Embedding recognized speech in a multilingual environment

H Heyns

orcid.org/ 0000-0002-0802-5005

Dissertation accepted in fulfilment of the requirements for the degree *Master of Science in Engineering Sciences with Computer and Electronic Engineering* at the North West University

Supervisor: Prof E Barnard

Graduation: June 2021

Student number: 26144042

Declaration

I, Austine Heyns, hereby declare that the dissertation entitled "Embedding recognized speech in a multilingual environment" is my own original work and has not already been submitted to any other university or institution for examination.

A. Heyns

Student number: 26144042

Signed on the 20th day of November 2020 at Potchefstroom.

Acknowledgements

I thank the Novus Group for providing the data used in this study. I also thank Dr. Charl van Heerden, CEO of Saigen, for his contributions and providing the speech-recognition results of the data.

I would also like to thank:

1. Ulrike Janke for taking care of all the administrative work that went along with this study, as well as the advice and support in all related matters.

2. My supervisor, Prof Etienne Barnard. Thank you for your guidance and encouragement. Your knowledge is a true inspiration.

3. My parents for their love, support and giving me the opportunity to further my studies. Through your example you have cultivated within me the love of learning.

4. My fiancé, Wicus. Thank you for the interest that you show in everything I do and thank you for inspiring and broadening my own interests.

Abstract

Word embeddings are widely used in natural language processing tasks. Most work on word embeddings focuses on monolingual languages with large available datasets. For embeddings to be useful in a multilingual environment, as in South Africa, the training techniques have to be adjusted to cater for a) multiple languages, b) smaller datasets and c) the occurrence of code-switching. One of the biggest roadblocks is to obtain datasets that include examples of natural code-switching, since code switching is generally avoided in written material. A solution to this problem is to use speech recognised data. Embedding packages such as Word2Vec and GloVe have default hyper-parameter settings that are usually optimised for training on large datasets and evaluation on analogy tasks. When using embeddings for problems such as text classification in our multilingual environment, the hyper-parameters have to be optimised for the specific data and task. We investigate the importance of optimising relevant hyper-parameters for training word embeddings with speech recognised data, where code-switching occurs, and evaluate against the real-world problem of classifying radio and television recordings with code switching. In this dissertation we present findings on the application of word embeddings to recognised speech in a multilingual environment.

Keywords: *Word embeddings, speech recognition, code-switching, multilingual environment*

Contents

List of Figures

List of Tables

List of Acronyms

NLP Natural Language Processing

GloVe Global Vectors for Word Representation

NER Named Entity Recognition

BoW Bag of Words

CBoW Continuous Bag of Words

ELMo Embeddings from Language Models

BERT Bidirectional Encoder Representations from Transformers

ML Matrix Language

M-Index Multilingual Index

CMI Code-mixing Index

I-Index Integration Index

POS-tagging Part of Speech tagging

SVM Support Vector Machine

MSE Mean Square Error

TF-IDF Term Frequency–Inverse Document Frequency

MNB Multinomial Naive Bayes

AUC Area Under the Curve

ROC Receiver Operating Characteristics

PR Precision-recall

Chapter 1

Introduction

In this chapter we introduce the study and discuss why it is relevant. We will define the scope, research questions and the objectives.

1.1 Background

Word embeddings are popularly used to address natural language processing (NLP) tasks (e.g., language identification, named entity recognition, sentiment analysis etc.) because they encode many semantic relationships between words. Word embeddings can be trained on unsupervised data while conventional methods to train NLP tasks require large annotated datasets. This is particularly relevant for low-resource languages, as they generally have only small, and often no, annotated datasets available. For word embeddings to perform well, a large training dataset is nevertheless necessary in order to identify word pairs that co-occur in the data. When the dataset is too small, the co-occurrence of words is often missed and will result in a sparse co-occurrence matrix [2]. Leading approaches such as Global Vectors for Word Representation (GloVe) and Skip-gram struggle to deal with a sparse co-occurrence matrix. Thus, word embeddings are a significant advance over conventional NLP methods, given they do not require large annotated datasets, but their application to low-resource languages remains a challenge.

There are 11 official languages in South Africa and many other languages are also spoken. Many South Africans are multilingual and will often switch between languages in speech. This phenomenon does not occur in written work as often; therefore, it is hard to get access to textual data that includes natural code-switching. As a solution to the lack of available data we propose is to use speech recognition systems on South African data - for example radio and television broadcasts. Named entities are shared across languages: for example, names of people, products, companies or places. These names are often used in broadcasts and will be beneficial when gathering data that represents code-switching.

Some NLP tasks have specific constraints that provide us with interesting research problems when multilingual word embeddings are employed. These NLP tasks include sentiment analysis and named entity recognition (NER). Training embeddings for sentiment analysis is a problem because polar opposite words such as good and bad will have similar embeddings but opposite sentiment [3]. Thus, ideally embeddings should be able to identify the characteristics of both the semantics and the sentiment of a word. NER can be a problem in a multilingual environment because words that can be classified as named entities in one language might be classified as adjectives in another language. For example, common South African personal names such as Precious or Lucky might be tagged as adjectives in English and would not be seen as named entities.

1.2 Problem statement

Word embeddings are usually trained on large language-specific datasets. Most research related to word embeddings has utilised large English datasets. These datasets do not take into account the phenomena of code-switching and intensive borrowing that often occur in the speech of multilingual speakers. Code-switching makes it difficult to determine what language a specific word in the speech belongs to. These words would often be transcribed incorrectly and thus make it difficult to determine the embedding for the word. Because of these factors, there are no trained embeddings for South African languages that are widely available. South African languages do not have large enough datasets that represent the occurrence of code-switching, as written work usually avoids code-switching, while code-switching is a common occurrence in spoken dialogue. Our goal is to investigate different approaches to embeddings for code-switched data with resource-scarce languages in a South African multilingual environment and study the capabilities of these approaches when performing selected NLP tasks.

1.3 Project scope

Given the problem statement above, we restrict the initial scope of the research to data obtained in a typical commercial environment in South Africa. Thus, the majority of both data and resources will be in (South African) English, with code switching to the other national languages occurring with various frequencies. We will investigate typical NLP tasks in this environment, such as text classification.

1.4 Research questions

With the aim to investigate the use of embeddings in a multilingual environment, the following research questions will be pursued:

- Can embeddings be used in a resource-constrained multilingual environment where code-switching occurs?

- Is the quality of an embedding dependent on having large amounts of data available in the target language or can embeddings perform on a competitive level when trained on a small dataset?
- How can embeddings be used to train specific NLP tasks such as text classification?
- Do embeddings improve system performance over more conventional approaches such as N-grams and bag-of-words (BoW) in this environment?

1.5 Objectives of the study

The objectives of the study include the following, considering the research questions:

- Determine suitable methods to train embeddings in a multilingual environment.
- Investigate the effect of the dataset size on embeddings.
- Find appropriate classifiers to evaluate the performance when applying these embeddings to NLP tasks such as those mentioned in Section 1.3.

1.6 Research methodology

This study consists of the following steps:

- **Literature review:**
  Gain a proper understanding of embeddings. Investigate approaches to multilingual embeddings and recent research focusing on text classification using embeddings.

- **Data and pre-processing:**
  1. Gather data and speech-recognition outputs produced by a commercial speech-recognition system on South African data (for example, recognition of radio and television broadcasts by the Saigen[1] speech recogniser.
  2. Gather and customise relevant off-the-shelf text embedding systems.

- **Experimental development:**
  1. Develop customised text embedding systems.
  2. Create classifiers that utilise embeddings as well as baseline features such as BoW to accomplish selected NLP tasks.

- **Assess and discuss the experimental findings:**
  Develop and apply metrics for the comparison of different combinations of features and classifiers in order to understand how well they perform the selected tasks.

1.7 Dissertation overview

This dissertation aims to investigate the use of word embeddings in a multilingual environment. This study consist of an empirical investigation followed by a practical application of embeddings to a real-world NLP task. The dissertation is structured as follows:

- In Chapter 2 we look at the necessary background information and review existing literature on word embeddings and their application in a multilingual environment.

- In Chapter 3 we consider the data used in this study and discuss the pre-processing steps.

[1] `https://www.saigen.co.za/`

- In Chapter 4 we describe the experimental setup that is used to train and evaluate several word embedding architectures.
- In Chapter 5 we examine the effect that different hyper-parameters have on the embeddings and describe the trained systems.
- In Chapter 6 we evaluate the models' performance on an extrinsic task.
- In Chapter 7 we review the objectives met during the study. We summarise the key findings and their implications. We look at future work.

1.8 Publications

The paper 'Word embeddings for recognised multilingual speech', accepted for presentation at the *2020 Southern African Conference for Artificial Inteligence Research*, summarises the research described in this dissertation.

Chapter 2

Background

In this chapter we give background information from the relevant literature and introduce several concepts that are essential in this study.

2.1 Introduction

In this chapter we give a brief overview of the history of embeddings and how they work. We assess the leading architectures in the field, consider approaches in related studies that use embeddings with different data-related constraints, and consider different ways to evaluate embeddings.

Word embeddings are typically applied to written text; in the current contribution, we study their application to speech as recognised by an automatic speech recognition system. We further restrict our attention to this system as deployed in South Africa.

2.2 Related work

2.2.1 Brief overview of word embeddings

Several different terms are used to refer to word embeddings. The field of computational linguistics often refers to the term distributional semantic model. Distributed word representations or semantic vector space or word space are other terms used to refer to word embeddings.

Embeddings are continuous vector representations of words, typically trained on a very large unlabelled dataset. Modern embeddings have their roots in the Continuous BoW (CBoW) and Skip-gram vector learning models, known as Word2Vec, as proposed by Mikolov et al. [4]. Since then, word embeddings have been widely used to address NLP tasks (e.g., language identification, text classification, sentiment analysis etc.) because they encode syntactic as well as semantic relationships between words. Embeddings are based on the distributional hypothesis introduced by Harris in 1956 [5]. The hypothesis states that words with similar meanings will occur in similar contexts. An embedding is a learned representation of text where words with similar meanings will have similar vector representations, thus capturing the meaning of a word.

One-hot encodings were a conceptual predecessor of present-day embeddings. In this method, the data is cleaned and each word is assigned a one-hot encoding that is mapped to word vectors. Large training datasets are necessary and the approach is computationally expensive. Sparse word representations such as one-hot encoding use thousands or millions of dimensions, whereas embeddings usually consist of hundreds of dimensions.

For embeddings, words are represented in a vector space where words are mapped to individual vectors. The way in which the vectors are learned is similar to how neural networks learn. Popular embedding models such as Skipgram and CBoW use shallow neural networks. Models using explicit factorisation, such as GloVe have also become popular. These methods are summarised below.

2.2.2 Standard word-embedding architectures

Word2Vec

Word2Vec was developed by Mikolov et al. [4] in 2013. An important advantage of this approach is that high-quality embeddings can be learned efficiently, making it possible to add more dimensions from large datasets. Two learning models were introduced as part of the Word2Vec architecture: the CBoW and Skip-gram models. The CBoW model learns the embedding by predicting a word based on its context. The objective of the Skip-gram model is to identify word vectors that are useful for predicting the context words. Mikolov et al. [4] formally state the objectives as follows: given a sequence of training words $w_1, ..., w_T$, the Skip-gram model will try to maximise the log-likelihood of

$$\sum_{t=1}^{T} \sum_{c \in C_t} \log p(w_c | w_t) \tag{2.1}$$

where the context C_t is the set of words surrounding word w_t. The vector parameters are the probability of a context word w_c given the target word w_t. A scoring function s maps word and context pairs to scores in $\mathbb{R}$. Softmax is used to predict the context word:

$$p(w_c | w_t) = \frac{e^{s(w_t, w_c)}}{\sum_{j=1}^{W} e^{s(w_t,)}}. \tag{2.2}$$

With this model only one context word can be predicted, given a target word. The problem can be rephrased as a set of independent binary classification tasks so that the context words can be predicted independently. All context words can be treated as positive examples and negative examples can randomly be sampled from the dictionary. Using the binary logistic loss function, the negative log-likelihood for a context position c can be obtained:

$$\log(1 + e^{-s(w_t, w_c)}) + \sum_{n \in N_{t,c}} log(1 + e^{s(w_t, n)}) \tag{2.3}$$

where $N_{t,c}$ is a set of negative examples sampled from the vocabulary. Using the logistic loss function $l : x \mapsto log(1 + e^{-x})$, the objective can be revised as:

$$\sum_{t=1}^{T}\left[\sum_{c\in C_t} l(s(w_t,w_c)) + \sum_{n\in N_{t,c}} l(-s(w_t,n))\right]. \tag{2.4}$$

If for each word w in the vocabulary an input vector u_w and an output vector v_w in $\mathbb{R}^d$ are defined, the score s can be computed as the scalar product between the target word w_t and context vectors w_c as $s(w_t, w_c) = u_{w_t}^T v_{w_c}$.

