

- How well can BERT, compared to a naive Bayes classifier, distinguish between written and spoken text?

- How does the performance of BERT change when using spoken text compared to written text?

1.4 Method and limitations

The work has been limited to BERT and naive Bayesian classifiers in exploring how well they can distinguish between written and spoken text. While BERT is popular amongst methods based on deep neural networks and transformers, other methods such as RoBERTa or ELMo that have been trained on large corpuses of written text could have been be implemented instead. Further, the naive Bayes classifier serves well as a baseline model because of its simplicity. Other methods like Linear Support Vector Machine or logistic regression could also have been used.

The spoken data that has been collected consists mostly of dialogues between people in the form of radio shows and podcasts. It could be argued that this type of spoken language is not representative of spoken language in general. However, other forms of spoken language such as speeches or monologues are often scripted and thus could be similar to written language.

1.5 Contents

In order to explore the questions above we have studied relevant theory in NLP, BERT and text classification. The theory involved in this study will be presented in Chapter 2. A large portion of the project has also been devoted to the process of collecting, cleaning and transforming data into a suitable format. This process is described in detail in Chapter 3. Furthermore, we have implemented a fine-tuned BERT model and a naive Bayes classifier on the binary classification problem of distinguishing between written and spoken text. We have also used BERT for masked word predictions in order to analyze the differences of BERT's performance on written and spoken text. The methods and results of the project are presented in Chapter 3 and 4 respectively. All coding has been implemented in Python. In Chapter 5 we will discuss and conclude our results.

2

Theory

The following chapter introduces the theoretical framework required for the work presented here. Section 2.1 is an introduction to language models and word embeddings. This is followed by an introduction to the naive Bayes (NB) classifier in Section 2.2. In Section 2.3 the Transformer architecture and the concept of *attention* is introduced. Lastly, the BERT model is introduced in Section 2.4.

2.1 Language models and word embeddings

Word embeddings are functions that maps words, from a collection of words, to an n-dimensional vector space. The simplest form of word embeddings are one-hot vectors, where each word is mapped to a vector of size N (N is the number of unique words in the collection). To distinguish between each word, the one-hot vector has a single cell with the value 1 and the rest of the cells consist of 0s. Further, more advanced word embeddings aim to create vector representations of words that capture both the semantic and syntactic information of each word. Even though a one-hot vector is technically a word embedding, it will henceforth be implied that word embeddings refer to a vector representation aimed at capturing both semantics and syntactics.

Language models are a collection of models created to, as the name implies, *model* language. What it means to model language can vary depending on the objective and different models have different approaches to creating representations of text. A language model that creates word embeddings is a model that, given a text corpus, creates a *representational* word embedding vector space in $\mathbb{R}^n$. The idea behind a language model with word embeddings is to create a vector representation of the words from a text corpus that as accurately as possible captures the meaning of each word relative to every other word. Since the words are mapped to a word embedding vector space, mathematical operations can be performed to infer on the relationship between different words and their meaning. To illustrate this, let π be a word embedding mapping $W \longrightarrow \mathbb{R}^n$, where W is the word space and $\mathbb{R}^n$ is an n dimensional vector space. A language model that uses word embeddings and has been well trained on a large text corpus could produce the following approximate relationship:

$$\pi(water) - \pi(coffee) + \pi(bag) \approx \pi(tea) \tag{2.1}$$

Figure 2.1 is an illustration of the word embedding vector space produced by the

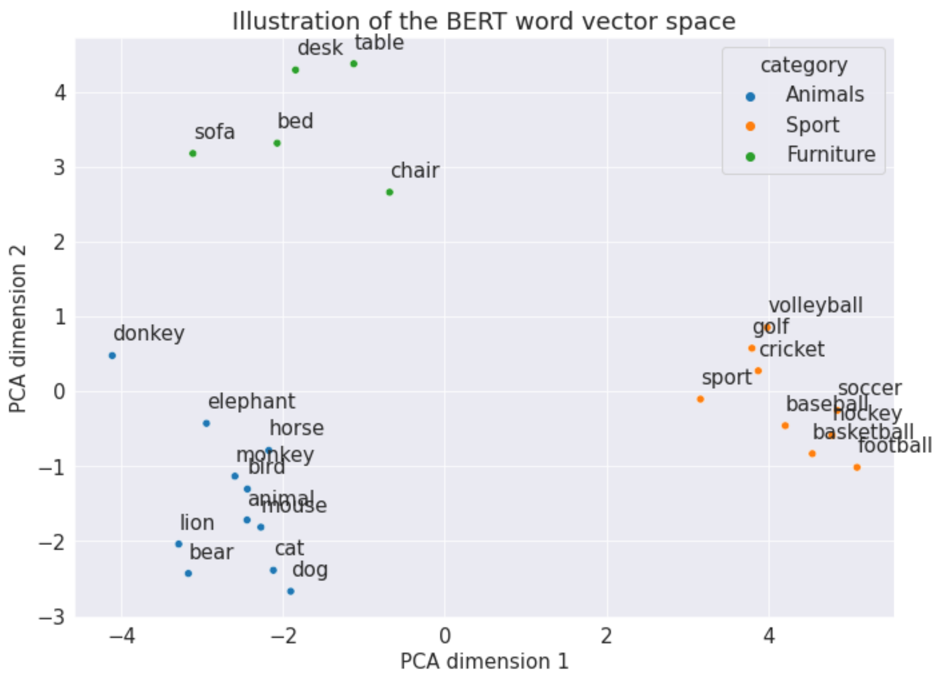

Figure 2.1: Illustration of the BERT word vector space.

language model BERT (see also Section 2.5 below). The dimensionality of BERT's output has been reduced to two-dimensions using Princicpal Component Analysis (PCA). In the figure, representations of words from three different categories have been plotted and the clusters that appear are a good illustration of the semantic properties of a well trained language model. Word embeddings can also capture contextualized meaning, where the same word can have different meanings depending on the context of the sentence. Consider the following two sentences:

"Can you saw down that tree?"

"I saw a bird in the sky."

The word "saw" has a different semantic meaning in each sentence and the embedding of the word should have different vector representations in a language model that aims to capture contextualized meaning. Figure 2.2 illustrates contextualized meaning where the word "saw" is represented by two different points depending on the sentence it belongs to.

2.2 N-grams

N-grams are defined as a sequence of N words and can be used to create language models. When using N-grams in a language model the base features are often represented as a matrix of N-gram counts, where each column represents an N-gram and each row represents some text. Each element in the matrix is then the count of how many times a particular N-gram appears in a particular text. In comparison to

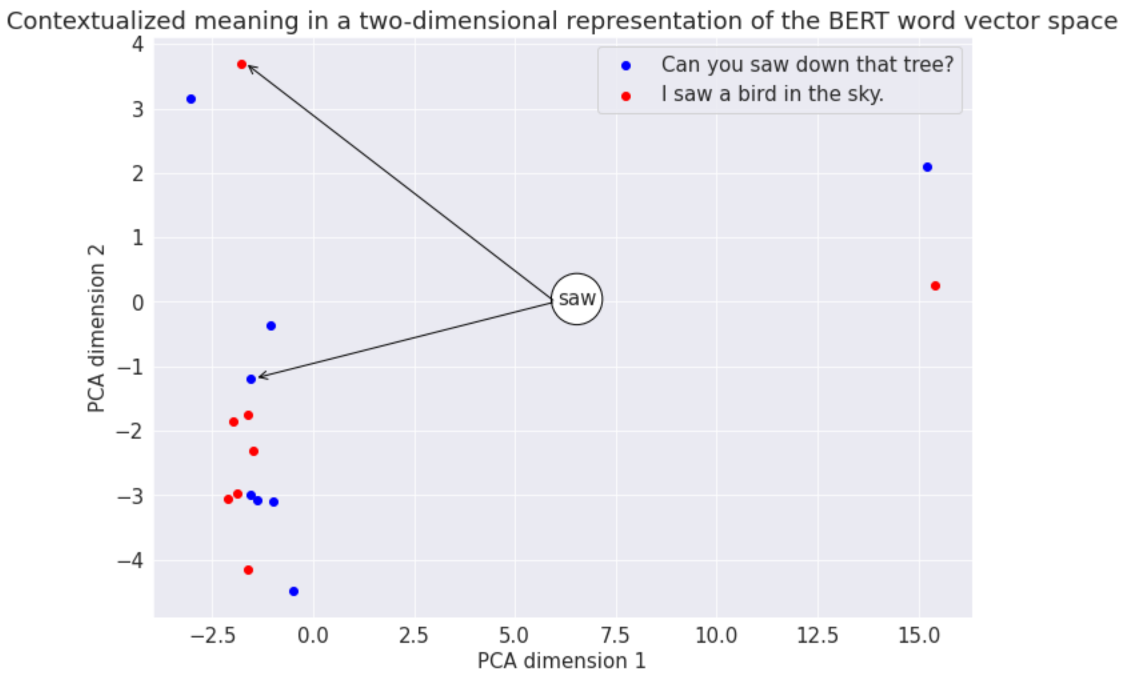

Figure 2.2: Illustration of BERT's ability to apply context in its representation. Each blue and red dot corresponds to BERT's two-dimensional representation of the corresponding token. Note that the number of tokens is not always the same as the number of words in a sentence. It depends on the length of the longest sentence and some words can be tokenized into multiple tokens.

a language model with word embeddings, a language model that uses N-grams does not create a vector representation of words that capture any meaning or relationship between words.

2.3 Naive Bayes classifier

The NB classifier is a machine learning algorithm commonly used in text classification. The algorithm is simple and fast to implement. Given an observation, the algorithm assigns the most likely class based on its feature vector. The features are assumed to be independent given the class to which a sample belongs, which is why the classifier is referred to as *naive*. Given a class c, observation y and features $x_1, x_2, ..., x_n$, Bayes theorem [9] states that the probability of observation y belonging to class c given the features is

$$P(y \in c | x_1, ..., x_n) = P(c) \frac{P(x_1, ..., x_n | y \in c)}{P(x_1, ..., x_n)}, \tag{2.2}$$

where $P(c)$ is the prior probability of class c, $P(x_1, ..., x_n | y \in c)$ is the conditional probability of the features given that observation y belongs to c and $P(x_1, ..., x_n)$ is the joint probability of the features. Using the conditional independence assumption the numerator can be expanded into separate conditional probabilities, and since the joint probability of the features is not dependent on the class, the right hand side of (2.2) becomes proportional to the left hand side as follows

$$P(y \in c | x_1, ..., x_n) \propto P(c) \prod_{i=1}^{n} P(x_i | y \in c). \tag{2.3}$$

Now, assume that C is the set of all possible classes. The observation y is then assigned to the class that produces the highest probability

$$\text{class}(y) = \arg \max_{\forall c \in C} P(c) \prod_{i=1}^{n} P(x_i | y \in c). \tag{2.4}$$

The NB classifier is often used in text classification tasks such as predicting if an email is a spam or not [10], if a movie review is positive or negative [11], or predicting which genre a book belongs to [12]. The features used in NB text classification are often represented as a matrix of n-gram frequencies from a given text corpus. Thus, the NB classifier is not a language model in the sense that it can capture meaning from text. If a NB classifier classifies a movie review as 'positive' it still does not have a numerical vector representation of what a 'positive' review means, but only the frequencies of words used in both positive and negative reviews. Despite its simplicity, this algorithm can still be effective in specific text classification tasks where semantics are not a primary concern and frequencies of words are enough to provide information for the task.