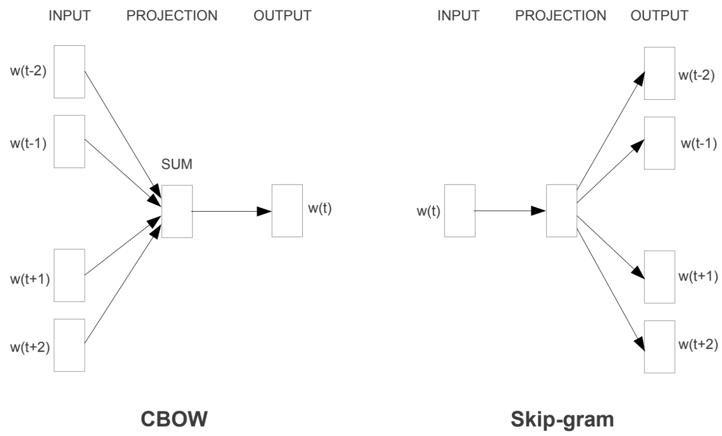


Figure 2.1: Word2Vec training models, according to Mikolov et al. [1]

GloVe

GloVe is an extension of the Word2Vec model developed by Pennington et al. [6]. GloVe uses factorisation techniques such as latent semantic analysis based on global text statistics and combines them with the context-based learning of Word2Vec. GloVe creates a word co-occurrence matrix using statistical methods across the whole text corpus.

FastText

The FastText model is an extension of the Skip-gram model introduced by Mikolov et al. [4] that takes into account sub-word information. Bojanowski et al. [7] argue that embeddings for morphologically rich languages can benefit from using character-level information because the formation of words is constrained by rules. Bojanowski et al. [7] train embeddings for character n-grams. Words are represented as the sum of the relevant n-gram vectors. To indicate prefixes and suffixes, the start and end of a word are identified by the boundary symbols <and >. The word is then divided into n-grams. For example, the word "applied" will be written as "<applied>" and if n=3, the word would be represented by its character tri-grams:

<ap, app, ppl, pli, lie, ied, ed>

The entire word w, as well as the n-grams, is included in the dictionary. Given a dictionary of size G and a word w, the set of n-grams appearing in w is represented by $G_w \subset \{1, ..., G\}$. A vector representation z_g is associated with each n-gram g. The vector representations of the n-grams in a word are summed together as

$$s(w, c) = \sum_{g \in G_w} z_g^T v_c. \tag{2.5}$$

FastText is especially good at creating embeddings for rare words and smaller datasets by using the character-level information and sharing information across words.

ELmo

Embeddings from Language Models (ELMo) is a deep contextualised embedding developed by Peters et al. [8] that models the syntax and semantics of a text. It can efficiently distinguish the meaning of a word based on the context in which it occurs. It is considered a deep embedding because it combines all the layers of a pre-trained neural network. ELMo embeddings are character-based. Similar to FastText, character-based embeddings allow the model to create embeddings for out-of-vocabulary words.

BERT

Bidirectional Encoder Representations from Transformers (BERT), developed by Devlin et al. [9], is designed to apply bidirectional training to text by looking at the entire sequence of words at once. Previous architectures consider a sequence of text during training from left to right or they train from both left to right and right to left. Devlin et al. [9] show that a bidirectional model has better understanding of the context than a single-direction model. Additional output layers can be added to the pre-trained BERT model to train different NLP tasks. The size of the dataset is very important when working with a BERT model. Larger datasets will perform significantly better. This method is also computationally more expensive than other embedding architectures.

2.2.3 Code switching

Code switching often occurs among people who have similar fluency in multiple languages, as they tend to borrow words from familiar languages at regular intervals during informal conversations. Sitaram et al. [10] stated that commercial enterprises can improve their advertisement strategies by understanding user sentiment about products expressed through code-switched language. According to Sitaram et al. [10], code switching typically occurs in informal situations, and some tasks, e.g. sentiment analysis, are more likely to benefit from code-switching models. Myers-Scotton [11] introduced a matrix language (ML) frame model that states that two languages spoken by a bilingual speaker are not equal in status. There is always a ML and an embedded language. The ML will determine the morphological and syntactic frame into which elements of the embedded language are inserted. It is possible for the ML to change during an utterance. Predictions can be made as to which elements of an utterance will come from which language by distinguishing between content and system morphemes.

Rijhwani et al. [12] used word-level language identification to discover code-switched sentences on Twitter. They used the number of switch points in a tweet to indicate the degree of code switching. For example, English-German tweets generally have only one switch point, which implies that the tweet is usually only a translation of the same content and not an example of true code switching.

The dataset used to train and test embeddings has a substantial effect on the result. Comparisons between embeddings should take into account the dataset used during training and testing. Gambäck and Das [13] argued that as the level of code switching increases in a dataset, the performance of the model is expected to decrease. There are several statistical measures to compare the level of code switching in datasets. Most measures presume that the datasets are monolingual.

1. M-index: Introduced by Barnett et al. [14], the multilingual index (M-index) quantifies the ratio of languages in the corpus based on the Gini-coefficient. It measures the degree to which a language is distributed in the dataset. However, the M-index does not indicate if the languages are integrated with each other and therefore it is unable to indicate whether code switching occurred or not. The M-index is calculated by the equation

$$M - Index = \frac{1 - \sum_j p_j^2}{(k-1)\sum_j p_j^2} \tag{2.6}$$

   where $k > 1$ is the number of languages, p_j is the number of words in a language over the total number of words in the dataset. j ranges over all the languages present in the dataset. The M-index is bounded between 0 (a monolingual dataset) and 1 (where each language in the dataset is represented equally).

2. CMI: The code-mixing index (CMI) measures the amount of code switching by looking at the frequencies of words. CMI was proposed by Gambäck and Das [15]. CMI can be used on an utterance and corpus level. To calculate the CMI at the utterance level, the most frequently used language is first identified and then the frequency of all the words in the utterance that belong to other languages is determined.

$$CMI = \frac{\sum_{i=1}^{N}(w_i) - max\{w_i\}}{n-u} \tag{2.7}$$

where $\sum_{i=1}^{N}$ is summed over all N languages. w_i are all the tagged words for a language and $max\{w_i\}$ is the highest number of words in a single language present in the utterance. n is the total number of words, and u is the number of unique language tags of n. If there is only one unique language tag present in the utterance, then $n = u$ and the index will be zero. The original CMI proposed by Gambäck and Das [15] did not consider the integration of code switching either and Gambäck and Das [13] expanded the index to account for the integration between languages:

$$C_c = \frac{\sum_{x=1}^{U} C_u(x) + w_p \delta(x)}{U} + w_s \cdot \frac{S}{U} \cdot 100 \tag{2.8}$$

$$\delta(x) = \begin{cases} 0 & x = 1 \vee L_{x-1} = L_x \\ 1 & x \neq 1 \vee L_{x-1} \neq L_x \end{cases} \tag{2.9}$$

where U is the number of utterances in the dataset. S is the number of utterances that contain code switching. P is the number of code-switched points in an utterance, and δ shows the frequency of inter-utterance switching.

3. I-index: Introduced by Guzman et al. [16], the integration index (I-index) complements the M-index by summing up the probability that there has been a switch. It serves as a simplified version of the revised CMI index of Gambäck and Das [13]. A valuable benefit of the I-index is that it does not require the dataset to be divided into utterances and does not contain computing weights. Given a dataset where each token has a language tag $\{l_i\}$ where i ranges from 1 to n, the size of the dataset:

$$I - index = \frac{1}{n-1} \sum_{1 \leq i < i+1 \leq n} S(l_i, l_{i+1}) \tag{2.10}$$

where $S(l_i, l_{i+1}) = 1$ if $l_i \neq l_{i+1}$ and 0 otherwise. The index is bounded between 0 (where no switching occurs) and 1 (if code switching occurs between each word pair).

2.2.4 Multilingual approaches

Bilingual embeddings use cross-lingual supervision such as parallel corpora [17] or a bilingual lexicon [18] to train embeddings and map them to a shared space [19]. Methods that use parallel corpora or dictionaries require language identification to determine the right mapping to apply. The monolingual embedding spaces of two languages can also be unevenly aligned owing to lexical borrowing, code switching or the distribution of named entities. Multilingual embeddings build on the same idea as bilingual word embeddings to represent words from multiple languages in a single distributional vector space. According to Chen and Cardie [19], the main problem with multilingual embeddings are, that they require monolingual embeddings trained on independent datasets. The embeddings for all languages are then mapped to the embedding space of a target language (usually English).

This approach is problematic for low-resource languages in several ways. Firstly, the inter-dependencies between languages are ignored. Identifying these inter-dependencies are important in the context of code switching, as different languages often share words through named entities or lexical borrowing. According to Chen and Cardie [19], the quality of the shared embedding space will degrade when similar languages (such as isiZulu and isiXhosa) are individually mapped to the embedding space of the target language (English) that does not share many similarities with the other languages. A second shortcoming of this method is that resource-scarce languages do not have the necessary data to train bilingual word embeddings for these language pairs. Wick et al. [20] propose using a large corpus of multilingual text to produce a single word embedding that generalises across many languages. The basis for this method is the hypothesis that words across languages will appear in the same context, thus words in one language can borrow distributional statistics from context words in another language. Chen and Cardie [19] propose a novel unsupervised method to train multilingual word embeddings using monolingual corpora, which exploits the inter-dependencies between languages and maps the monolingual embeddings to a shared multilingual embedding space. Their model achieves higher performance than state-of-the-art supervised methods when tested on multilingual word translations and cross-lingual word similarities.

2.2.5 Resources

Various studies have been conducted in the area of training NLP systems with code-switched data. However, the limited availability of code-switched data remains a problem. Various studies have proposed creating code-switched datasets synthetically by combining monolingual resources from different languages. According to Pratapa et al. [21], code-switched embeddings should ideally be trained using a code-switched dataset if it can be procured. Other techniques train a system on monolingual data and tune the model with a small code-switched dataset.

Sitaram et al. [10] have stated that although informal conversations available on Twitter, Facebook, internet forums etc. exist, the pattern that these text data follow is different from code-switched speech. The type of data needed thus depends on the environment and task for which the model will be used.

Methods for training embeddings for resource-scarce languages include transfer learning, where a model is trained on a language with a large amount of available data, and that model is then applied to a language with limited data availability [22]. Some approaches to multilingual word embeddings combine multiple languages into a single vector space. Low-resource languages can then benefit from the large number of data available in a high-resource language [20].

2.2.6 Evaluating embeddings

Various evaluation methods for embeddings have been proposed; however there is still no standard way of evaluating word embeddings. According to Wang et al. [23] there are a few properties that good word embedding should have:

1. Non-conflation: The model should be able to identify differences in different contexts and allow the local context around a word to influence the embedding.

2. Robustness against lexical ambiguity: Models should be able to get the sense of a word from its context and link it with the appropriate embedding.

3. Demonstration of multifacetedness: Syntactic, phonetic and morphological properties of a word should influence the embedding. For example, the embedding of a word should change in accordance with the prefix, suffix or tense of a word.

4. Reliability: Different representations of models should score consistently.

5. Good geometry: A small set of more frequently used, unrelated words should be evenly distributed, while a larger set of more infrequent words should cluster together around a frequent word.

There are two types of methods to evaluate word embeddings, namely intrinsic and extrinsic methods.

1. Intrinsic methods test the quality of an embedding independently of how well they perform on some NLP task.The syntactic and semantic relationship between words is measured directly, according to Wang et al. [23]. Intrinsic methods include cosine word similarities, word analogies, concept categorisation, outlier detection and QVEC proposed by Tsvetkov et al. [24]. Wang et al. [23] found that cosine word similarities, word analogies and concept categorisation perform more consistently over different models and thus serve as better intrinsic evaluation methods. Each different intrinsic method measures a different property of word-embedding models. It is generally best to use all three of the better performing intrinsic evaluation methods to get a broader understanding of the embedding quality.

2. Extrinsic methods compare how well an NLP task performs if an embedding is used as an input feature. Some NLP tasks show better correlation to the intrinsic evaluation methods, e.g. the performance of a model on sentiment analysis

and word analogies shows correlation, as both focus on the combination of word meaning. Neural machine translation methods show a stronger correlation with word cosine similarities, as the mapping between word pairs is important in both cases. Extrinsic methods can be preferred over intrinsic methods, as none of the intrinsic methods provide high correlation to NLP tasks that depend on sequential information, e.g. part-of-speech tagging (POS-tagging), chunking and NER.

2.2.7 Applying embeddings to NLP tasks

As stated in the previous section, there is no gold standard to measure word embeddings. Different embedding models will yield better performance on different tasks. That is why we argue that it is best to evaluate embeddings on a specific NLP task. POS-tagging, chunking and NER offer interesting problems in a multilingual environment because the code switching that occurs would most likely present difficulties. Given the dataset that we used, text classification would best suit our model. In Chapter 6 we will discuss our decision in more detail.

2.3 Conclusion

In this chapter, we gave a short overview of what embeddings are and why they are useful. We reviewed the leading embedding architectures and related work. We introduced key concepts and constraints relating to our study and discussed evaluation approaches.

Chapter 3

Data and pre-processing

In this chapter we discuss the data that will be used in the study. We will discuss the pre-processing methods that were used.

3.1 Introduction

South African NLP tools should be derivable from the smaller datasets that are typical of low-resource languages such as the indigenous South African languages. In addition, South Africa is a multilingual society where people often switch between languages in speech. Resource constraints are exacerbated by the fact that code-switched text corpora are hard to find, as written materials tend to avoid code switching. While word embeddings are usually trained on large language-specific datasets, we propose to train word embeddings on a South African code-switching dataset, consisting of speech-to-text transcriptions of radio and television broadcast recordings.

Most research done on bilingual word embeddings train word embeddings using monolingual datasets. Pratapa et al. [25] argue that using monolingual datasets is not sufficient when dealing with a multilingual environment because monolingual datasets do not represent the syntactic structures and cross-lingual semantic associations present in a code-switched dataset. Pratapa et al. [25] used a synthetic code-switched dataset generated using linguistic models to train their word embeddings. We propose to go one step further by using recognised speech data that contain natural code switching as our dataset.

The semantic information in a document can be used to differentiate between texts using the same vocabulary to express different ideas on the same subject. This is especially useful when classifying texts according to sentiment. A few techniques are used to enhance the input in order to include information about the semantic relationships between words or phrases. Probabilistic latent semantic analysis and latent Dirichlet allocation are methods that create a low-dimensional space where concepts are represented by multiple expressions. NLP methods, such as NER, POS-tagging and semantic role labelling, as well as sources such as WordNet [26] and Wikipedia, can also be used to enrich the dataset. These features are usually still sparse and redundant information exists among them [27]. Bojanowski et al. [28] proposed a simple method to update the word vectors to fit the distribution of a new dataset in order to extend the lexicon by combining low-frequency words from different datasets. As stated by the authors, this method is not a definitive solution and rather serves as a proof of concept.

3.2 Analysing the data

3.2.1 Dataset

For our South African code-switching dataset, we used speech-recognition outputs produced by a commercial speech-recognition system developed by a South African speech-analytics company, Saigen[1], on South African radio and television broadcasts

[1] `https://www.saigen.co.za/`

obtained from the media-monitoring group Novus[2]. The dataset includes recordings of 103 different South African radio and television stations. On these stations 12 different languages are spoken: English, Afrikaans, isiZulu, isiXhosa, Sesotho, Tshivenda, Sepedi, Siswati, Setswana, Ndebele, xiTsonga and Hindi. Figure 3.1 shows the number of radio and television stations in each language. Seventy-seven of these stations broadcast in English, eight in Afrikaans, six in Setswana, and four in isiZulu. The rest of the languages each has one dedicated station. For the purpose of this study, only English data (as defined below) were used. These data, however, still include much code switching to the other languages and are a useful representation of spoken South African English.

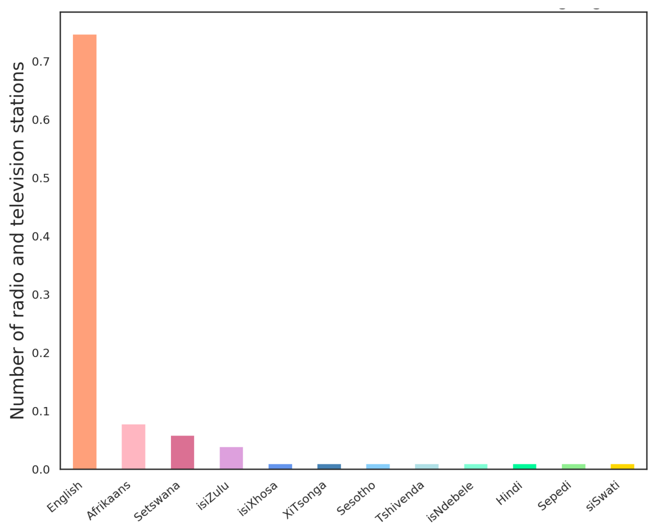


Figure 3.1: Number of radio and television stations in each language

3.2.2 Creating a code-switched dataset

Filtering English data

The dataset is tagged with the main broadcasting language of the radio and television station. However, many of the non-English radio stations still include English

[2]http://novusgroup.co.za/

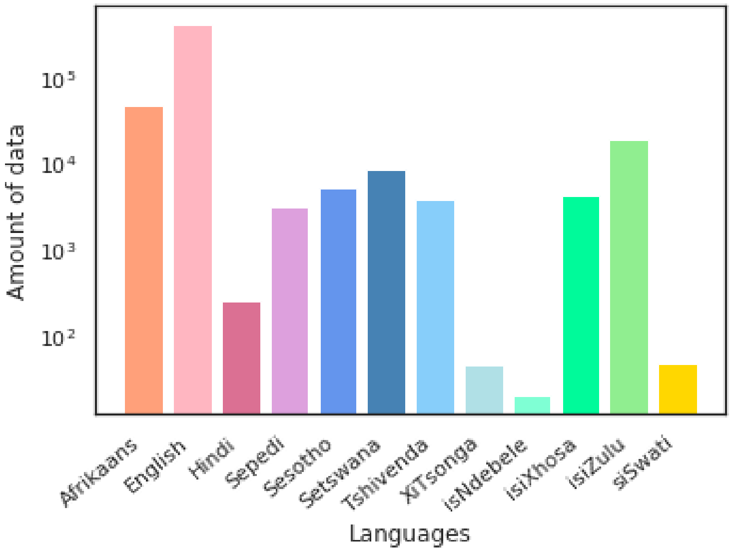


Figure 3.2: Amount of data in each language

utterances. To create a corpus with all the English sentences, a language identifier (LID) was used to identify whether a sentence was English. If the language was English, it was added to the code-switched dataset and the language of the radio station was kept as a tag. FastText has a model for language identification, which was used. An English and an Afrikaans model have already been trained. For the other South African languages, custom models were trained using a file where each sentence had a language tag. We used the NCHLT [29] Text corpora, obtained from SADiLaR [3] [30], that included monolingual text corpora for all South African languages to train the FastText LID. Each sentence was tagged with the NCHLT corpus's language. All the sentences were added to the same file and shuffled. We used 70% of the NCHLT corpus to train the LID and 30% for testing. We only included the utterances that were identified as English by the FastText LID in our code-switched dataset. These utterances were broadcast on South African radio and television stations and is therefor a realistic representation of code switching in a South African environment. The FastText LID identifies the language on a sentence level. If most words in the sentence is English, the sentence will be classified as English although some words might still be in a different language. Figure 3.3 shows the number of radio and television stations included in the code-switched dataset. Most of the data were obtained from English radio and

[3] `https://www.sadilar.org/`

television stations; however, some utterances on radio and television stations of other languages were also identified as English by the LID and were thus also included in the data. We presumed that most of the code switching would occur on these radio and television stations thus it was important to include this information in the dataset. Figure 3.4 shows the language distribution of the code-switched dataset.

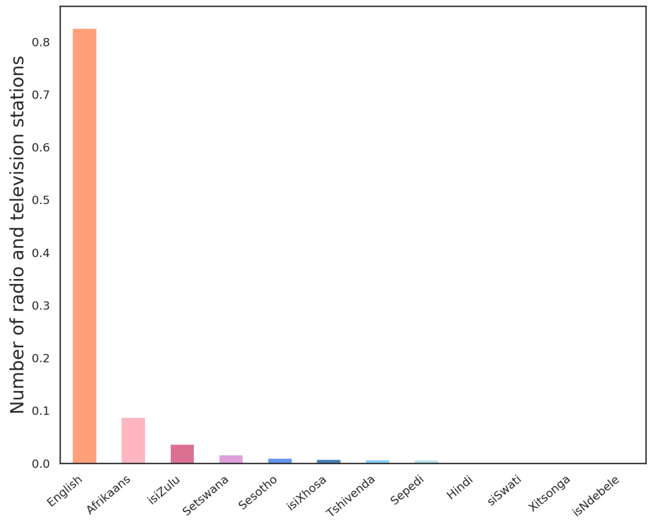


Figure 3.3: Number of radio and television stations in each language included in the code-switched dataset.

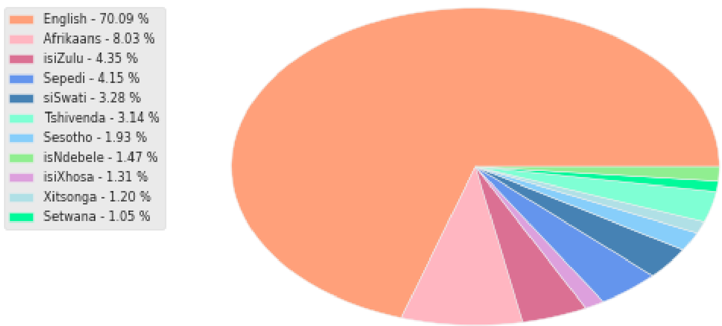


Figure 3.4: Number of words in each language included in the code-switched dataset.

Measuring the amount of code switching in the corpus

The Novus dataset consists of predominantly English radio and television stations that make up 82.3% of the dataset. The ML is English and the embedded language varies between other South African languages. To measure the M-index and the I-index, each word in the corpus has to be tagged with its language. WhatLang is an extension of FastText's LID model that can identify languages on word level. Custom models for the South African languages were trained and used to identify each word in the corpus and tag them with the LID. Sometimes code switching occurs inside a word. The LID then classifies these words as unknown. Sometimes words are not known to the model and are then also classified as unknown. We treat all the unknown words as an additional language. The number of different languages that was identified was summed. The number of times a language switch occurred in the data was calculated using the I-index discussed in Chapter 2. The I-index for the code-switched dataset is 0.299, suggesting a large amount of code switching. The distribution of the languages in the dataset was calculated using the M-index. The M-index for the code-switched dataset is 0.32, which indicates uneven representation of the different languages.

A study done by Khanuja et al. [31] compared the amount of code switching in different English-Hindi and English-Spanish datasets. In table 3.1 we compare the indexes of our South African code-switched dataset with some of their findings. Among other datasets, Khanuja et al. [31] calculated the indexes of the following datasets:

- Fire dataset [32] is an English-Hindi dataset that was created for the transliterated search sub-task.
- EMNLP 2014 [33] is an English-Spanish dataset.
- The Bangor-Miami dataset by AlGhamdi et al. [34] is a large English-Spanish dataset.
- ICON 2016 [35] is an English-Hindi dataset that consist of social media text.
- An English-Hindi Twitter dataset was created by Singh et al. [36].

Table 3.1 shows that the code-switched distribution of our South African dataset is similar to most of the code-switched datasets listed. The EN-HI Twitter dataset has higher index values that indicate a higher integration between languages. This might be due to the nature of Twitter messages, used in this dataset.