2.4 Neural networks

A more sophisticated language model utilizes some kind of neural network. While there are many variations of neural networks, this section will only provide an overview of the basic concept of *multi-layer perceptrons.*

A multi-layer perceptron (also called deep feedforward network) is a neural network consisting of multiple layers where each layer is a group of units [13]. The layers are arranged in a chain structure so that each layer is a function of the preceding one. The first layer is defined as

$$f^{(1)}(x) = g^{(1)}(W^{(1)T}x + b^{(1)}) \tag{2.5}$$

where x is the input vector and $W^{(1)}$, $b^{(1)}$, and $g^{(1)}$ are the weight matrix, bias vector and activation function corresponding to layer 1, respectively. In a multilayered structure, the output of $f^{(1)}$ corresponds to the hidden neurons in the first hidden layer of the network.

$$f^{(1)}(x) = h^{(1)} \tag{2.6}$$

The size of vector $h^{(1)}$ is defined by the architecture of the network. The proceeding layer is defined as

$$f^{(2)}(x) = g^{(2)}(W^{(2)T}h^{(1)} + b^{(2)}), \tag{2.7}$$

and so on. With this chain-like structure, we can define a three-layered network as

$$f^{(3)}(f^{(2)}(f^{(1)}(x))) = o, \tag{2.8}$$

where o is the output vector. Figure 2.3 illustrates how a multi-layer perceptron can be structured. The arrows in the figure represent elements (weights) in the weight matrix $W^{(l)}$, modeling how strong the connection between two nodes is. While there are many different activation functions, one of the most commonly used is the rectified linear unit function, which is defined as $g^{(l)}(x) = \text{ReLU}(x) = \max(0, x)$. In order to measure how well the network models the data, a loss function compares the predicted values with the target values. Fitting the network to the data involves finding weights and biases that minimize the loss function. Finding these weights and biases is done by backpropagation[14]. Backpropagation computes the gradient of the loss function in respect to the weights and biases, which is used for updating these parameters.

A common issue with neural networks is that they are prone to overfitting. Techniques that mitigate this issue are called *regularization techniques.* During this project, two such techniques have been utilized: *dropout* and *l2-regularization.* Dropout works by simply putting each neuron in a layer to zero with a specified probability. The l2-regularization instead penalizes weights with large magnitude and hence keeps the weights small.

2.5 The Transformer

The *transformer* is a sequence transduction model that transforms input sequences into output sequences [15], which is suitable for language modeling and machine translation problems. In the original article [16], the transformer was evaluated on machine translation tasks, but the architecture has since been applied in numerous state-of-the-art language models where it has been modified to some extent [17], [18]. Sequence

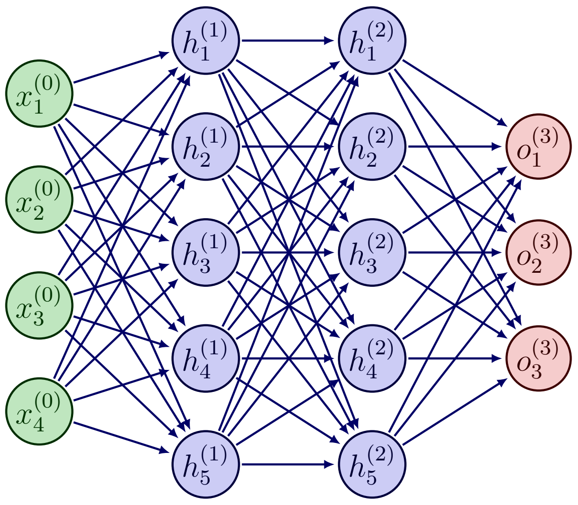

Figure 2.3: Example structure of a multilayer perceptron with four input nodes, two hidden layers with 5 nodes each and an output layer with three nodes.

transduction models that are used in the context of NLP commonly have an encoder-decoder structure [19], [20]. The objective of the encoder is to take the sequence input and encode it into a latent representation. The decoder uses the latent representation of the sequence as input and then generates a sequence. Before the *transformer* was introduced, most encoder-decoder structures were based on recurrent neural networks (RNNs) or convolutions to handle word sequences. In RNNs, sequences are processed item by item in either the right or left direction, which is highly time-consuming. The Transformer relies entirely on a mechanism called self-attention to compute representations of inputs and outputs without the use of RNNs or convolution. The self-attention process is based mainly on matrix computations which can be parallelized and are therefore computed rather quickly with a graphic processing unit (GPU). This means, among other things, that transformer-based models can be trained in a shorter amount of time or with less computer power, compared to those based on RNNs.

The transformer has an encoder-decoder structure where both the encoder and decoder have a multiple-layer structure, all trained simultaneously with a version of gradient descent. Each layer in the transformer also contains additional sublayers, see Figure 2.4. All the layers in the encoder have identical structures and two sublayers. Likewise, all layers in the decoder have the same structure but with three sublayers. There are two types of sublayers, the multi-head attention layer, and the feedforward layer. Every sublayer has a residual connection around it and is followed by a normalization [21], as follows

$$\text{LayerNorm}(x + \text{sublayer}(x)). \tag{2.9}$$

The feedforward layer consists of two linear transformations with a rectified linear unit (ReLU) activation in between. For an input x that is

$$\text{feedforward}(x) = \max(0, xW_i + b_1)W_2 + b_2. \tag{2.10}$$

The additional layer in the decoder is a multi-head attention layer (described below) that uses the output of the encoder in combination with the output from the previous

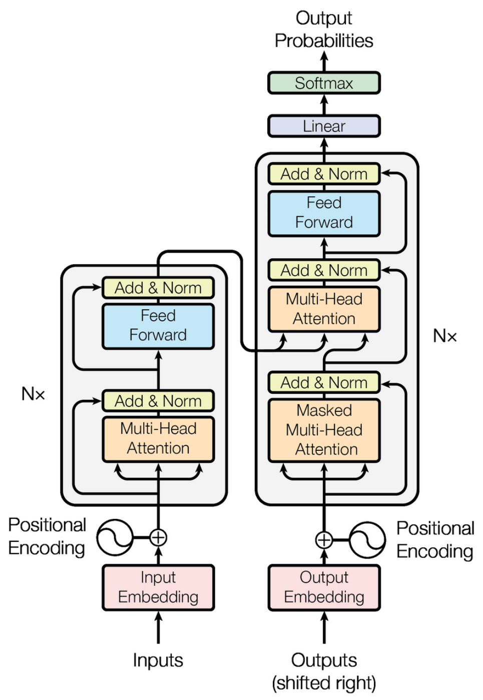

Figure 2.4: The transformer architecture. Image taken from the original article [16]. The left gray bar represents the encoder and the right represents the decoder.

layer in the decoder. The decoder is applied sequentially in order to include the previous decoder outputs in the context used to predict the next element in the sequence.

2.5.1 Self-attention

The transformer is based on an attention mechanism called self-attention. Attention is a technique to extract information about what parts of a sequence are mostly related to each other and what parts are containing the most valuable information (where to focus attention). In self-attention, for each element in a sequence, a query-, key-, and value vector is generated by using projection matrices and an embedded representation of the input. Intuitively, one can look at queries, keys, and values as parts of a retrieval system. The queries describe what to look for and are therefore matched with the keys that act like tags. If a query and a key match well, that is, two tokens are highly related, the corresponding value will be retrieved (weighted high). In the transformer, both the encoder and the decoder use a variant of self-attention called scaled dot-product attention defined as

$$\text{attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{2.11}$$

where Q, K, and V are the matrices containing the so-called queries, keys, and values respectively with dimension $d_k \times d_{model}$. The dimensions d_k correspond to the dimension of each query, key, and value vector, and d_{model} is the dimension of the embedding vector of each input and output token. In the transformer, this self-attention mechanism is run multiple times in parallel in what the authors call multihead-attention.

$$\text{Multihead}(Q, K, V) = \text{concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)W^O \tag{2.12}$$

with

$$\text{head}_i = \text{attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.13}$$

and $i \in \{1, 2, ..., h\}$, where matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_k}$ and $W^O \in \mathbb{R}^{hd_k \times d_{\text{model}}}$ are projection matrices. A masking procedure is performed to limit the attention of the decoder to only process the part of the output sequence that has already been predicted. In the original transformer article $d_k = 64$ and $d_{\text{model}} = 512$.

2.6 BERT

Bidirectional encoder representations from transformers (BERT) is a transformer-based NLP model that was introduced in 2018 in paper [22]. The model architecture of BERT is almost identical to the original transformer [16] (described above) but without the decoder stack. Instead, the BERT model consists of a stack of transformer encoders. The transformer encoder and decoder use self-attention to derive relationships between elements in a sequence. Since the encoder (in contrast to the decoder) is not restricted to only focusing attention on the left part of the sequence, it is considered bidirectional. That means that the mechanism estimates a relation between all elements in a sequence. The pre-trained version of BERT, takes a tokenized sequence of words as input and generates a high dimensional vector representation of each input token. These word and sentence embeddings contain information that can be used in a variation of NLP tasks.

There are two different sizes of the BERT model: $BERT_{\text{BASE}}$ and $BERT_{\text{LARGE}}$ with transformer parameters $L = 12$, $d_{\text{model}} = 768$, $h = 12$ and $L = 24$, $d_{\text{model}} = 1024$, $h = 16$ respectively. Here, L represents the number of layers (encoders), d_{model} the size of the individual token embeddings, and h the number of self-attention heads. BERT was trained in two steps: *pre-training* and *fine-tuning*. The pre-training consisted of masked word prediction and next sentence prediction (NSP) of unlabelled data from English Wikipedia (2500M words) and BooksCorpus (800M words) [23]. For fine-tuning, data for the specific tasks were used. See Figure 2.5 for an illustration of the training procedures for BERT.

2.6.1 Pre-training

The data used in pre-training of BERT were represented with WordPiece embeddings [19] with a 30,000 token vocabulary to represent the sequences. In addition to the WordPiece vocabulary tokens, [CLS], [SEP], and [MASK] tokens were used. Rare words that do not exist in the vocabulary are split until every part of the word has a corresponding subtoken. The [CLS] token is a classification token that acts as a representation of the whole sequence and is placed as the first token of each sequence.

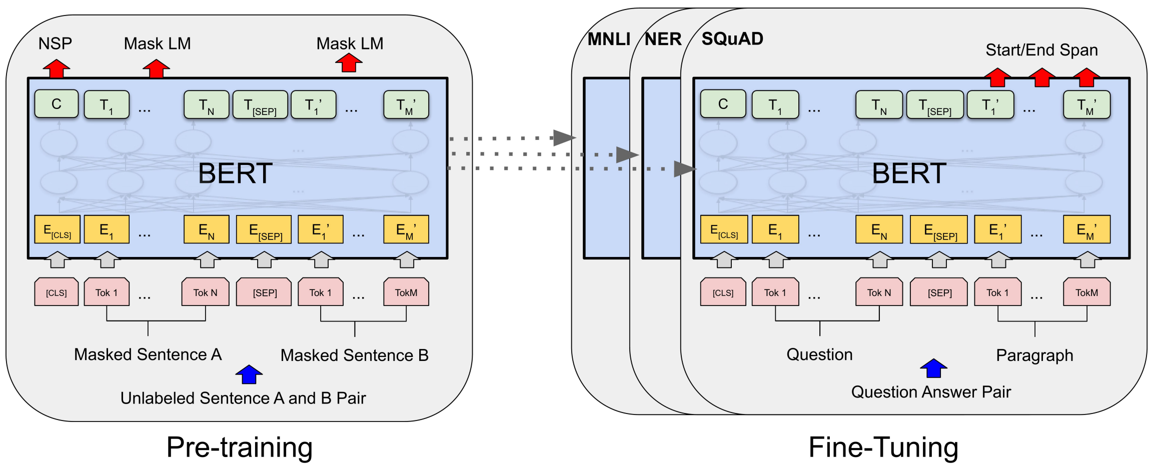

Figure 2.5: Pre-training (left) and fine-tuning (right) procedures for BERT. Image taken from the original article [22]. On the right, MNLI, NER, and SQuAD refer to separate tasks of the GLUE benchmark[24].