Table 3.1: Code-switched metrics for different datasets.

Dataset	I-index	M-index
South African Code-switched dataset	0.29	0.32
Fire [32]	0.33	0.39
EMNLP 2014 [33]	0.29	0.33
Bangor [34]	0.27	0.32
ICON 2016 [35]	0.34	0.4
EN-HI Twitter [36]	0.53	0.64

3.3 Pre-processing

Simple pre-processing was done using Gensim's pre-processing library [37]. The data were lower-cased, tokenised, lemmatised and stop words were removed. After pre-processing the code-switched dataset consisted of 100 K words, which were split into a training and test set of 70 K and 30 K respectively.

3.4 Testing data

The testing data were divided into categories to test the embeddings on text classification. Figure 3.5 shows the data distribution over all the classes. The classes are very imbalanced; 38% of the data are sports data, 36.5% news data, 14.7% advertisements, 7% traffic reports and 3.8% weather data. The embeddings are expected to perform better with high resource classes. Table 3.2 shows an example of an utterance for each class. News and advertisement sentences are generally longer with average sentence lengths of 41 and 48 words respectively. Sport and traffic sentences are generally between 20 to 25 words and weather sentences are short with an average sentence length of 16. Figure A.3 in Appendix A shows the performance of each of the classes with the different models. The results are discussed in detail in Chapter 6.

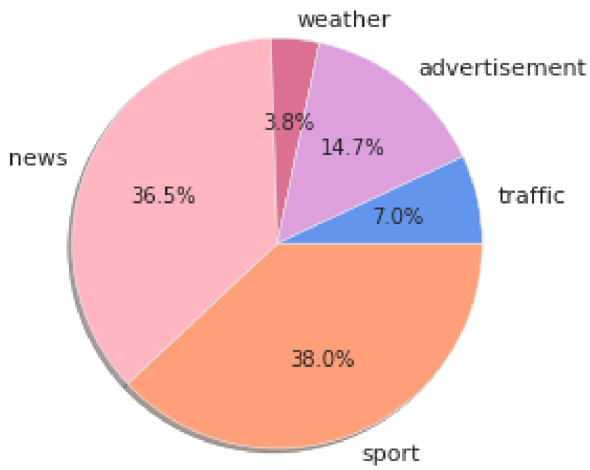


Figure 3.5: Data distribution of the classes.

3.5 Conclusion

In this chapter, we analysed the data obtained for radio and television stations in South Africa. The data included 12 different languages spoken in South Africa. We filtered these data so that the ML is English and code switching to the other languages occur. We used FastText's LID and the WhatLang extension to identify the language of an utterance. All the English utterances were included in the code-switched dataset. Some of these English utterances included words from other languages. The amount of code switching that occurred in the dataset was calculated using the I-index. The code-switch dataset had an I-index of 0.299; compared to other code-switched datasets a healthy amount of code switching occurred in our dataset. Simple pre-processing was done on the code-switched dataset. Furthermore, we explained that the classes into which the testing data were divided were very unbalanced and special measures would have to be taken during the classification phase to deal with these class imbalances.

Table 3.2: Examples of sentences in the dataset tagged with their appropriate categories.

Tag	**Sentence:**
News	in limpopo the body of the deceased was found next to the road in the bush buck ridge area last week police spokesperson leonard tladi says the mother appeared in court yesterday also on a charge of murder
Sport	the southern kings go in search of their second win of the season against conduct while the cheetahs take aim at...
Traffic	inbound slows down between sable road and the elevated freeway traffic lights are faltering retreat at prince george drive and military road
Advertisement	with our easy to use guide dstv keeps you up to date with latest episodes and action from us connect to your explorer just schedule recordings so you never miss download the app and keep the world at your fingertips
Weather	it’s around twenty-one degrees in joburg good morning

Chapter 4

Experimental setup

In this chapter we discuss the experimental setup. We analyse the hyper-parameters used to fine-tune the Word2Vec model. We consider the classification model that we will use to compare the different embedding models.

4.1 Introduction

Machine learning algorithms such as decision trees, naive Bayes classifiers and support vector machines (SVM) are commonly used in the scientific community for text classification [38]. A BoW usually represents the input where a single dimension represents each word and the dimensionality of the vector is the size of the vocabulary. Not only does a BoW model have high dimensionality and a sparse feature matrix; BoW models do not represent the relationship between words - neither syntax nor semantics. The semantic information in a document can be used to differentiate text that is using the same vocabulary to express different ideas on the same subject. This is especially useful when classifying text according to sentiment. Sinoara et al. [39] used pre-trained vectors to obtain knowledge-enhanced document embeddings. They used a combination of Word2Vec and NASARI embedded vectors that provide vector representations for the noun synsets in BabelNet. Huang et al. [27] proposed a neural network architecture to classify text documents. A low dimensional embedding similar to word embeddings represents each document. For this method, the word embeddings were learned directly in the text classification task because they argued that learning the embedding in a separate step is not optimal for text classification. Le and Mikolov [40] used paragraph vectors where embeddings can be derived from text varying in length ranging from sentences to documents.

For this study, we will be exploring the embedding architectures Word2Vec and GloVe. The Word2Vec model, still widely used today, is standardised with default values optimised by Mikolov et al. [4], where the embeddings were used to derive analogies for word pairs. These embeddings can, however, be used on a wide range of NLP tasks. Each NLP task has peculiar characteristics and challenges; hence, it is important to configure the hyper-parameter settings to fit the needs of the specific task of interest. Our main contribution is to show how the hyper-parameters of the Word2Vec model should be chosen so that the model can be applied in the context of recognised multilingual speech. Using Word2Vec, this study evaluates embeddings resulting from different hyper-parameter modifications to identify which hyper-parameters should be adjusted and in what way.

Optimising the hyper-parameters of word-embeddings is known to be important. Fanaeepour et al. [41] evaluated parameter settings for word embeddings created with the use of Swivel (a word-embedding architecture similar to Word2Vec and GloVe). They tested their embeddings on similarity and analogy tasks. Li et al. [42] compared different word-embedding architectures and found that when all hyper-parameter settings are standardised across the different architectures, they all perform very similarly. Thus, the specific embedding architecture is not of major importance, as long as the hyper-parameters are optimised for the specific tasks of interest. For our study, we will be investigating the importance of different hyper-parameters and their optimum value for Word2Vec.

In this section we will elaborate on the different models we will use during the study. We examine the hyper-parameter settings that commonly have an effect on the embedding quality. We will briefly discuss classification models used for text classification. We consider evaluation measures used for text classification tasks and choose the appropriate metric to use in this study.

4.2 Embedding architectures

4.2.1 Baseline model

For a baseline model, we implement a simple BoW model using CountVectorizer from the Scikit-learn library [43]. The vocabulary is converted to a fixed-length vector of 1000 elements and the threshold is set to 1. The shape of the resulting matrix is 25000 documents with the length of 1000 elements. A BoW creates a vocabulary list and counts how many times each word in the vocabulary appears in the dataset. A BoW model ignores the ordering of relationships in the text. The document is translated to a vector where each term is assigned weights. Each dimension corresponds to a term and each value represents the relevance. This results in a large, sparse matrix causing dimensionality problems with large datasets. Pre-processing is an important step to help reduce the vocabulary list and in turn, reduce the dimensionality of the model. We use the code-switched dataset to train the BoW baseline model. By comparing our custom Word2Vec model with the BoW baseline model, we aim to answer the research question stated in Chapter 1: do embeddings improve system performance over more conventional approaches in a code-switched environment?

4.2.2 Pre-trained embedding models

We will also be comparing our Word2Vec models with a pre-trained GloVe model, introduced by Pennington et al. [44]. The Glove model is trained on the non-zero entries of a Global word-word co-occurrence matrix that counts the number of times words co-occur in a corpus [44]. The GloVe model was pre-trained on the Wikipedia 2014 + Gigaword 5 datasets that have a vocabulary size of 400 K. We chose the pre-trained model to determine if the type of data has an effect on the performance of an embedding in the application to which it is applied, to test whether the type of data used to train the embedding is more important than the amount of data used. The model was averaged with mean square error (MSE) and term frequency - inverse document frequency (TF-IDF) respectively. The GloVe embedding was pre-trained on a much larger dataset than the code-switched dataset that has a vocabulary of 100 K, of which 70 K is used for training. By comparing our custom Word2Vec model with the GloVe model we also aim to answer the research question stated in Chapter 1 to determine if the quality of an embedding is dependent on having large amounts of data available or if embeddings can perform on a competitive level when trained on a small dataset.

4.2.3 Custom embedding models

As discussed in Chapter 2, two methods are proposed by Mikolov et al. [4] when training Word2Vec models: CBoW and Skip-gram. According to Mikolov et al. [1], the CBoW model performs better with news data and needs less data than the Skip-gram model. Kim et al. [45] also claims that when classifying news articles, the accuracy of a CBoW model is higher and more stable compared to that of a Skip-gram model. We will use the CBoW method to train the custom Word2Vec models, for which different hyper-parameter settings will be experimented with in Chapter 5. For these experiments, we will be using the code-switched dataset discussed in Chapter 3.

After we optimised the hyper-parameter settings for the custom Word2Vec model, we will train an additional Word2Vec model on a South African English dataset of similar size to the code-switched dataset. This dataset does not include code switching. With this comparison, we aim to establish if it is important to train embeddings on a code-switched dataset when used in a multilingual environment, or if an embedding trained on a monolingual dataset can obtain similar performance.

In table 4.1 we summarise the models we will be training during this study.

Table 4.1: Models trained in this study

Model	Dataset
BoW	Code-switch dataset
GloVe pre-trained	Wikipedia 2014 + Gigaword 5
Monolingual Word2Vec model	NCHLT English dataset
Code-switched Word2Vec models	Code-switched dataset

4.3 Embedding architecture configuration

When creating a word-embedding model there are a few default parameter values that should be fine-tuned to fit the evaluation task as well as the dataset. These parameters include embedding size, window size, training time, minimum count, learning rate, negative sampling and the negative sampling distribution. It is necessary to identify which of these parameters are important for us to fine-tune by looking at the characteristics of each of the parameters, the dataset and the evaluation task. We next discuss the different hyper-parameters that usually have an effect on the embedding quality.

Embedding size The embedding size is the number of dimensions in the embedding layer of the network. If the embedding dimension is higher than the vocabulary size, it will lead to over-fitting because there is no cross-word interference. The embedding would then resemble a one-hot encoding rather than a true embedding. The embedding size should thus correlate with the dataset size. A smaller dataset will perform better when using a smaller embedding size and vice versa. Caselles-Dupré et al. [41] tested the embedding size on three different datasets of varying size and observed similar results with all three dataset sizes. The performance on their tasks stopped showing improvement when the embedding size was increased to 200. These findings are in contrast to the theory that the embedding size should correlate with the dataset size, as the same embedding size was optimal for all three dataset sizes. The task the embedding is used for can however also play a role; e.g. text classification uses fewer dimensions than sentence generation.

Window size The window size is the number of context words around the target word. Research has indicated that when using a smaller window size (e.g. 2 to 15), the clusters will show interchangeable terms, so synonyms and antonyms are clustered together. Fanaeepour et al. [41] tested their embeddings on similarity and analogy tasks and found that a window size greater than four has no significant impact on the performance of their tasks. With a larger window size, the clusters are expected to show the relatedness of words, which is preferable with tasks such as text classification. Some researchers have stated that the window size does not make a significant difference when using a larger dataset; any reasonable size is acceptable.

Training time The number of times the algorithm iterates over the training data is known as the number of training epochs. The default number of epochs for word embedding models is 5. Increasing the training time can be computationally expensive, especially with larger datasets.

Minimum count The model ignores words that occur less frequently than the minimum count. This is less important when working with large datasets.

Learning rate Dillenberger [46] found that a higher learning rate over a longer time is beneficial for rare words but that a learning rate decay ultimately creates better word embeddings. Research shows that the range of the learning rate usually lies between 0.05 and 0.5.

Negative sampling To optimise the Word2Vec model, negative sampling is used. With negative sampling, the calculations of the loss function are only performed on a subset of the input, which speeds up the process [46].

Negative sampling distribution The exponent is used to shape the negative sampling distribution. A uniform distribution of α=1.0 will sample words in proportion to their frequencies. A unigram distribution of α=0 will sample all words equally, and a negative α value will sample low-frequency words rather than high-frequency words [47]. The default value of α=0.75 is set according to experiments in the original Word2Vec paper of Mikolov et al. [4] where the parameters were tested on intrinsic tasks. Caselles-Dupré et al. [47] suggest that other values may perform better for recommendation applications.

4.4 Classification

4.4.1 Classification task

There are two approaches to evaluate word embeddings: intrinsic and extrinsic tasks. Intrinsic tasks (e.g. similarity and analogy tasks) are used to test the quality of word embeddings. However, these tasks do not always correlate with real-world applications. Extrinsic tasks test how well an embedding performs on a real-world application. We propose to test the performance of the models trained with different parameter settings extrinsically by applying them to text classification. Because we are working with television and radio broadcasts, it is natural to classify the data into five categories, namely news, advertisements, sport, traffic and weather, as these are the topics that generally occur in the broadcasts. We will evaluate the other models mentioned in Table 4.1, on the same classification task. With this comparison, we aim to answer the research question stated in Chapter 1: how can embeddings be used to train specific NLP tasks such as text classification? Embeddings cannot be directly used to train a text classifier, as documents have words of various lengths. That is why a weighted average of all the words in the document has to be used. Each of these models was averaged with MSE and TF-IDF respectively.

4.4.2 Classification model

Several different classifiers are commonly used for text classification. The multinomial naive Bayes (MNB) classifier is one of the earliest methods used for text classification and still popular today because it is computationally inexpensive [38]. The MNB classifier in its base form does not perform well on unbalanced datasets [48]. One of the simplest classification algorithms is logistic regression [38] that predicts the probability of a text belonging to a class. Logistic regression separates the various classes with linear hyperplanes. Multi-class SVM is another popular text classification algorithm [38] with several variations to choose from. The main problem with SVM is that it requires a large number of support vectors that provide non-transparent results [38]. Decision tree algorithms and K-nearest neighbour have also obtained success in the field of text classification. Decision trees are very fast but can easily overfit the data [38], while K-nearest neighbour is data-dependent and is limited in larger search problems [38]. Deep learning methods such as recurrent neural networks (including long short-term memory), fully connected neural networks and convolutional neural networks, are also used to classify text successfully. The focus of this study is the performance of the embedding model and we will therefore use a simple logistic regression classifier to classify the text.

4.4.3 Evaluation

The performance of a classification model depends on the quality of the training data and the quality of the representation model. The model performance will also give varied results, depending on the aim of the evaluation metric that was used. Three types of metrics are used for classification: threshold, probability and ranking [49]. To evaluate text classification a confusion matrix is used, from which the precision, recall, f-score and accuracy, as well as the micro-average and macro-average, can be derived. The relevance of each of these measures may depend on the application to which it is applied [38].

MSE measures the difference between the class predictions and the real class labels [50]. The smaller the MSE value is, the better the model performs. The MSE is defined as

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(P_j - A_j)^2 \tag{4.1}$$

where P is the predicted value and A is the real value. The MSE is subject to weight discrimination in case of imbalanced classes [50].

Matthews correlation coefficient is commonly used to measure the quality of a binary classification problem and works well with imbalanced datasets.

F_b **Score** is the harmonic mean of precision and recall scores [51].

$$F_B = (1 + B^2) * \frac{precision * recall}{(B^2 * precision) + recall} = \frac{(1 + B^2)tp}{(1 + B^2)tp + B^2 fn + fp} \tag{4.2}$$

B indicates the significance of recall with regard to precision [51]. The standard setting is B=1, indicating that recall and precision are equally important. Common adjustments to B are 0.5 or 2, indicating that precision is twice as important as recall and vice versa.

Area under the curve (AUC) is a popular ranking type metric that ranks the overall performance of the classifier by measuring the area under the receiver operating characteristics (ROC) curve. [50]. The computational cost of the AUC curve, especially for multi-class problems, is high [50].

ROC is a popular graphical-based metric to evaluate a classifier. Graphical-based metrics provide a more in-depth analysis of the model performance [49]. However, the graphical-based output limits the discriminating properties of the metric [50]. ROC curves also tend to attribute over-performance to classifiers.

Precision-recall (PR) curves plot precision versus recall values and are another graphical-based metric used to assess performance, given unbalanced datasets [52]. Where the ROC curves focus on false classifications, the PR curves shift the focus to positive cases. Instances are ranked according to a confidence score that indicates if the instance is of a positive class. The instances are selected incrementally at each selection step where the precision and recall values are computed [53].The values are derived from the confusion matrix generated by the classifier. Precision is the fraction of the observations with a positive predicted value that are truly positive. Recall is the fraction of the examples with positive labels that are predicted to be positive [54]. If both precision and recall are 100%, the PR-curve passes through the upper right corner. Ideally a PR-curve should be as close to the upper right corner as possible [54]. In text classification tasks positive classifications are outnumbered by negative classifications, and thus of more importance [49]. The x-axis of the ROC curve represents the false positive rates. With the PR curve, the x-axis represents the true positive rate [49].

Micro- and macro-averages Micro-averaging considers each element of the class result as a binary prediction, whereas macro-averaging gives equal weight to the classification of each class [55]. The macro-average precision is the average between the precision and recall values for each instance. The micro-average is computed by summing the individual true positives, false positives and false negatives derived from the confusion matrix. Both micro- and macro-averages are important. The macro average can be used to evaluate the overall performance of a model, while micro-average are ideal when a dataset is imbalanced [55].

Because our dataset is imbalanced, we do not want to give equal weight to each class; therefore, we look at the micro-average PR curve of each model.

4.5 Conclusion

Word embeddings are used to address a wide range of NLP tasks because they encode syntactic meaning as well as semantic relationships between words. The Word2Vec architecture is a popular choice to train these models. Word2Vec is standardised with default hyper-parameter values optimised in the original research done by Mikolov et al. [4], where the embeddings were used to derive analogies for word pairs. Each NLP task has its own characteristics and challenges. In this chapter we discussed the different models we will be training to answer our research questions stated in Chapter 1. We explain the experiments we will be conducting in Chapter 5 to optimise the Word2Vec model for our classification task. We discuss the classification task that we will use to compare all the different models in this study, look at different evaluation metrics for text classification and choose the most appropriate one.

Chapter 5

Architecture configuration

In this chapter we compare the performance of the different hyper-parameter settings of the Word2Vec model using the code-switched dataset on the text classification task.

5.1 Introduction

In Chapter 4 we discuss the different hyper-parameters that usually have an effect on an embedding model. In this chapter, we will be testing the effect that some of these hyper-parameters have on an embedding model when trained on a code-switched dataset and applied to a text classification task. We identified the hyper-parameters that research has shown to have a substantial effect on the embedding performance. We start by testing the hyper-parameters in isolation to see the effect of each hyper-parameter on the model. Then we combine certain hyper-parameters to find the optimum model for our task.

5.2 Hyper-parameter selection

The hyper-parameters we will be exploring in this study are the embedding size, window size, training time, minimum count, learning rate, negative sampling and the negative sampling distribution. Table 5.1 lists the range of values for all hyper-parameters we explore in this study.

Table 5.1: Variable testing range for each hyper-parameter. Highlighted in red are the default values for each hyper-parameter.

Hyper-parameter	Testing range
Embedding size	50, 100, 150, 200, 250, 300
Window size	2, 5, 10, 15, 30, 40, 50
Training time	2, 5, 10, 20, 40, 80, 160
Minimum count	0, 2, 4, 6
Learning rate	0,0025, 0.025, 0.25
Negative sampling	0, 5, 10, 15, 20
Negative sampling distribution	-0.5, -0.25, 0, 0.25, 0.5, 0.75, 1

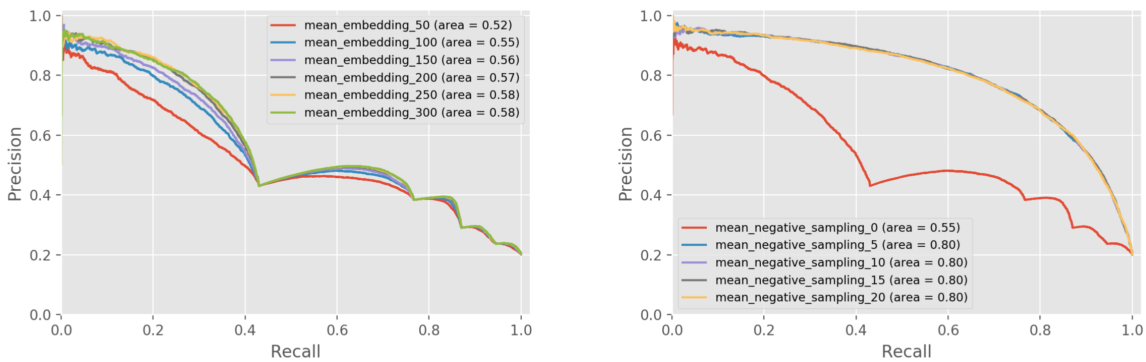
5.2.1 Comparing the feature selection models

Each Word2Vec model trained with different hyper-parameters was used to classify the testing data using a multinomial logistic classifier. The LogisticRegression model from the SciKit-learn library [43] was used. To support multi-label classification, the estimator was wrapped in a one versus rest classifier to produce binary comparisons for each class. The Saga algorithm was used to optimise the classifier and random state was used to shuffle the data. For the initial test, each hyper-parameter was changed in isolation so that the effect that it has on the default model could be observed. The biggest improvement in the model's performance was seen when negative sampling was used. A small sampling rate of 5 was enough to increase the model's performance by 25%. The PR curve is smoother and has monotonic attributes when negative sampling is used. The model showed a 6% increase in performance when the embedding size was increased to 250. After the embedding size was increased to 250, the model stopped showing further improvement, comparable to the findings of Fanaeepour et al. [41], who did not observe any improvements when the embedding size was increased beyond 200. The experiments showed that changing the window size, training time and minimum count in isolation has no effect on the model. Figure 5.1 shows the micro-averaged PR curve for the models trained with different embedding sizes and negative sampling rates. The models averaged with MSE performed slightly better than the models averaged with TF-IDF. To assist the readability of the graphs, only the models averaged with MSE are included.

5.2.2 Comparing the interactions of multiple parameters

Krebs and Paperno [56] showed the importance of the interaction of different hyper-parameters. Their research focused on three hyper-parameters: sub-sampling, shifted pointwise mutual information and context distribution smoothing. Their work showed that the same performance can be achieved when using datasets of different sizes, as long as the combination of hyper-parameters is optimal. We continued our hyper-parameter optimisation by looking at the effect that some of the hyper-parameters had on one another. The window size, training time, minimum count and learning rate had no effect on the model's performance when varied in isolation. Since there are too many combinations to test all interactions exhaustively, we discuss five combinations that were found to interact significantly. We found that changing the negative sampling in combination with other hyper-parameters can have a significant effect on the model. The only hyper-parameters that do not show an effect when changed in combination with negative sampling are window size and minimum count. Changing the window size in combination with minimum count gives a slight improvement of 1% that resulted in the best performing model for our task.

Learning rate + negative sampling Figure 5.2a shows that when the learning rate is



(a) Changes in the embedding size (b) Changes in the negative sampling rate

Figure 5.1: Micro-averaged precision-recall curves for different embedding size and negative sampling rate of feature selection models

changed in combination with negative sampling, a difference of up to 10% is reached.

Learning rate + negative sampling distribution Figure 5.2b shows that using the default smoothed unigram distribution where $\alpha = 0.75$, the model will benefit by a smaller learning rate of 0.0025, but ultimately the model will perform better when using a unigram distribution where $\alpha = 0$ in combination with a higher learning rate of 0.025.

Negative sampling + negative sampling distribution We found that when changing the negative sampling rate to 5 and using a unigram distribution where $\alpha = 0$, the performance increased by an additional 2%. This combination continued showing the best performance independently of how other hyper-parameters were changed.

Training time + negative sampling Figure 5.2c shows the performance of the model when the training time is increased. The optimum time to train the model when negative sampling is optimised is 10 epochs, after which the model's performance starts to decrease.