If a sequence contains two sentences, the [SEP] token is used to separate them in the sequence. The [MASK] token is used for masking. Since BERT does not process the sequences in a sequential manner, positional embeddings are used to provide the model with information about the position of each word. Also, segment embeddings inform BERT about to which sentence each word belongs. All three embeddings are element-wise added before being fed to the model.

The masking procedure was performed by randomly masking 15% of all tokens in each sequence with the [MASK] token. The network is then trained by only predicting the masked word tokens by using the final hidden vectors corresponding to the masked tokens. These vectors are fed to a softmax layer over the vocabulary to produce a probability for each word. For the fine-tuning part of training, masked tokens are not used. In order to prevent the model from becoming dependent on the use of [MASK] token, not all out of those 15% chosen for masking were replaced with the masked token. Out of all the tokens that were selected for masking, 80% were replaced with the [MASK] token, 10% were replaced by a random token, and 10% remained unchanged.

Sentence pairs of sentence A and sentence B are packed together into a single sequence. In order to train the model for NSP, 50% of sentences B were not actually the next sentence following sentence A. These sequences were labelled NotNext. For the remaining 50%, sentences B were the actual next sentence following A and were labeled IsNext. The final hidden vector corresponding to [CLS] is fed to a softmax layer over the two classes. During pre-training, a cross-entropy function was used to compute the loss.

2.6.2 Fine-tuning

The fine-tuning part of training refers to the training of task-specific models that are initialized with the same pre-trained parameters. For classification, the output $C \in \mathbb{R}^{d_{model}}$ corresponding to the [CLS] token is fed into an output layer. In the original article, for training on GLUE [24], the classification loss is defined as

$$\log(\text{softmax}(CW^T)), \tag{2.14}$$

where for the ith element $x_i = [CW^T]_i$

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{K} \exp(x_j)}. \tag{2.15}$$

Here $W \in \mathbb{R}^{K \times d_{model}}$ and K is the number of labels.

2.6.3 Tokenization for BERT

As mentioned above, there is a process of tokenizing the text data before it is fed to BERT. Firstly, every word is matched with a token in the vocabulary of 30,522 tokens. If a word does not exist in the vocabulary, it is split into subtokens until every part can be matched with a token. The first token of a sequence is always the [CLS] token, and the [SEP] token is always present at the end of a sentence. For masked word prediction, a selected token is exchanged with the [MASK] token, and for padding, the [PAD] token is used for padding sequences to the desired length. Every token has an index in the vocabulary list. These indices act as initial numerical representations of the tokens and account for what is being fed to BERT. Below is an example of how tokenization can be performed. BERT then performs an embedding of each index into a 768-dimensional vector. Below, a tokenization example is presented.

We use the following sentence for an illustration of the tokenization process:

"This is an illustration of BERT tokenizer.".

This sentence receives the following tokenization:

[*'[CLS]', 'this', 'is', 'an', 'illustration', 'of', 'bert', 'token', 'izer', '.', '[SEP]'*].

The corresponding token indices are then:

[101, 2023, 2003, 2019, 14614, 1997, 14324, 19204, 17629, 1012, 102].

3

Method

This chapter introduces the methods used in the project. Section 3.1 is a description of how the data were collected, cleaned and split into training and testing sets. Section 3.2 is an overview of descriptive statistics used for text data. Further, in Section 3.3 the method of classifying written and spoken data using BERT and the NB classifier is detailed. Lastly, in Section 3.4 the method of masked word predictions using BERT is described. The methods have been implemented using the programming language `Python` [25]. The implementation of the NB classifier has been carried out using the `Python` machine learning learning library `scikit-learn` [26]. For the implementations of BERT, the `transformers` library from Hugging Face [27] together with the `PyTorch` framework [28] from the `Torch` library has been used, specifically the pretrained models `BertModel` and `BertForMaskedLM`.

3.1 Data

3.1.1 Data Collection

Data have been obtained by an exhaustive online search for multiple data sets from a variety of different sources. The aim was to create a data set that is as large as possible, balanced between written and spoken text, and where each class should consist of texts from a diverse set of topics. For each class there are rules to ensure that the data are consistent with the aim of the project. Written text should not contain dialogue or quotes, as these can be considered a form of spoken text. Spoken text should be unscripted and hence forms of spoken language like speeches and monologues are considered scripted even though a speech or monologue may be unscripted. The written data consist of texts from publicly available English Wikibooks data set [29] and articles from different news papers in English [30] [31] [32]. The spoken data consists of transcripts from various podcasts on Spotify [33], transcripts from the radio shows This American Life [34] and NPR [35], and transcripts from the British National Corpus (BNC) [36]. The data set from Spotify is approved for use in this project but is not publicly available. This American Life, NPR and BNC are all publicly availaible.

3.1.2 Data Cleaning

The data were processed so that each observation is represented by a sentence, where a sentence is defined as a sequence of words that end with a punctuation mark, question mark or exclamation mark. Sentences that are too long or too short are removed, the minimum sentence length is set to six words and the maximum sentence length is set to 19 words. The limits are motivated by too short sentences not containing enough information and too long sentences mostly being written texts. A sentence is removed if it contains any special characters, parentheses or brackets. If a sentence contains any scientific formulae or mathematical expressions, it is also removed. Lastly, a written sentence is removed if it contains any quotation marks, which might indicate a dialogue. Table 3.1 is an overview of the data. The *text*-column contains each sentence, the *label*-column is 0 for a written sentence and 1 for a spoken sentence and the *source*-column contains the name of the original source of each sentence. For example, if *source*= *wiki* the sentence belongs to the Wikibooks data set.

text	label	source
The Justice Department has said it is working t...	0	finance
To learn how to overload and override a method,...	0	wiki
People making their own end of life plan may no...	0	NEWS
I thank you all so much for your thoughts, and ...	1	NPR_kaggle
Judith Moskowitz says these skills are widely a...	1	NPR_kaggle
Its one or the other I believe now.	1	Spotify1

Table 3.1: An example of how the data is formatted. The data consists of three columns, the sentence, the label and the source.

3.1.3 Data Splitting

Text data that originates from different sources can have their own unique vocabulary, occurrences of words, and possibly structure of sentences. This can induce source specific bias, meaning that a classifier learns to classify the source of the data rather than the class of the data. The *source* of the data can be seen as a subclass to the *class* of the data. For example, an observation from the New York Times belongs to the *class* of written data but its underlying *source* is the New York Times magazine. The data used in the project is collected from different sources with their own potentially unique vocabulary. Thus, the classifiers are at risk of source specific bias. To mitigate this the data is split so that the training and testing sets contain data from separate sources. That is, the models will be trained on a specific set of sources and tested on a different set of sources. In order to also reduce bias that emerge from the sample size differences of the sources, each source should be contributing with an equal number of observations. To make the this dataset large, only the four largest sources (in terms of observations) from each class, are used when sampling. From these datasets, a total of 16 different combinations of training and testing sets are sampled, where each combination has three sources from each class in the training set and one source from each class in the testing set. Since we have four sources in each class, a total of 16 combinations of training and testing sets are produced so that each source from each class will belong to the testing set together with all its possible combinations from the other class at least once. The number of observations sampled from each source is 259, 712, the size of the test sets are 519, 424 and the size of the training sets are 1, 558, 272. Table 3.2 is an overview of the sources used when splitting according to sources.

Written	**Spoken**
Wikibooks dataset	Spotify podcast
US Financial News Articles	This American Life
NYT Articles	British National Corpus
BBC News Summary	NPR

Table 3.2: The datasets used in the source split.

3.2 Descriptive Statistics

In order to provide an overview of the collected data and to check if the previously documented differences between the written and spoken language hold also in this case, we compute seven descriptive statistics. Based on the differences discussed in [3], average word length, average sentence length, and lexical variation are computed. Lexical variation is computed as average number of unique words per 1000 instances in 32 random samples of the data. For these three averages, the corresponding standard deviations are also computed. In [37] the author shows that there are more so called consciousness-of-projection (CoP) terms, self-reference (SR) terms, positive allness (PA), and negative allness (NA) terms in spoken language compared to written language. The definition of CoP terms are: "words which indicate that the observed is in part a function of the observer". In [2] the author argues that there are more negations in spoken language. We will include negations in NA terms. A list of words that represent each type of term can be seen in Table 3.3. The occurrences of the terms in Table 3.3 will be counted for all data belonging to both classes. Altogether, seven descriptive statistical features will be considered.

Kind of terms	**Terms**
Conscious-of-projections (CoP)	apparently, appear, appears, certainly, clearly, definitely, likely, may, maybe, might, obviously, perhaps, possibly, presumably, probably, seem, seemed, seemingly, seems, surely
Self-reference (SR)	I, me, my
Positive allness (PA)	all, every, always
Negative allness (NA)	neither, never, no, nobody, none, nor, not, nothing, nowhere

Table 3.3: The CoP-, SR-, PA-, and NA terms used.

3.3 Classifying written and spoken data

The task of classifying written and spoken data is a binary classification problem with two classes: *written* and *spoken*. The objective is to classify each sentence in our data as either of the two classes using both the NB classifier described in Section 3.3.1 and the BERT-model described in Section 3.3.2. Both models are trained and evaluated on all of the 16 combinations of data detailed in Section 3.1.3 and then compared to each other.

3.3.1 Naive Bayes classifier

The multinomial NB classifier is used as a baseline approach. The model uses unigrams as features to classify a sentence as either written or spoken. For example, the sentence '*my name is*' would consist of the features ['*my*','*name*','*is*']. The classifier takes numerical values as input. Therefore, the data is transformed into an $m \times n$ matrix Z of unigram counts, where m is the number of sentences and n is the number of unique unigrams. Each element z_{ij} in the matrix represents the count of unigram j in sentence i. We have two classes, written and spoken belonging to the set $C = \{written, spoken\}$. Let c represent an arbitrary class in our data. The probability of unigram j belonging to class c is denoted by π_{cj} and estimated by:

$$\hat{\pi}_{cj} = \frac{K_{cj} + \alpha}{K_c + \alpha N} \tag{3.1}$$

where:

- $K_{cj} = \sum_{z_{ij} \in c} z_{ij}$ is the total frequency of unigram j in sentences belonging to class c in the training data.
- $K_c = \sum_{j=1}^{N} K_{cj}$ is the sum of all counts for class c.
- N is the total number of unigrams.
- α is a smoothing parameter accounting for unigrams that are not present in the training data i.e preventing zero probabilities. The value of α is set to to 1.

In some cases, a unigram j might be absent for a specific class c in the training data. If a sentence in the validation data contains the absent unigram j, the probability that the sentence belongs to class c will be zero. The parameter α accounts for this by preventing any probability $\hat{\pi}_{cj}$ of being zero. If $\alpha = 1$ and unigram j did not appear in class c in the training data, we get the estimated probability

$$\hat{\pi}_{cj} = \frac{1}{K_c + N}, \tag{3.2}$$

and since K_c is approximately the same for both classes, a unigram is weighted equally for being absent in the written or the spoken class. Also considering the amount of data used in the project, a unigram which has not appeared in the training set should be heavily punished by a low probability.

Further, let y represent a sentence containing a fixed number of unigrams j. The probability that sentence y belongs to class c given the feature unigrams j in y is then estimated by:

$$P(y \in c | j \in y) = \hat{\theta}_c \prod_{j \in y} \hat{\pi}_{cj} \tag{3.3}$$

where $\hat{\theta}_c = \frac{\sum_{y_i \in c} y_i}{\sum y_i}$ is the fraction of sentences belonging to class c. In practice, 3.3 is transformed into log-probabilities to prevent underflow and the fraction of sentences that belong to either class is always balanced, so that $\hat{\theta}_c$ can be disregarded. A sentence y is then classified by:

$$\text{class}(y) = \arg\max_{\forall c \in C} \sum_{j \in y} \log(\hat{\pi}_{cj}) \tag{3.4}$$

The feature processing is done using the `CountVectorizer`-function from `scikit-learn` and the following steps:

1. Use the training data to define the vocabulary of which words to count.
2. From this, the training data is transformed into a matrix of word counts $X_{\text{train}} \in \mathbb{N}^{m \times n}$ using the tokenizer function `CountVectorizer()`, where m is the number of rows in training data and n is the number of words in the vocabulary. This is a sparse matrix with non-zero elements corresponding to word counts.
3. Given the vocabulary from the training set, the evaluation set is transformed into a matrix of word counts $X_{\text{val}} \in \mathbb{N}^{k \times n}$, where k is the number of observations in the evaluation set.
4. The matrix X_{train} is then used to calculate the probabilities (3.1) and X_{val} is predicted on.

3.3.2 BERT classifier

To transform the text data to a numerical representation BERT-tokenization was used, see Section 2.6.3. Since sequences in the data are of difference lengths, a padding procedure followed the tokenization. Since the index of the [PAD] token is 0, padding the indices to a specific length imply adding elements of zeroes to all sequences being shorter than the longest sequence in each batch.

The architecture and training procedure of the BERT classifiers in this project are mimicking that of the fine-tuning procedure described in the BERT article [22]. The models produced in this project that are partly initialized with the BERT pre-trained encoder stack will be referred to as BERT classifiers henceforth. The sequential model structure is presented in Table 3.4 and is identical for all 16 versions of evaluation data. The encoder stack takes a sequence as input and output the last hidden state corresponding to the [CLS] token. This vector is then passed through a dropout layer with $p = 0.1$, meaning that each weight is put to zero with probability p and then fed through a linear classification layer. The loss function used is the *cross entropy loss*. Since the cross entropy loss function applies a log-softmax, a layer of this kind is not necessary for this sequential model. The chosen values of hyperparameters are presented in Table 3.5. These are motivated by the recommendations in [22] and by the results from the initial test runs. Due to limited access to computational power, a comprehensive evaluation of hyperparameter choice has not been conducted.

The initialized weights in the encoder stack utilized in the BERT-models are those of $BERT_{\text{BASE}}$. The choice is motivated by the increase in computational power that $BERT_{\text{LARGE}}$ would imply. It's important to note that in the original BERT article [22], it is mentioned that *"We find that BERT$_{LARGE}$ significantly outperforms $BERT_{BASE}$ across all tasks, especially those with very little training data."*. The performances of the BERT-models presented in this report could therefore potentially be improved by using a larger model.

Layer	(Input dim), (Output dim)
EncoderStack	$(m,\, d_{\text{model}})$, $(1,\, d_{\text{model}})$
DropOut	-
Linear	$(1,\, d_{\text{model}})$, (1, 2)

Table 3.4: Layers in the BERT-model with corresponding input and output dimensions (excluding batch dimension). From top down, layers are written in the same sequential order as in the model structure. The variable m represents the length of the input sequence.

Hyperparameter	Value
Batch size	32
Learning rate	5e-5
Number of epochs	2
Decay	0.01
Dropout rate	0.1

Table 3.5: The hyperparameters that were used when training the BERT-models. The decay parameter is referring to degree of L2-regularisation.

Let $x \in \mathbb{R}^{m \times d_{\text{model}}}$ be an arbitrary embedded input sequence and let B be function representing the BERT encoder stack. Then $B(x) = H$, where $H = [h_0, h_1, ..., h_m]$ and $h_0 \in \mathbb{R}^{1 \times d_{\text{model}}}$ represent the last hidden state corresponding to the [CLS] token. The dropout layer can be interpreted as a vector $v \in \{0, 1\}^{1 \times d_{\text{model}}}$ where each element v_i is zero with probability 0.1. Let diag(v) be the diagonal $d_{\text{model}} \times d_{\text{model}}$ matrix whose entries are the elements of v. The last classification layer is defined by a weight matrix

$W \in \mathbb{R}^{d_{\text{model}} \times 2}$ and a bias vector $b \in \mathbb{R}^{1 \times 2}$ as follows

$$(h_0 \text{diag}(v))W + b = o, \tag{3.5}$$

where and $o \in \mathbb{R}^{1 \times 2}$ is the output vector.

3.4 Masked word predictions using BERT

Masked word prediction is a common NLP task that is used to evaluate a language model's ability to predict an intentionally masked word from a given sentence. Predicting masked words can be seen as an indicator of the models ability to apply contextualized meaning, since the information used to predict a masked word are the words surrounding the masked word. To investigate BERT's performance on applying contextualized meaning on written versus spoken language, masked word predictions are implemented in two different ways. Firstly by randomly masking one word per sentence and secondly by masking specific words based on frequency.

The standard output of BERT is a tensor of size [`batch`, `max length`, `hidden states`]. In this case, `batch` refers to the number of sentences that BERT receives as input, `max length` is the number of tokens per sentence and `hidden states` is the size of the output vector for each token, the standard is 768 hidden states. For example, if BERT receives 5 sentences split into 20 tokens per sentence it will output a tensor of size $[5, 20, 768]$. To fine-tune BERT for masked word predictions, the last hidden states of BERT is passed through a feed forward layer. The feed forward layer takes the hidden state vector for each token in a sentence and outputs a logit vector for each token. The logit vectors are the same size as the BERT vocabulary (30,522 tokens) and each value of the logit vector can be viewed as BERT's belief that the corresponding token is the correct one. Each logit vector is then passed through a softmax layer to produce a probability distribution and the token corresponding to the maximum value of the distribution is taken as BERT's prediction. Figure 3.1 is a flowchart illustrating the masked word prediction process and Figure 3.2 shows the process after obtaining the logit vectors from the fine-tuned BERT model.

3.4.1 Random selection

The first version of masked word prediction is to randomly mask one word per sentence for every sentence in both written and spoken data. The accuracy of BERT's predictions on both spoken and written data are then compared. This can be seen as a general assessment of BERT's performance on written versus spoken text. Due to the computational demands of BERT, a random sample of 100,000 sentences is used when randomly masking one word per sentence.

3.4.2 Frequency-based selection

The second version is to mask a specific set of words based on their frequency. The idea is to mask a set of words that are simultaneously common for one class and uncommon

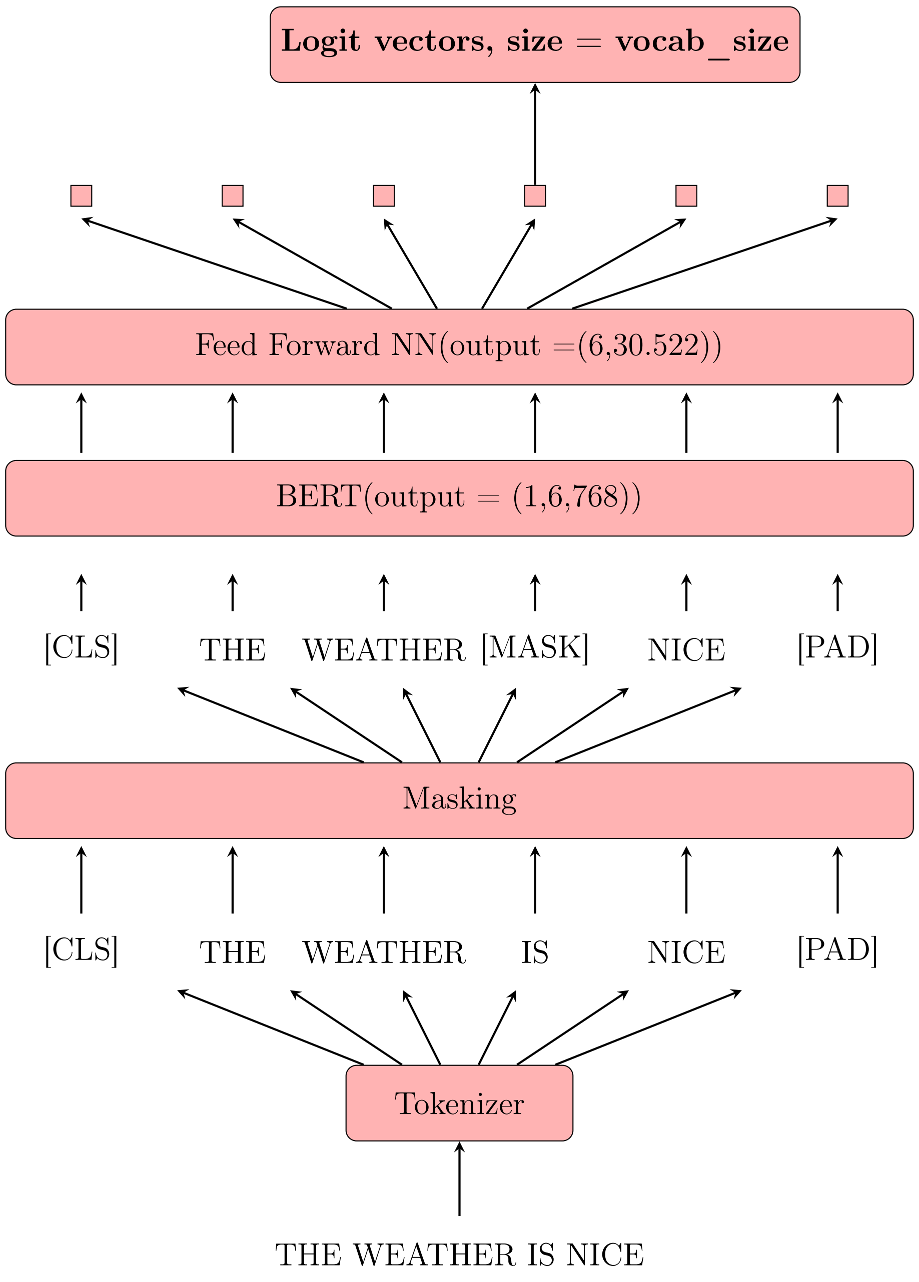

Figure 3.1: Flowchart of using BERT for masked word prediction.

for the other and then compare BERT's performance in terms of how it predicts these words. For example, if BERT is significantly better at predicting typical written words compared to typical spoken words or vice versa, that could be an indication of BERT being worse at interpreting context in sentences from either class.