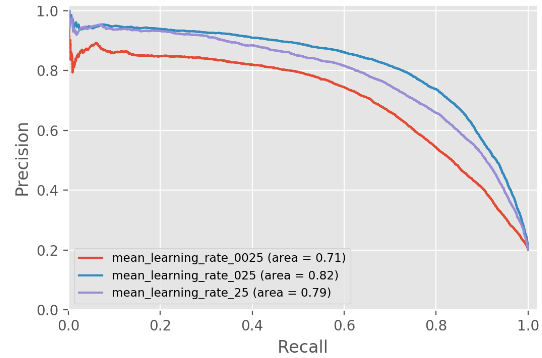
Window size + minimum count Figure 5.2d shows that when the window size is 30 or higher, all minimum count values achieve the same performance. Overall, changing the minimum count and window size does not show significant performance differences in the model.

5.2.3 Optimised model

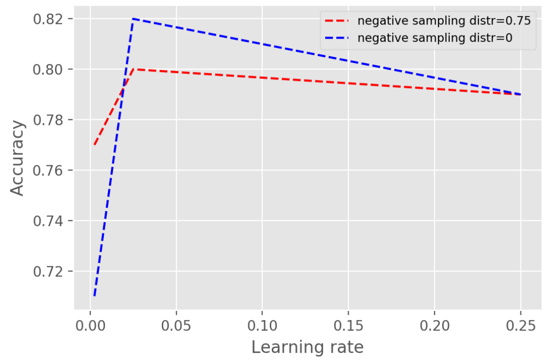
For our task we found that increasing the embedding size, window size, training time and negative sampling rate creates a better model. The default learning rate performed best of all learning rates. Changing the minimum count did not have any significant impact on the model, and the default value can be kept. Table 5.2 lists the default Word2Vec hyper-parameters along with the optimised values. Figure 5.3 compares the performance of the default hyper-parameter values with the optimised hyper-parameters of the Word2Vec model. The model showed a 31% improvement on the default values when the values are optimised. Figure A.1 in Appendix A shows a 3D visualisation of the entire optimised Word2Vec model. Figure A.2 visualises the default and optimised Word2Vec embeddings using t-SNE. The words in the optimised Word2Vec model have a clear correlation to the associated class and the clusters are much more prominent than with the default Word2Vec model.

5.3 Conclusion

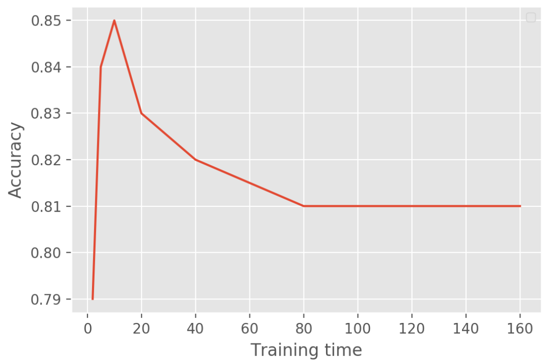
In this chapter, we evaluated the effect that fine-tuning certain hyper-parameters has on the model's performance when evaluated on an extrinsic text classification task. We tested each of the hyper-parameters mentioned in section 5.1, in isolation. We found that hyper-parameters such as the window size, training time and minimum count do not make a difference when they are changed in isolation, though the combination of these hyper-parameters is somewhat important. There are already a large number of variables that can affect the quality of the embedding. Adding the effects that the combination of different variables can have on one another to the mix becomes impracticable to measure. In this study, we tested only a few hyper-parameter combinations.



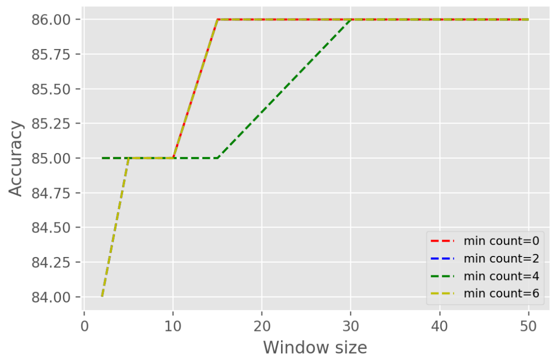
(a) Changes in the learning rate performance when negative sampling is optimised



(b) Changes in the learning rate performance when negative sampling distribution is changed



(c) Changes in the model when the training time is increased



(d) Changes in the model when the window size and minimum count are changed in combination

Figure 5.2: Changes in the model when hyper-parameters are changed in combination.

Table 5.2: Default and optimised hyper-parameter values.

Hyper-parameter	Default value	Optimised value
Embedding size	100	250
Window size	5	15
Training time	5	10
Minimum count	2	2
Learning rate	0.025	0.025
Negative sampling	0	5
Negative sampling exponent	0.75	0

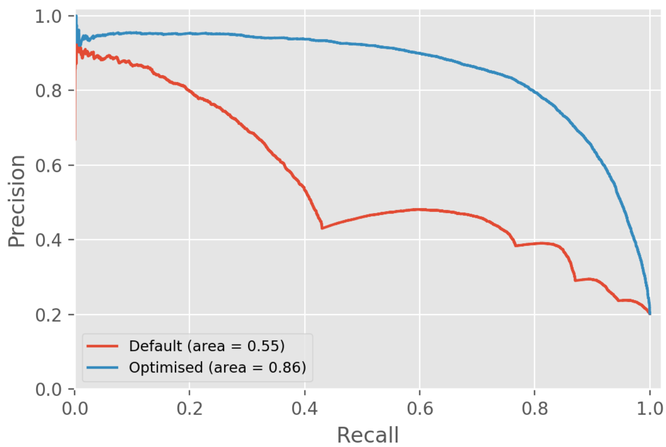


Figure 5.3: Micro-averaged precision-recall curve over all classes for the default values vs the optimised values

We found that fine-tuning the embedding size and adding negative sampling in combination with choosing the appropriate distribution for negative sampling, to the model is very important. We focused on studying the effect that negative sampling has when used in combination with other hyper-parameters. We found that optimising the parameters can produce a 31% performance improvement on a text classification task. These findings show the importance of optimising the relevant hyper-parameters to fit the task at hand.

Chapter 6

Comparing embedding architectures

In this chapter we compare the different embedding architectures and evaluate which architecture performs best when applied to text classification

6.1 Introduction

In Chapter 5 we test the effect that specific hyper-parameters have on an embedding model, when it is trained on a code-switched dataset and applied to a classification task. In this chapter, we will compare the optimised embedding model from Chapter 5 with other baseline models described in Chapter 4. We will evaluate how well each model performs with classifying text according to news categories. We will be using the code-switched test set for these experiments.

According to Faruqui et al. [57], embeddings should be tested on the specific task for which they will be used, as embeddings can capture different information depending on their parameter settings and the dataset used to train them. A number of studies by Chiu et al. [58], Schnabel et al. [59], Linzen [60] and Gladkova and Drozd [61] investigated the relationship between intrinsic and extrinsic evaluation methods. These studies all concluded that there is no direct correlation between the performance of an embedding on intrinsic tasks and its performance on a real-world problem. In this study, we will be testing our embeddings with an extrinsic evaluation method.

6.2 Comparison between baseline and feature selection models

We compare five different models with one another:

1. BoW baseline model
2. GloVe pre-trained model
3. Default Word2Vec model trained on the code-switched data
4. Optimised Word2Vec model trained on the code-switched data
5. Optimised Word2Vec model trained on the NCHLT data.

Figure 6.1 shows the precision-recall curve for the five different models. The optimised Word2Vec model shows competitive results, whereas the default model is outperformed by both the pre-trained GloVe model, as well as the BoW model. These findings show both the importance of optimising certain hyper-parameters to fit the task, and the potential of embedding representations to process recognised multilingual speech.

6.3 Comparing code-switched data with monolingual data

As discussed in Chapter 4, we compare the optimised Word2Vec model trained on the code-switched dataset with the same optimised Word2Vec model when it is trained on

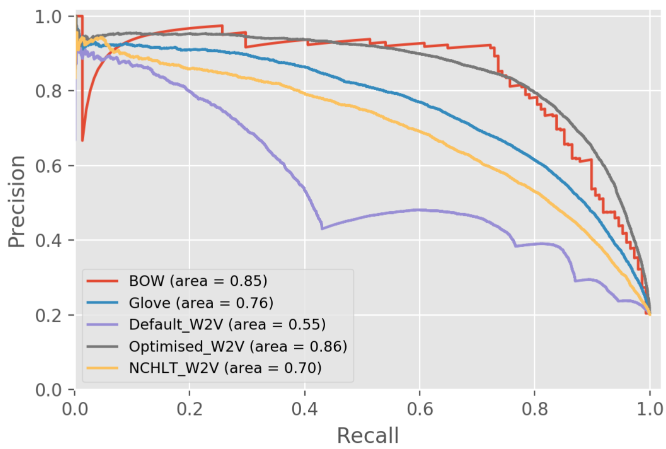


Figure 6.1: Comparison between baseline and feature selection models.

a monolingual dataset. Our aim with this experiment is to investigate the importance of a code-switched dataset in a multilingual environment. Figure 6.2 shows the performance of the two models evaluated on the code-switched text classification task. The model trained on the code-switched data shows 16% performance improvement over the same model trained on a monolingual dataset. This proves that it is important that the data fits the environment. If an embedding is to be implemented in a multilingual environment, then the dataset should contain applicable code switching. This can further be supported by the results found when comparing the code-switched optimised model with the GloVe pre-trained model. The code-switched optimised model also outperforms the GloVe pre-trained model, trained using the Google News dataset, a dataset that only includes English examples. These results suggest that the presence of code-switching examples in our relatively small training set is more important than the much larger size of the GloVe training set.

6.4 Performance regarding dataset size

These experiments show that embeddings can indeed be used in a resource-constrained multilingual environment. The quality of an embedding does not necessarily depend on having large amounts of data available. Embeddings trained on a small dataset can perform on a competitive level, as long as the hyper-parameters are optimised.

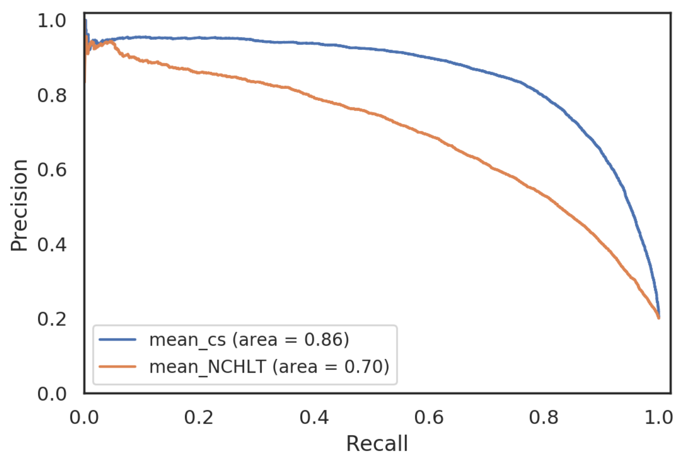


Figure 6.2: Comparison between Word2Vec model trained on the code-switched and monolingual datasets respectively.

6.5 Performance on NLP tasks

Pre-trained embeddings can be used on NLP tasks; however, our research showed that better performance on the NLP task can be achieved when the embedding is optimised for the specific task. As demonstrated by comparing the pre-trained GloVe model with the optimised Word2Vec model in figure 6.1, the optimised model has a 10% better performance accuracy than the pre-trained model. The benefit of a pre-trained embedding is that the user does not need access to a large dataset and no knowledge of training embeddings is needed.

6.6 Performance of individual classes

Figure A.3 in Appendix A, shows the ROC curves for the individual classes of the different models. For the optimised Word2Vec model the advertisement and sport classes performed the best, followed by the weather, news and traffic classes. With the default model and the Word2Vec model trained on the NCHLT dataset, we found that the advertisement class, again, performed the best followed by the news, sport, weather and lastly the traffic classes. Interestingly, the class performance does not correlate to the amount of data in each class. Instead the length and the quality of the data might rather be the main factor. Certain classes has more prominent keywords that might be more easily identified by the model.

6.7 Conclusion

When comparing the optimised model to a BoW baseline model and a pre-trained GloVe model, we find that the code-switched optimised Word2Vec model outperforms the pre-trained models and shows results that are competitive to the BoW model. The performance of the code-switched optimised Word2Vec model compared to the other embedding models, demonstrates the importance of a) optimising the model's hyper-parameters to suite the task, and b) the dataset being appropriate for the application environment.

Chapter 7

Conclusion

In this chapter we outline the key findings of this study and discuss future work.

7.1 Introduction

The aim of this dissertation is to investigate whether embeddings can be beneficial in a resource-constrained multilingual environment where code switching occurs. Embeddings are usually trained on large, language-specific datasets that do not take into account the phenomena of code switching. Research has been done on bilingual embeddings, but research on multilingual embeddings in the field has been limited. South Africa is a multilingual environment that can benefit from multilingual embeddings. Datasets that capture the multilingual properties of South African languages are scarce; therefore, there are no widely available embeddings for South African languages. In this study, we use speech recognised radio and news data that include instances of natural code switching to all South African languages. This dataset is still small compared to the English datasets on which embeddings are usually trained. In this study, we consider whether embeddings can be trained on small datasets or whether the quality of a dataset is dependent on the amount of data that is available. We test the performance of the embedding models when applied to text classification. We examine whether embeddings have a performance advantage over conventional methods such as BoW.

In this chapter, we review whether the initial objectives of the study were met. We will outline the key findings and contributions of this study and discuss future work that is necessary in this field.

7.2 Objectives of the study

The objectives stated in Chapter 1 have been met and can be summarised as:

1. We showed that embeddings can be used in a resource-constrained multilingual environment where code switching occurs.
2. We demonstrated that embeddings do not require a large dataset to perform on a competitive level and can be trained using smaller datasets, as long as the hyper-parameters are properly optimised.
3. We found that embeddings perform better when they are optimised during training to perform on a specific NLP task, rather than pre-training an embedding and using it with a variety of NLP tasks.
4. We found that embeddings perform on a competitive level with conventional approaches such as BoW. The results from embeddings can be more consistent than conventional approaches, and require less pre-processing and computational cost.

7.3 Contributions

The main contributions of this study are:

1. We demonstrate a trained embedding that can be used in the multilingual environment of South Africa where code switching occur.

2. We show how the hyper-parameters of the Word2Vec model interact with one another and demonstrate how to choose the hyper-parameters so that the model can be applied in the context of recognised multilingual speech.

3. We demonstrate that embeddings can be trained on smaller code-switched datasets and perform on a competitive level to conventional methods as well as outperforming pre-trained embeddings.

7.4 Future work

During this study, we optimised the embedding model for text classification in a multilingual environment. We believe that the field can benefit from optimisation done on document embeddings in a multilingual environment. We used the default hyper-parameter settings to train a document embedding using Doc2Vec. Figure A.4 in Appendix A shows the performance of the Doc2Vec model compared with the default and optimised Word2Vec models. The Doc2Vec model outperforms the default Word2Vec model and optimising the Doc2Vec model to suite the task, might yield results that outperforms the optimised Word2Vec model. This experiment falls outside of the scope of this study, however we would like to explore this field further.

7.5 Conclusion

Embeddings prove valuable tools to train NLP task without annotated datasets. There is limited research on embeddings in South Africa owing to constrained datasets. We hope that this study will contribute to future development of embeddings for South African languages by showing that large datasets are not necessary when training embeddings. The development of embeddings in South Africa can further aid the development of other NLP tasks.

References

[1] T. Mikolov, G. Corrado, K. Chen, and J. Dean, "Efficient estimation of word representations in vector space," vol. abs/1301.3781, Jan 2013, pp. 1–12.

[2] C. Jiang, H.-F. Yu, C.-J. Hsieh, and K.-W. Chang, "Learning word embeddings for low-Resource languages by PU learning," *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1024–1034, 2018. [Online]. Available: http://aclweb.org/anthology/N18-1093

[3] Y. Feng and X. Wan, "Learning bilingual sentiment-specific word embeddings without cross-lingual supervision," in *NAACL*, 2019, pp. 420–429. [Online]. Available: https://www.aclweb.org/anthology/N19-1040

[4] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in Neural Information Processing Systems*, vol. 26, Oct 2013.

[5] Z. S. Harris, "Distributional structure," *WORD*, vol. 10, no. 2-3, pp. 146–162, 1954. [Online]. Available: https://doi.org/10.1080/00437956.1954.11659520

[6] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[7] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with

subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, Jul 2016.

[8] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. of NAACL*, 2018.

[9] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.

[10] S. Sitaram, K. R. Chandu, S. K. Rallabandi, and A. W. Black, "A survey of code-switched speech and language processing," *CoRR*, vol. abs/1904.00784, 2019. [Online]. Available: http://arxiv.org/abs/1904.00784

[11] C. Myers-Scotton, *Contact linguistics: Bilingual encounters and grammatical outcomes*, New York : Oxford University Press, 2018.

[12] S. Rijhwani, R. Sequiera, M. Choudhury, K. Bali, and C. S. Maddila, "Estimating code-switching on Twitter with a novel generalized word-level language detection technique," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 1971–1982. [Online]. Available: https://www.aclweb.org/anthology/P17-1180

[13] B. Gambäck and A. Das, "Comparing the level of code-switching in corpora," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 1850–1855. [Online]. Available: https://www.aclweb.org/anthology/L16-1292

[14] R. Barnett, E. Codó, E. Eppler, M. Forcadell, P. Gardner-Chloros, R. van Hout, M. Moyer, M. C. Torras, M. T. Turell, M. Sebba, M. Starren, and S. Wensing, "The lides coding manual: A document for preparing and analyzing language interaction data version 1.1—july, 1999," *International Journal of Bilingualism*, vol. 4, no. 2, pp. 131–132, 2000. [Online]. Available: https://doi.org/10.1177/13670069000040020101

[15] A. Das and B. Gambäck, "Identifying languages at the word level in code-mixed Indian social media text," in *Proceedings of the 11th International Conference on Natural Language Processing*. Goa, India: NLP Association of India, Dec. 2014, pp. 378–387. [Online]. Available: https://www.aclweb.org/anthology/W14-5152

[16] G. A. Guzman, J. Serigos, B. E. Bullock, and A. J. Toribio, "Simple tools for exploring variation in code-switching for linguists," in *Proceedings of the Second Workshop on Computational Approaches to Code Switching*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 12–20. [Online]. Available: https://www.aclweb.org/anthology/W16-5802

[17] W. Y. Zou, R. Socher, D. Cer, and C. D. Manning, "Bilingual word embeddings for phrase-based machine translation," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1393–1398. [Online]. Available: https://www.aclweb.org/anthology/D13-1141

[18] T. Mikolov, Q. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," *CoRR*, vol. abs/1309.4168, Sep 2013.

[19] X. Chen and C. Cardie, "Unsupervised multilingual word embeddings," *CoRR*, vol. abs/1808.08933, pp. 261–270, 2019.

[20] M. Wick, P. Kanani, and A. Pocock, "Minimally-constrained multilingual embeddings via artificial code-switching," *Proceedings of the 30th Conference on Artificial Intelligence (AAAI 2016)*, pp. 2849–2855, 2016. [Online]. Available: https://people.cs.umass.edu/{~}mwick/MikeWeb/Publications{_}files/wick16minimally.pdf

[21] A. Pratapa, G. Bhat, M. Choudhury, S. Sitaram, S. Dandapat, and K. Bali, "Language modeling for code-mixing: The role of linguistic theory based synthetic data," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 1543–1553. [Online]. Available: https://www.aclweb.org/anthology/P18-1143

[22] L. Duong, H. Kanayama, T. Ma, S. Bird, and T. Cohn, "Multilingual training of crosslingual word embeddings," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, vol. 1. Valencia, Spain: Association for Computational Linguistics, 2017, pp. 894–904. [Online]. Available: https://www.aclweb.org/anthology/E17-1084

[23] B. Wang, A. Wang, F. Chen, Y. Wang, and C.-C. J. Kuo, "Evaluating word embedding models: methods and experimental results," *APSIPA Transactions on Signal and Information Processing*, vol. 8, p. e19, 2019.

[24] Y. Tsvetkov, M. Faruqui, W. Ling, G. Lample, and C. Dyer, "Evaluation of word vector representations by subspace alignment," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 2049–2054. [Online]. Available: https://www.aclweb.org/anthology/D15-1243

[25] A. Pratapa, M. Choudhury, and S. Sitaram, "Word embeddings for code-mixed language processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3067–3072. [Online]. Available: https://www.aclweb.org/anthology/D18-1344

[26] C. Fellbaum, Ed., *WordNet: An electronic lexical database*. Cambridge, MA: MIT Press, 1998.

[27] C. Huang, X. Qiu, and X. Huang, "Text classification with document embeddings," in *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, M. Sun, Y. Liu, and J. Zhao, Eds. Cham: Springer International Publishing, 2014, pp. 131–140.

[28] P. Bojanowski, O. Celebi, T. Mikolov, E. Grave, and A. Joulin, "Updating pre-trained word vectors and text classifiers using monolingual alignment," *CoRR*, vol. abs/1910.06241, Oct. 2019.

[29] M. Puttkammer, M. Schlemmer, W. Pienaar, and R. Bekker, "Nchlt text corpora," Sadilar, Feb. 2018. [Online]. Available: https://repo.sadilar.org/handle/20.500.12185/293?show=full

[30] North-West University; Centre for text technology (CTexT), *South African centre for digital language resources*, 2018 (accessed Oct 3, 2020). [Online]. Available: https://www.sadilar.org

[31] S. Khanuja, S. Dandapat, A. Srinivasan, S. Sitaram, and M. Choudhury, "GLUECoS: An evaluation benchmark for code-switched NLP," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 3575–3585. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.329

[32] R. S. Roy, M. Choudhury, P. Majumder, and K. Agarwal, "Overview of the fire 2013 track on transliterated search," in *Post-Proceedings of the 4th and 5th Workshops of the Forum for Information Retrieval Evaluation*, ser. FIRE '12 amp; '13. New York, NY, USA: Association for Computing Machinery, 2013. [Online]. Available: https://doi.org/10.1145/2701336.2701636

[33] T. Solorio, E. Blair, S. Maharjan, S. Bethard, M. Diab, M. Ghoneim, A. Hawwari, F. AlGhamdi, J. Hirschberg, A. Chang, and P. Fung, "Overview for the first shared task on language identification in code-switched data," in *Proceedings of the First Workshop on Computational Approaches to Code Switching*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 62–72. [Online]. Available: https://www.aclweb.org/anthology/W14-3907

[34] F. AlGhamdi, G. Molina, M. Diab, T. Solorio, A. Hawwari, V. Soto, and J. Hirschberg, "Part of speech tagging for code switched data," in *Proceedings of the second workshop on computational approaches to code switching*, Jan. 2016.

[35] A. Jamatia, B. Gambäck, and A. Das, "Collecting and annotating indian social media code-mixed corpora," A. Gelbukh, Ed.

[36] V. Singh, D. Vijay, S. S. Akhtar, and M. Shrivastava, "Named entity recognition for Hindi-English code-mixed social media text," in *Proceedings of the Seventh Named Entities Workshop*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 27–35. [Online]. Available: https://www.aclweb.org/anthology/W18-2405

[37] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proceedings of LREC 2010 workshop new challenges for NLP frameworks*, May 2010, pp. 45–50.

[38] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information (Switzerland)*, vol. 10, Apr 2019.

[39] R. Sinoara, J. Camacho-Collados, R. Rossi, R. Navigli, and S. Rezende, "Knowledge-enhanced document embeddings for text classification," *Knowledge-Based Systems*, Oct 2018.

[40] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, p. II–1188–II–1196.

[41] M. Fanaeepour, A. Makarucha, and J. Lau, "Evaluating word embedding hyper-parameters for similarity and analogy tasks," Apr 2018.

[42] P. Li, Y. Liu, M. Sun, T. Izuha, and D. Zhang, "A neural reordering model for phrase-based translation," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, Aug. 2014, pp. 1897–1907. [Online]. Available: https://www.aclweb.org/anthology/C14-1179

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[44] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://www.aclweb.org/anthology/D14-1162

[45] B. Jang, I. Kim, and J. W. Kim, "Word2vec convolutional neural networks for classification of news articles and tweets," *PLOS ONE*, vol. 14, no. 8, pp. 1–20, 2019. [Online]. Available: https://doi.org/10.1371/journal.pone.0220976

[46] J. Dillenberger, "Evaluation of model and hyperparameter choices in word2vec," Thesis, Computer Science, Institute for web science and technologies, 2019.