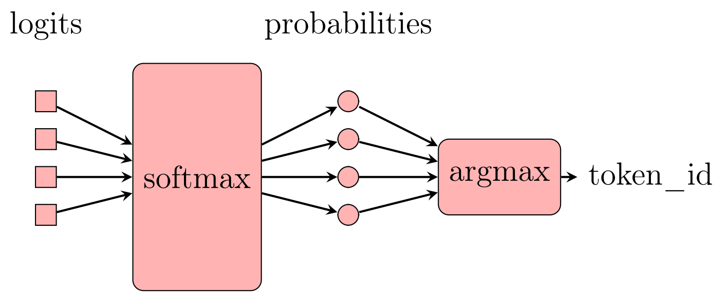

Figure 3.2: Flowchart of using BERT for masked word prediction.

4

Results

The following chapter presents the results obtained with the methods in Chapter 3. Section 4.1 contains the results of the descriptive statistics used to generally compare written and spoken text. Section 4.2 presents the classification results from both the naive Bayes classifier and the BERT classifier in differentiating between spoken and written sentences. Further, Section 4.3 contains the results from masked word predictions using the fine-tuned BERT-model.

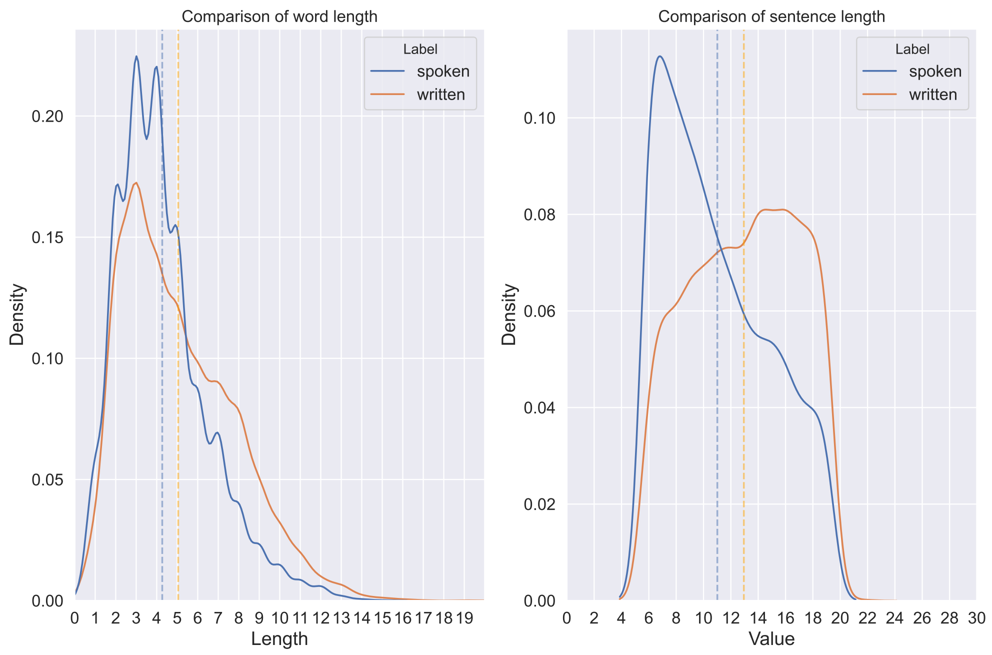

Figure 4.1: The left panel shows the distributions of word lengths for both classes whereas the right panel shows the distributions of sentence lengths for both classes. Each of the four distributions is based on a random sample of 5,000 observations.

4.1 Data

In this section, we present the results of the descriptive statistical analysis of the two text type classes *written* and *spoken*. When comparing the columns in Table 4.1, it is clear that the two classes differ in all of the seven features. The result implies that the differences between the two classes in our data coincide with the previously documented differences between written and spoken language (referenced in Section 3.2). The table shows that the written data have longer words and longer sentences on average, in addition to a higher lexical variation (see 3.1). The CoP-, SR-, PA-, and NA terms all have higher occurrences in the spoken data compared to the written data. Figure 4.1 illustrates the distributions of word lengths and sentence lengths for the written and spoken data. It is clear that the distributions of sentence lengths differ between the classes. The figure shows that there is a higher proportion of written sentences that are above a length of 12 words compared to spoken sentences. There also seems to be a high proportion of spoken sentences that are of lengths of around seven words. The distributions of word lengths are quite similar for the two classes.

Feature	Written	Spoken
Average word length	5.05 (2.85)	4.27 (2.22)
Average sentence length	12.95 (3.93)	11.01 (3.89)
Lexical variation	4674 (67)	2567 (53)
CoP	31731	38538
SR	13530	63188
PA	27365	52661
NA	70982	81231

Table 4.1: The descriptive statistics for the two classes written and spoken. From top down the features are: *average word length, average sentence length, lexical variation per 1000 instances, number of Consciousness-of-projection terms, number of self-reference terms, number of positive allness terms* and *number of negative allness (and negation) terms*. The values in the parentheses are the corresponding standard deviations.

4.2 Differentiating between spoken and written texts

In this section, we present the results of the BERT classifier and compare them with the results of the NB classifier in the task of differentiating between written and spoken texts. Figures 4.2, 4.3, and 4.4 in this section present the results of the BERT classifier from a random sample of size $100,000$, drawn from the 16 evaluation sets. The three figures show the relationship between certainty and sentence length, self-reference terms, and positive allness terms, respectively. Here certainty refers to BERT's predicted probability of an observation belonging to a specific class. These features are chosen because they show the strongest trends that correspond to the descriptive statistical differences presented in Table 4.1. For the features that were not chosen, there were no visible differences between the two classes. For Figures 4.2, 4.3, and 4.4, the y-axis' on the left (*prob0*) represents the probability of an observation being written, and on the right (*prob1*), represents the probability of an observation being spoken. Written data are plotted with probabilities of being written and likewise, spoken data are plotted with probabilities of being spoken. This means that the probabilities are the BERT models' confidence of observations belonging to a specific class given that they actually belong to that class. The boxes display the interquartile range (IQR). The IQR covers the middle half of the distribution, meaning that the middle 50% of values are inside the box, and the upper 25% and lower 25% of values are outside the box. The horizontal line in the box represents the median. The whiskers represent the outlier boundary of 1.5 times IQR.

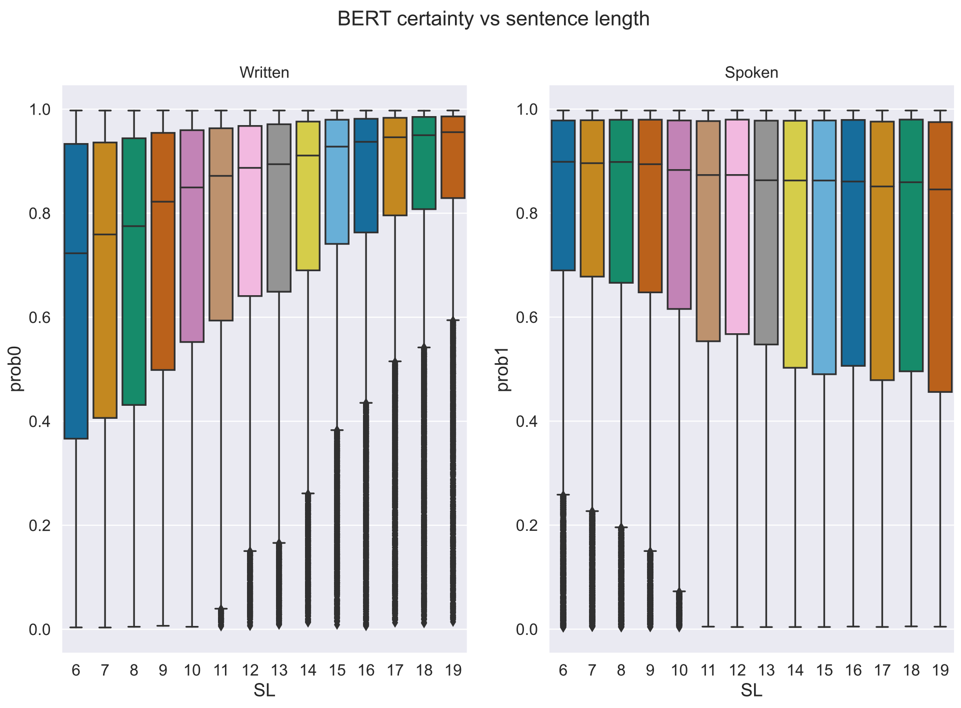

Figure 4.2: The left panel shows the relationship between BERT's predicted probabilities of observations being written, given that they are written (y-axis) and the length of the sentences (x-axis). The right panel shows the relationship between BERT's predicted probabilities of observations being spoken, given that they are spoken (y-axis) and the length of the sentences (x-axis). The boxes represent the IQR with the horizontal line on each box representing the median. The whiskers end at 1.5 times IQR. The black dots outside the whiskers are outliers. This is calculated on a random sample of 100, 000 observations.

Figure 4.2 shows that for a written observation, SL can influence the certainty of predictions in such a way that longer sentence length increases the models' certainty of an observation to be written. For spoken data, sentence length has the opposite effect, the longer the sentence length, the lower the certainty. The influence of sentence length is significantly stronger for written data than for spoken. The trend for written data is shown by an increasing median and a decreasing spread as sentence length increases, and the opposite is seen in spoken data. In Figure 4.3, the relationship between the certainty of predictions and the number of SR terms is shown in two box plots. For written data, a trend towards lower certainty for an increasing number of SR terms is obvious. The green box, representing two SR terms, shows an IQR below a probability of 0.5 meaning that more than a fraction of 0.75 of the written observations containing two SR terms are incorrectly classified as spoken. One can observe similar indications for written observations containing one SR term. Analogously, spoken data containing one or two SR terms are related to high certainty. The relationship between

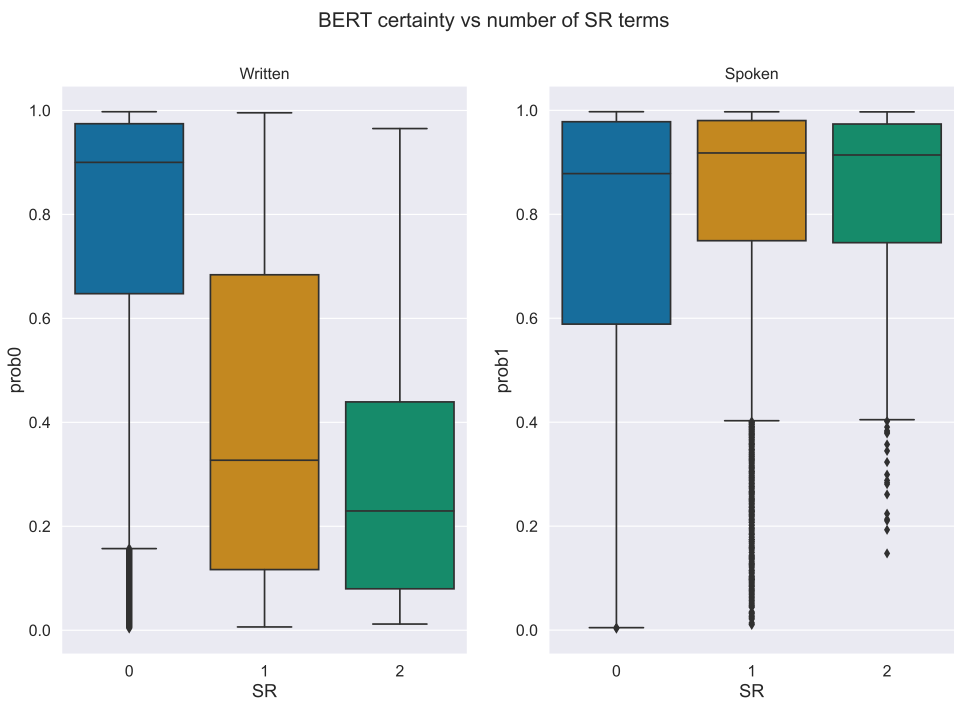

Figure 4.3: A box plot illustrating the certainty in the BERT model's predictions (y-axis) for sentences containing different numbers of self-reference (SR) terms (x-axis) calculated on a random sample of $100,000$ observations. The upper limit of the x-axes is two since there are no observations containing more than two SR terms. The boxes show the median and the IQR. The whisker represents the outlier boundary of 1.5 times IQR.

certainty and PA terms can be seen in Figure 4.4. For written data, a higher number of PA terms indicates a lower certainty in predictions. For spoken data two PA terms affect the certainty negatively. The difference in the certainty of predictions in relation to the number of PA terms between spoken and written data is not large.