[47] H. Caselles-Dupré, F. Lesaint, and J. Royo-Letelier, "Word2vec applied to recommendation: Hyperparameters matter," *Proceedings of the 12th ACM conference on recommender systems*, Apr 2018.

[48] E. Frank and R. R. Bouckaert, "Naive bayes for text classification with unbalanced classes," in *Proceedings of the 10th European conference on principles and practice of knowledge discovery in databases*, ser. ECMLPKDD'06. Berlin, Heidelberg: Springer-Verlag, Sep 2006, p. 503–510.

[49] R. C. Prati, G. E. A. P. A. Batista, and M. C. Monard, "A survey on graphical methods for classification predictive performance evaluation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 11, pp. 1601–1618, 2011.

[50] M. Hossin and M. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International journal of data mining & knowledge management process (IJDKP)*, vol. 5, no. 2, pp. 1–11, Nov. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3557376

[51] Y. Sasaki, "The truth of the f-measure," *Manchester: School of Computer Science, University of Manchester*, Oct 2007.

[52] K. Boyd, K. H. Eng, and C. D. Page, "Area under the precision-recall curve: Point estimates and confidence intervals," in *Machine Learning and Knowledge Discovery*

in Databases, H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 451–466.

[53] M. Lopes and G. Bontempi, "On the null distribution of the precision and recall curve," 09 2014, pp. 322–337.

[54] J. Miao and W. Zhu, "Precision-recall curve (prc) classification trees," 2020.

[55] M. Bihis and S. Roychowdhury, "A generalized flow for multi-class and binary classification tasks: An azure ml approach," in *2015 IEEE International Conference on Big Data (Big Data)*, 2015, pp. 1728–1737.

[56] A. Krebs and D. Paperno, "When hyperparameters help: Beneficial parameter combinations in distributional semantic models," in *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 97–101. [Online]. Available: https://www.aclweb.org/anthology/S16-2011

[57] M. Faruqui, Y. Tsvetkov, P. Rastogi, and C. Dyer, "Problems with evaluation of word embeddings using word similarity tasks," in *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 30–35. [Online]. Available: https://www.aclweb.org/anthology/W16-2506

[58] B. Chiu, A. Korhonen, and S. Pyysalo, "Intrinsic evaluation of word vectors fails to predict extrinsic performance," in *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1–6. [Online]. Available: https://www.aclweb.org/anthology/W16-2501

[59] T. Schnabel, I. Labutov, D. Mimno, and T. Joachims, "Evaluation methods for unsupervised word embeddings," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 298–307. [Online]. Available: https://www.aclweb.org/anthology/D15-1036

[60] T. Linzen, "Issues in evaluating semantic spaces using word analogies," in *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 13–18. [Online]. Available: https://www.aclweb.org/anthology/W16-2503

[61] A. Gladkova and A. Drozd, "Intrinsic evaluations of word embeddings: What can we do better?" in *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 36–42. [Online]. Available: https://www.aclweb.org/anthology/W16-2507

Appendix A

Supplemental figures

This appendix contains supplemental figures, referred to in the main text.

A.1 Appendix: Chapter 5

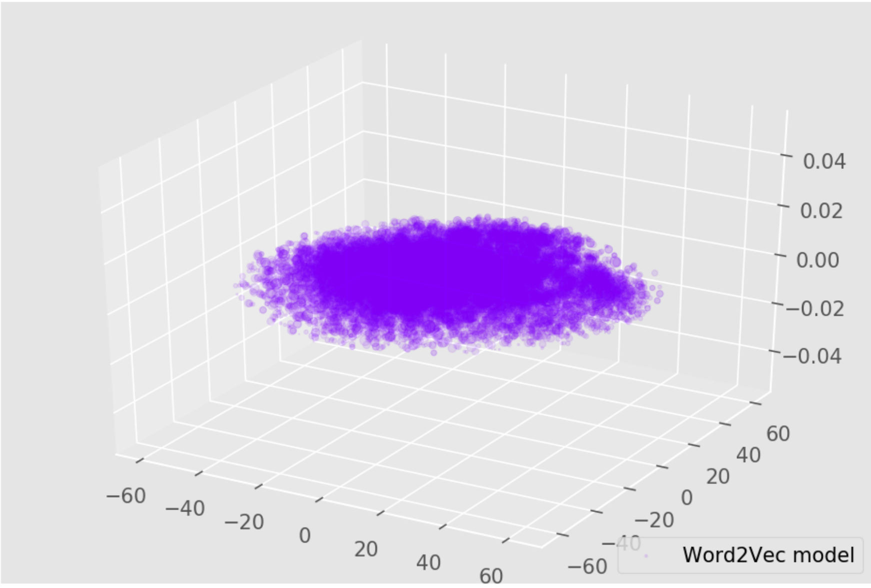
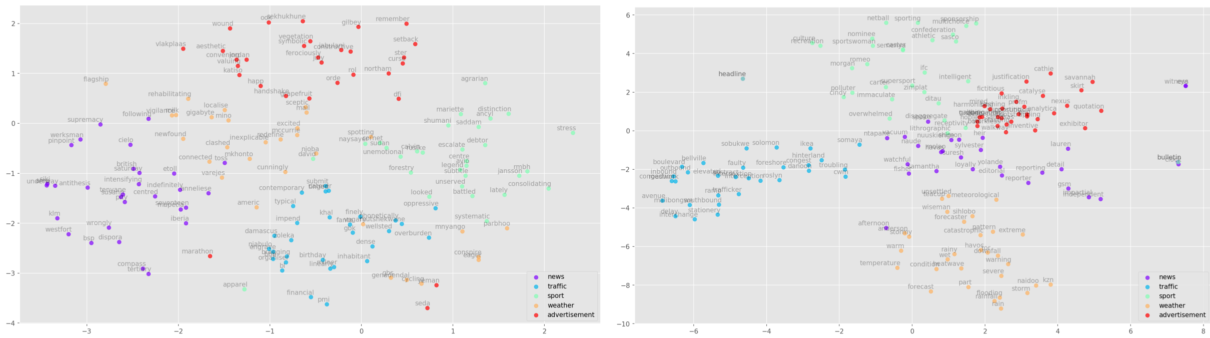


Figure A.1: Visualising the 3D optimised Word2Vec model using t-SNE.

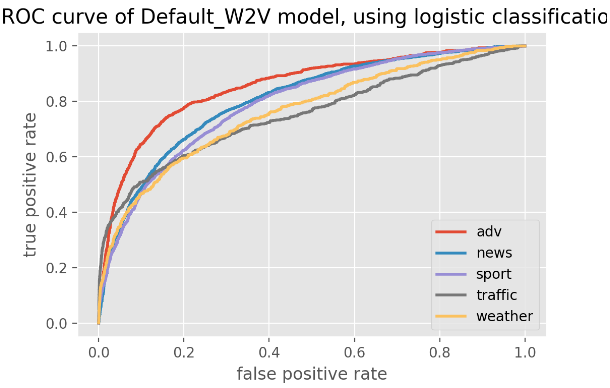


(a) Default Word2Vec model

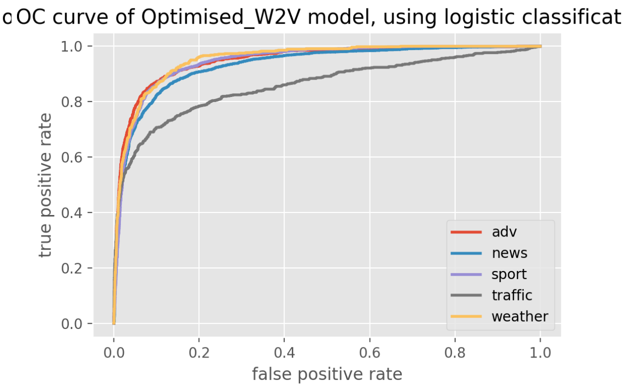
(b) Optimised Word2Vec model

Figure A.2: Visualising the Word2Vec models using t-SNE

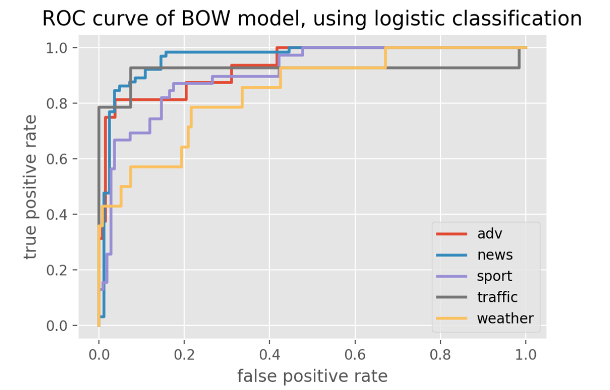
A.2 Appendix: Chapter 6



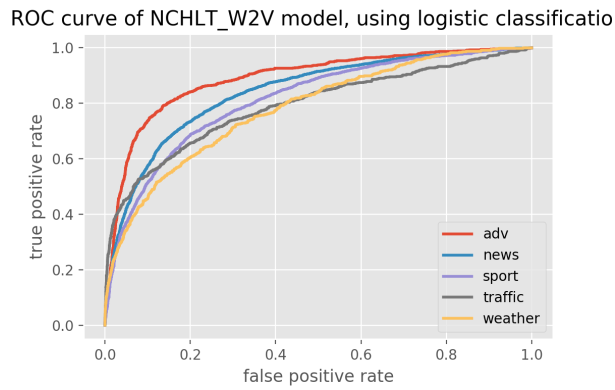
(a) Default Word2Vec model



(b) Optimised Word2Vec model



(c) BOW model



(d) Word2Vec model, trained on the NCHLT dataset

Figure A.3: ROC curves for each class of the different models

A.3 Appendix Chapter 7

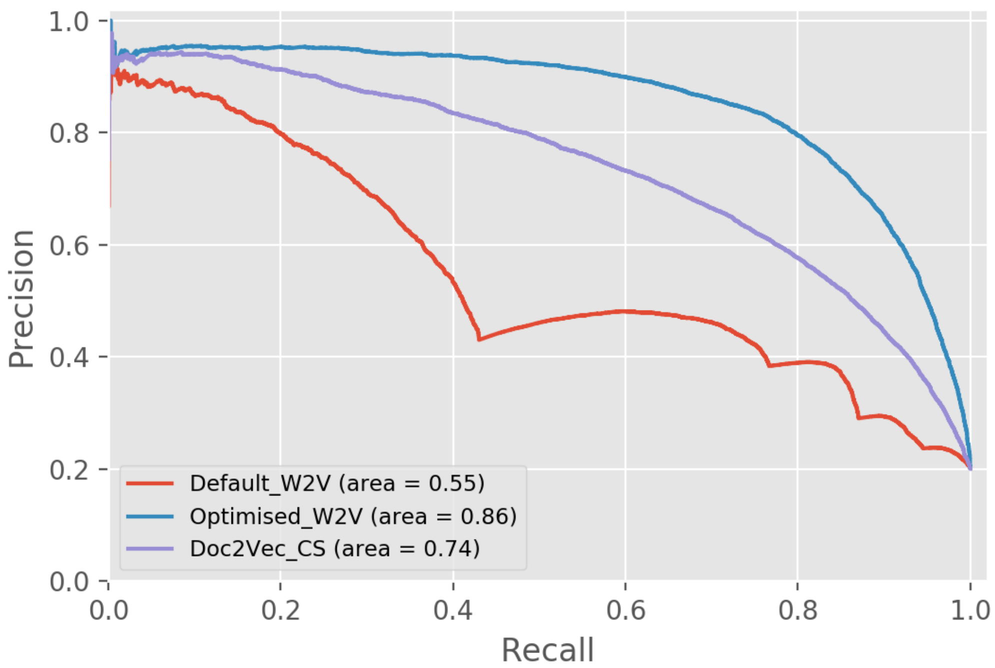


Figure A.4: Average precision-recall curve for the default and optimised Word2Vec models, compared to the Doc2Vec model.