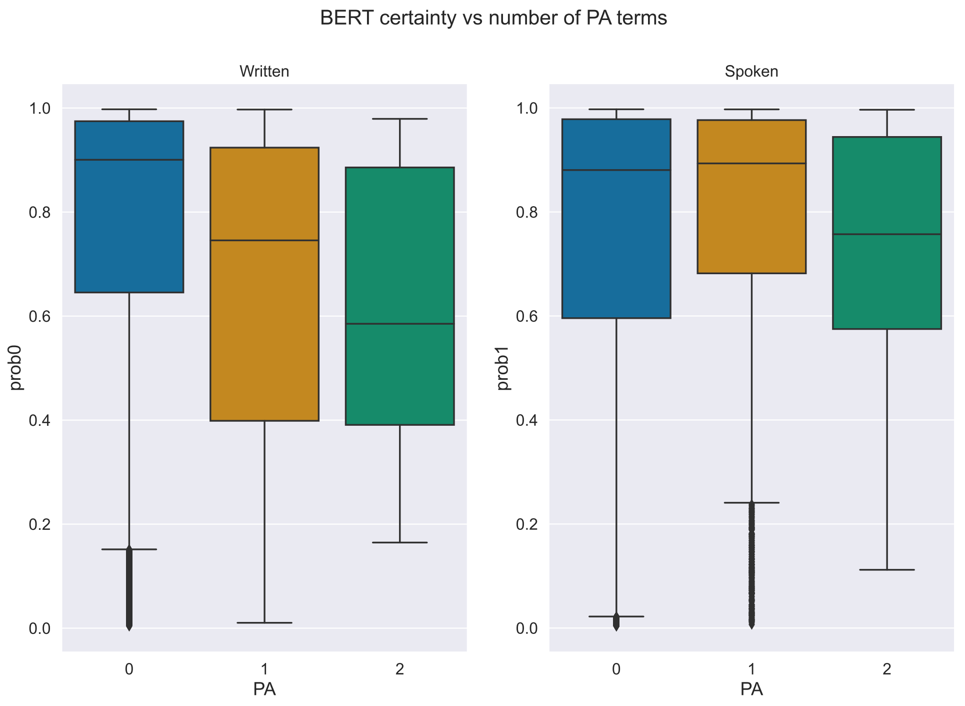

Figure 4.4: A box plot illustrating the certainty in the BERT model's predictions for sentences containing different numbers of positive allness (PA) terms calculated on a random sample of $100,000$ observations.

The accuracy of the BERT model and the NB model are compared in Figure 4.5. The figure shows that the BERT model gives higher overall accuracy than the NB model. The accuracy for written data is higher for BERT than for NB. Contrarily, for spoken data, NB achieves higher accuracy in comparison to BERT. The result for each evaluation set is presented in Table A.1 and Table A.2.

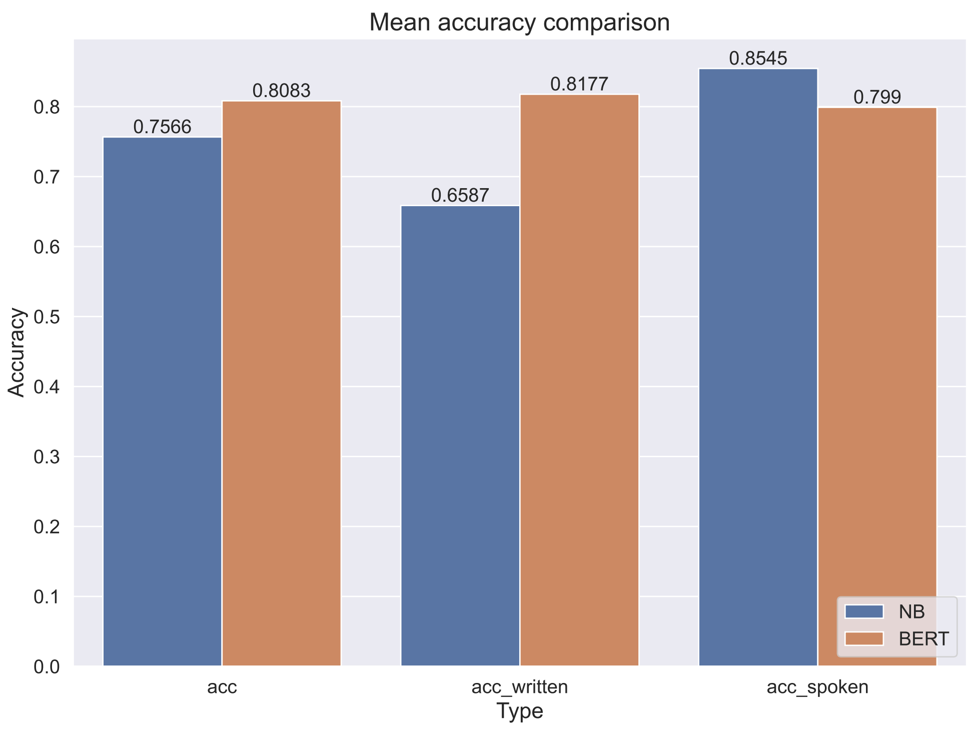

Figure 4.5: A mean accuracy comparison between NB and BERT models. It is the mean value over the 16 evaluation sets. From the left: the first, second, and the third pair of bars represent the overall accuracy, accuracy on the written data, and accuracy on the spoken data, respectively.

Two confusion matrices are presented in Figure 4.6, where the left is representing the results of the BERT model, and the right is the result of the NB model. The BERT model does not appear strongly biased towards either of the classes. There are marginally more false negatives than false positives and also slightly more true negatives compared to true positives. The NB model is biased towards the spoken class, as a fraction of $0.177 + 0.4275 = 0.6045$ of the evaluation data are predicted as spoken, with all the test sets being balanced.

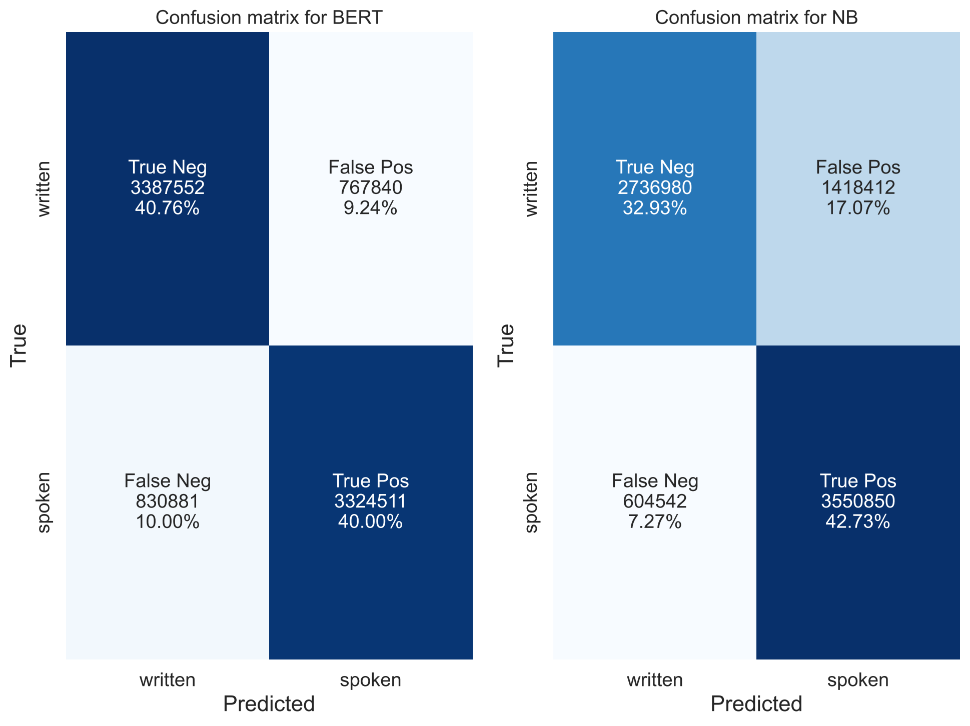

Figure 4.6: Confusion matrix for all evaluation sets when predicting with the BERT classifier (left) and predicting with NB classifier (right).

4.3 Masked word predictions

This section presents the results from BERT's masked word predictions. Due to the computational demands of BERT, a random sample of 100,000 sentences (25,000 sentences per source) per class has been drawn when using random selection. When using frequency-based selection, the total number of sentences containing the masked word are used.

There are six words that belong to the top 100 unigrams in the spoken dataset that do not belong to the top 500 unigrams in the written dataset. Similarly, there are three words that belong to the top 100 unigrams in the written dataset that do not belong to the top 500 unigrams in the spoken dataset. These words are shown in Table A.3 and A.4 respectively. The words "yeah" and "mean" are ranked 40 and 83 respectively in the spoken data. Similarly the words "million" and "company" are ranked 66 and 75 in the the written data. Therefore, these words are chosen to be masked in frequency-based selection. It is important to note that the masking procedure is done per class, meaning that the words "yeah" and "mean" are masked in the spoken data and similarly the words "million" and "company" are masked in the written data.

The accuracies of BERT's masked word predictions can be seen in Table 4.2. When using random selection, there is no significant difference in accuracy between written and spoken data. The frequency-based selection shows a big difference in accuracy. Predictions on the written dataset achieve an overall accuracy of 55 % where the word "million" is accurately predicted 59% of the time and the word "company" is accurately predicted 51% of the time. The predictions on the spoken dataset achieve an overall accuracy of 16% with the word "mean" accurately predicted 21% of the time and the word "yeah" accurately predicted 5.7% of the time. Furthermore, Figure 4.7 shows the distributions of BERT's classification probabilities for both the random- and frequency-based selection. The distributions for random selection are very similar and there is no indication of a systematic difference in BERT's predictive abilities on either class. The distributions for frequency-based selection show that the written distribution is skewed towards higher probabilities and the spoken distribution is skewed towards lower probabilities.

Accuracy	**Spoken**	**Written**
Random selection	0.5132	0.5166
Frequency-based selection	0.1618	0.5565

Table 4.2: Accuracies from the masked word predictions.

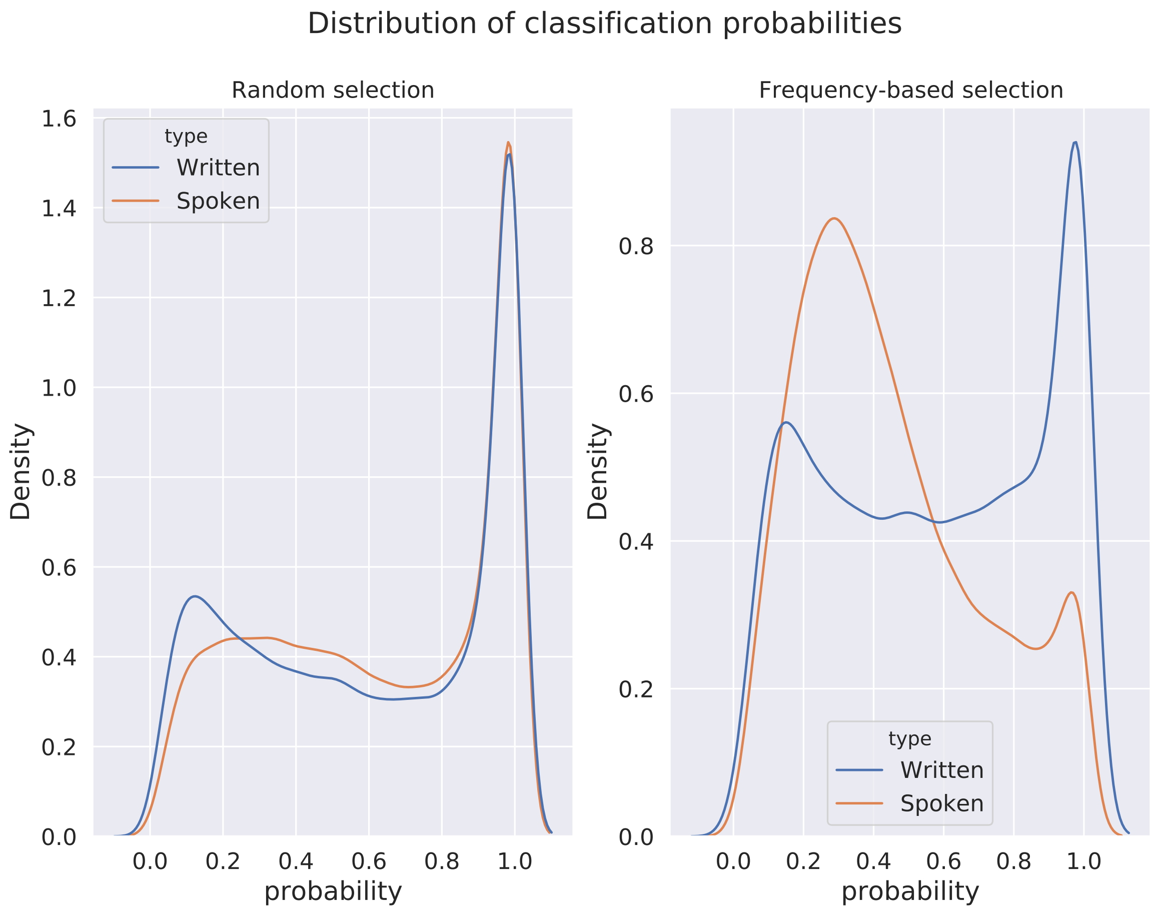

Figure 4.7: Distribution of classification probabilities. The left plot shows the distribution of BERT's classification probabilities when using random selection. The right plot shows the distribution of BERT's classifications probabilities using frequency-based selection.

5
Discussion

In this chapter, we discuss the results shown in the previous chapter and present thoughts and suggestions for future work.

5.1 The results of this project

A very interesting and surprising result is that the simple naive Bayes classifier outperforms BERT in classifying spoken data (see figure 4.5). This might be due to the fact that BERT has been primarily trained on written text. Conversely, the naive Bayes classifier is considerably worse at classifying written sentences compared to BERT. A suggestion is that the naive Bayes classifier is better at identifying a given sentence as spoken because spoken texts have a lower lexical variation. This will not affect BERT in the same way because BERT does not use word counts to classify sentences. However, the naive Bayes classifier is considerably worse at classifying written sentences compared to BERT. This might similarly suggest that the naive Bayes classifier is worse at identifying a given sentence as written because written texts have a higher lexical variation. The masked word prediction task gave mixed results. When randomly masking one word per sentence BERT manages to accurately predict the correct word approximately fifty percent of the time for both classes. Conversely, when masking a specific word based on the frequency of the word in both classes, there is a significant difference in BERT's performance (Table 4.2). Specifically, BERT seems to be better at predicting words that more often occur in the written data compared to words that more often occur in the spoken data. This result highlights the fact that a machine learning model is only as good as the data it has been trained on. Since BERT has primarily been trained on written data, the words that seldom occur in the written language seem to be difficult for BERT to predict.

The descriptive statistical analysis of the differences between written and spoken language in the project, coincides with previous research on their differences [2][3][4]. This is an indication that the collected data are representative of the spoken and written class. Both the statistical analysis and the classifier's ability to differentiate between the two classes strongly suggest that the distributions of the spoken and written data are in fact different.

5.2 Flaws and limitations

In order to produce balanced data sets, limitations regarding the number of observations per data source reduced the amount of possible training data. Since all sources contributes with the same number of observations, this limit was decided by the number of sentences in the smallest source specific data set.

Due to computational limitations, the project only looked at a few examples of masking specific words based on the frequency of their occurrences. The results indicate that BERT could be worse at predicting words exclusively common in the spoken language. However, a more extensive analysis in this area could provide additional information, perhaps by choosing a broader set of words to mask and further looking at how BERT's performance varies as a function of the frequency of the word in the pre-training data.

Furthermore, the choice of maximum sentence length was chosen quite arbitrarily and has possibly affected the results when classifying written and spoken sentences. Written sentences are on average longer than spoken sentences, and so limiting the maximum sentence length mostly affected our written sentences. This could possibly have made the differentiation between classes harder. However, the objective was to investigate whether it is possible for BERT to distinguish between written and spoken sentences from a semantic and syntactic perspective, and thus not limiting sentence length could perhaps have made the task too easy.

5.3 Future suggestions and considerations

The results of this project show that the simple naive Bayes classifier is better at classifying spoken data compared to BERT. However, if BERT was to be re-trained on a corpus also consisting of a large amount of spoken data the results might be different. Thus, a future suggestion is to re-train BERT with added spoken text and repeat the experiment. Further, the masked word prediction was a task designed to measure the difference between BERT's understanding of written and spoken text. To further investigate this, a sentiment analysis could provide added insight. A suggestion is to test BERT's ability to distinguish between positive and negative sentiments on data where classes contain the same amount of spoken and written texts. If the written texts are correctly classified to a greater extent in comparison to the spoken, this could be an indication that BERT's ability to understand spoken language is worse than written language. Lastly, it could be interesting to look at models similar to BERT and see if they perform in a similar fashion. This could gain further insight into whether the lack of spoken data present in language model training is a systematic issue.

5.4 Conclusion

This project has investigated the performance of the language model BERT in comparison to a naive Bayes classifier. The results have shown that the simple naive

Bayes classifier outperforms the more complex BERT model when classifying spoken data. When classifying spoken data BERT achieves an accuracy of 79.9% while the naive Bayes classifier achieves an accuracy of 85.5% (see Figure 4.5). The full table of classification accuracies for each evaluation set can be found in Tables A.1 and A.2. Furthermore, the results from the masked word predictions indicate that BERT might be worse at predicting words that are typical in spoken language but not commonly used in written language. When predicting masked written words, BERT achieves an accuracy of 55% and when predicting masked spoken words BERT achieves an accuracy of 16% (see Table 4.2).In general, the results of this project suggest that it is necessary to further investigate the implications of applying pre-trained language models on spoken data.

References

1. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv* **abs/1810.04805** (2019).
2. Biber, D. *Variation across Speech and Writing* (Cambridge University Press, 1988).
3. Drieman, G. Differences between written and spoken language: An exploratory study. *Acta Psychologica* **20,** 36–57 (1962).
4. Akinnaso, F. On The Differences Between Spoken and Written Language. *Language and Speech* **vol.25, no.2, pp. 97-125** (1982).
5. Wahde, M. & Virgolin, M. Conversational Agents: Theory and Applications. *Handbook of Computer Learning and Intelligence* **1** (2022).
6. Yoo, S. & Jeong, O. An intelligent chatbot utilizing BERT model and knowledge graph. *Journal of Society for e-Business Studies* **24** (2020).
7. Chen, Q., Zhuo, Z. & Wang, W. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909* (2019).
8. Jain, A., Rouhe, A., Grönroos, S.-A., Kurimo, M., *et al. Finnish ASR with Deep Transformer Models.* in *Interspeech* (2020), 3630–3634.
9. Lewis, D. D. *Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval* in *ECML* (1998).
10. Rusland, N. F., Wahid, N., Kasim, S. & Hafit, H. Analysis of Naive Bayes Algorithm for Email Spam Filtering across Multiple Datasets. *IOP Conf. Ser.:Mater.Sci.Eng* **226 012091.** `https://iopscience.iop.org/article/10.1088/1757-899X/226/1/012091/meta` (2017).
11. Narayan, V. & Bhatia, A. Fast and Accurate Sentiment Classification Using an Enhanced Naive Bayes Model. *Intelligent Data Engineering and Automated Learning - IDEAL 2013. Lecture Notes in Computer Science* **8206.** `https://doi.org/10.1007/978-3-642-41278-3_24` (2013).
12. Lee, S., Ji, H., Kim, J. & Park, E. What books will be your bestseller? A machine learning approach with Amazon Kindle. *The Electronic Library* **39.** `https://www.emerald.com/insight/content/doi/10.1108/EL-08-2020-0234/full/html` (2021).
13. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* `http://www.deeplearningbook.org` (MIT Press, 2016).

14. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning Representations by Back-propagating Errors. *Nature* **323,** 533–536. `http://www.nature.com/articles/323533a0` (1986).
15. Goldberg, Y. & Hirst, G. *Neural Network Methods in Natural Language Processing* ISBN: 1627052984 (Morgan amp; Claypool Publishers, 2017).
16. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. *Attention Is All You Need* 2017. arXiv: `1706.03762 [cs.CL]`.
17. Radford, A. & Narasimhan, K. *Improving Language Understanding by Generative Pre-Training* in (2018).
18. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R. & Le, Q. V. *XLNet: Generalized Autoregressive Pretraining for Language Understanding* in *Advances in Neural Information Processing Systems* (eds Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. & Garnett, R.) **32** (Curran Associates, Inc., 2019). `https://proceedings.neurips.cc/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf`.
19. Wu, Y. *et al.* Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR* **abs/1609.08144.** arXiv: `1609.08144`. `http://arxiv.org/abs/1609.08144` (2016).
20. Luong, M., Pham, H. & Manning, C. D. Effective Approaches to Attention-based Neural Machine Translation. *CoRR* **abs/1508.04025.** arXiv: `1508.04025`. `http://arxiv.org/abs/1508.04025` (2015).
21. Ba, J. L., Kiros, J. R. & Hinton, G. E. *Layer Normalization* 2016. `https://arxiv.org/abs/1607.06450`.
22. Devlin, J., Chang, M., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* **abs/1810.04805.** arXiv: `1810.04805`. `http://arxiv.org/abs/1810.04805` (2018).
23. Zhu, Y., Kiros, R., Zemel, R. S., Salakhutdinov, R., Urtasun, R., Torralba, A. & Fidler, S. Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. *CoRR* **abs/1506.06724.** arXiv: `1506.06724`. `http://arxiv.org/abs/1506.06724` (2015).
24. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. & Bowman, S. R. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *CoRR* **abs/1804.07461.** arXiv: `1804.07461`. `http://arxiv.org/abs/1804.07461` (2018).
25. Van Rossum, G. & Drake, F. L. *Python 3 Reference Manual* ISBN: 1441412697 (CreateSpace, Scotts Valley, CA, 2009).
26. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011).
27. Wolf, T. *et al.* HuggingFace's Transformers: State-of-the-art Natural Language Processing. *CoRR* **abs/1910.03771.** arXiv: `1910.03771`. `http://arxiv.org/abs/1910.03771` (2019).
28. Paszke, A. *et al.* in *Advances in Neural Information Processing Systems 32* 8024–8035 (Curran Associates, Inc., 2019). `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

29. *Wikibooks dataset* November 2021. https://www.kaggle.com/datasets/dhruvildave/wikibooks-dataset.
30. *NYT articles* November 2021. https://www.kaggle.com/datasets/tumanovalexander/nyt-articles-data.
31. Greene, D. & Cunningham, P. *Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering* in *Proc. 23rd International Conference on Machine learning (ICML'06)* (ACM Press, 2006), 377–384.
32. *Financial Articles* November 2021. https://www.kaggle.com/datasets/jeet2016/us-financial-news-articles.
33. Clifton, A. *et al. 100,000 Podcasts: A Spoken English Document Corpus* in *Proceedings of the 28th International Conference on Computational Linguistics* (International Committee on Computational Linguistics, Barcelona, Spain (Online), Dec. 2020), 5903–5917. https://www.aclweb.org/%20anthology/2020.coling-main.519.
34. Mao, H. H., Li, S., McAuley, J. J. & Cottrell, G. W. *Speech Recognition and Multi-Speaker Diarization of Long Conversations* in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Shanghai, China, 25-29 October 2020* (2020). https://arxiv.org/abs/2005.08072.
35. Majumder, B. P., Li, S., Ni, J. & McAuley, J. J. Interview: A Large-Scale Open-Source Corpus of Media Dialog. *CoRR* **abs/2004.03090.** arXiv: 2004.03090. https://arxiv.org/abs/2004.03090 (2020).
36. Consortium, B. *British National Corpus, XML edition* Oxford Text Archive. 2007. http://hdl.handle.net/20.500.12024/2554.
37. DeVito, J. A. Psychogrammatical factors in oral and written discourse by skilled communicators. *Speech Monographs* **33,** 73–76. eprint: https://doi.org/10.1080/03637756609375483. https://doi.org/10.1080/03637756609375483 (1966).

A

Appendix 1

Evaluation set	acc	acc_written	acc_spoken
wiki, This Amercian Life	0.728915	0.596234	0.861597
wiki, NPR_kaggle	0.662311	0.648846	0.675775
wiki, BNC	0.755102	0.597011	0.913192
wiki, Spotify1	0.750212	0.535466	0.964957
NEWS, This Amercian Life	0.708273	0.521805	0.894741
NEWS, NPR_kaggle	0.634280	0.561988	0.706571
NEWS, BNC	0.695432	0.478800	0.912064
NEWS Spotify1	0.717680	0.459994	0.975365
This Amercian Life, NYT	0.812377	0.749415	0.875339
NPR_kaggle, NYT	0.728022	0.782082	0.673962
NYT, BNC	0.787384	0.685752	0.889016
NYT, Spotify1	0.821514	0.675856	0.967171
This Amercian Life, finance	0.827784	0.810817	0.844751
NPR_kaggle, finance	0.755394	0.847381	0.663408
finance, BNC	0.842350	0.796032	0.888669
finance, Spotify1	0.878361	0.791042	0.965681
Average	0.756587	0.658658	0.854516

Table A.1: Results for the naive Bayes model on each evaluation set.

Evaluation set	acc	acc_written	acc_spoken
wiki, This Amercian Life	0.817681	0.826042	0.809320
wiki, NPR_kaggle	0.774123	0.872178	0.676068
wiki, BNC	0.791715	0.723721	0.859710
wiki, Spotify1	0.809861	0.730459	0.889262
NEWS, This Amercian Life	0.795269	0.735222	0.855317
NEWS, NPR_kaggle	0.734042	0.716883	0.751201
NEWS, BNC	0.746615	0.693306	0.799925
NEWS, Spotify1	0.796255	0.666357	0.926153
This Amercian Life, NYT	0.846786	0.854427	0.839145
NPR_kaggle, NYT	0.783913	0.940673	0.627152
NYT, BNC	0.810157	0.853584	0.766730
NYT, Spotify1	0.845259	0.871288	0.819231
This Amercian Life, finance	0.856874	0.892631	0.821117
NPR_kaggle, finance	0.817898	0.914902	0.720895
finance, BNC	0.823928	0.899804	0.748052
finance, Spotify1	0.883207	0.892242	0.874172
Averages	0.808349	0.8177325	0.798966

Table A.2: Accuracy, accuracy of spoken class, and accuracy of written class, for the BERT model on each evaluation set along with the averages of each column.

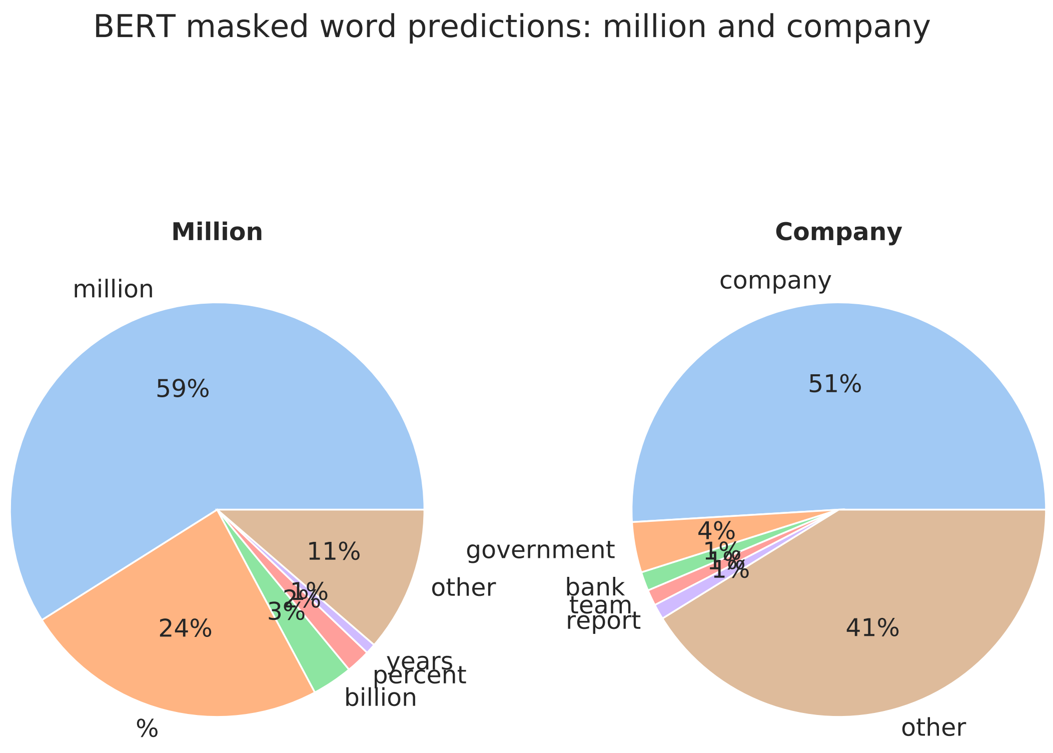

Figure A.1: The pie charts show the percentage of words being predicted as the masked word when masking the words "million" and "company".

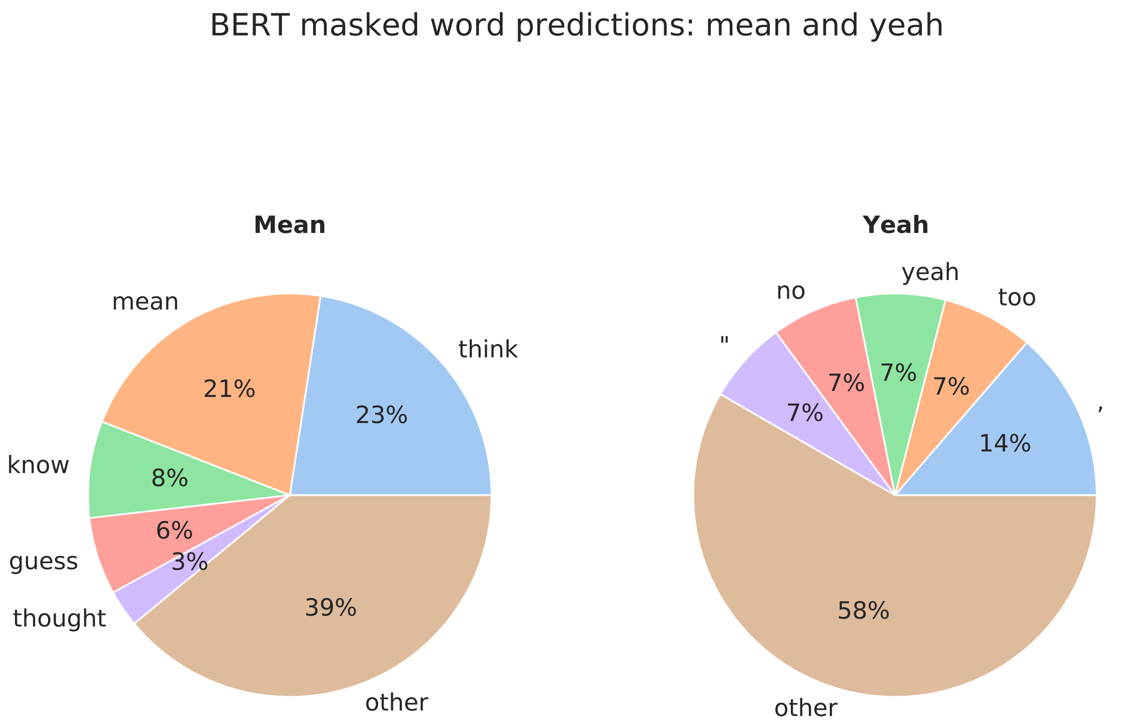

Figure A.2: The pie charts show the percentage of words being predicted as the masked word when masking the words "mean" and "yeah".

Word	Spoken rank	Spoken count	Written rank	Written count
yeah	40	40752	6336	195
mean	83	23375	920	1772
re	85	23119	7344	158
oh	91	20754	3400	443
er	94	18018	5363	245
hes	100	16591	592	2593

Table A.3: Common words spoken dataset.

Word	Written rank	Written count	Spoken rank	Spoken count
million	66	17009	551	1900
company	76	14937	516	2078
information	91	12260	638	1577

Table A.4: Common words written dataset.