Optimising word embeddings for recognised multilingual speech

Nuette Heyns[0000−0002−0802−5005] and Etienne Barnard[0000−0003−2202−2369]

Multilingual Speech Technologies (MuST), North-West University, South Africa; and CAIR, South Africa
{nuette.heyns, etienne.barnard}@gmail.com

Abstract. Word embeddings are widely used in natural language processing (NLP) 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 like 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. We compare these models with a bag of words baseline model as well as a pre-trained GloVe model.

Keywords: embeddings · hyper-parameter · text classification · Word2Vec.

1 Introduction

Distributed word representations or word embeddings are continuous vector representations of words, typically trained on a very large unlabelled corpus. Modern embeddings have their roots in the Continuous Bag-of-Words (CBOW) and Skip-gram vector learning models, known as Word2Vec, as proposed by Mikolov et al. [20]. Since then, word embeddings have been widely used to address natural language processing (NLP) tasks (e.g., language identification, text classification, sentiment analysis etc.) because they encode syntactic meaning as well as semantic relationships between words. Word embeddings are typically applied to written text; in the current contribution, in contrast, we study their application to speech as recognised by an Automatic Speech Recognition (ASR) system. We further restrict our attention to ASR as deployed in South Africa, for reasons that will become clear later in the paper.

South Africa is a multilingual society where people often switch between languages in speech which require custom embedding training techniques. In addition, the embeddings should be derivable from the smaller datasets that are typical of low-resource languages such as the indigenous South African languages. 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. We calculate the amount of code-switching in the dataset by measuring the I-index (Integration index) introduced by Guzman et al[12]. The code-switched dataset has an I-index of 0.299 suggesting a large amount of code switching. The distribution of the languages in the dataset is calculated using the M-index (Multilingual Index). The M-Index for the code-switched dataset is 0.32 which indicate an uneven representation of the different languages.

The Word2Vec model, still widely used today, is standardised with default values optimised in [20], 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.

There are two approaches to evaluate word embeddings; intrinsic and extrinsic tasks. Intrinsic tasks (e.g. similarity and analogy tasks) are widely used to test the quality of word embeddings, however, these tasks do not always correlate to real-world applications. Extrinsic tasks test how well an embedding performs on a real-world application. We propose to extrinsically test the performance of the models trained with different parameter settings by applying it 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. We found that optimising the hyper-parameters can produce a 31% performance increase on a text classification task. When comparing the code-switched model to a Bag of Words (BOW) baseline model and a pre-trained GloVe model, we found that the optimised Word2Vec model outperformed the pre-trained models and showed results that are competitive to the BOW model. These findings show the importance of optimising the relevant hyper-parameters to fit the task at hand.

2 Related work

Most research done on bilingual word embeddings train word embeddings using monolingual datasets. Pratapa et al. [21] 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. [21] 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 contains natural code-switching, as our dataset.

Optimising the hyper-parameters of word-embeddings is known to be important. Li et al. [17] 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.

According to Faruqui et al. [8] embeddings should be tested on the specific task it will be used for as embeddings can capture different information depending on their parameter settings and the dataset used to train them. A couple of studies Chiu et al. [5], Schnabel et al. [23], Linzen [18] and Gladkova and Drozd [11] 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 their performance on a real-world problem. In this study, we will be testing our embeddings with an extrinsic evaluation method.

Machine learning algorithms such as decision trees, Naive Bayes and Support Vector Machines (SVMs) are popular text classification methods. The input is usually represented as a BOW where each word is represented by a single dimension 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, but BOW models do not represent the relationship between words - neither syntax nor semantics. The semantic information in a document can be used to differentiate between 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. There are a few techniques used to enhance the input in order to include information about the semantic relationships between words or phrases. Probabilistic latent semantic analysis (PLSA) and Latent Dirichlet Allocation (LDA) are methods that create a low-dimensional space where concepts are represented by multiple expressions. NLP methods, such as Named Entity Recognition (NER), Part of Speech Tagging (POS-tagging) and semantic role labelling as well as sources like WordNet [9] and Wikipedia can also be used to enrich the dataset. These features are usually still sparse and redundant information exists among these features [13]. Bojanowski et al. [2] 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. Sinoara et al. [24] used pre-trained vectors to obtain knowledge enhanced document embeddings. Huang et al. [13] proposed a neural network architecture to classify text documents. Each document is represented by a low dimensional embedding similar to word embeddings. 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 [16] used paragraph Vectors where embeddings can be derived from text varying in length ranging from sentences to documents.

3 Methodology

In the following section, we will discuss the different hyper-parameters that usually have an effect on the embedding quality. When reviewing the experimental setup we will discuss the dataset, models and the hyper-parameters that we will be fine-tuning. Following the experimental setup, we will discuss the evaluation metrics and compare the different models.

3.1 Word embedding hyper-parameters

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 is important for us to fine-tune by looking at the characteristics of each of the parameters, the dataset and the evaluation task.

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 and the embedding would resemble a one-hot-encoding more than a true embedding. The embedding size should therefore correlate to the dataset size. A smaller dataset will perform better when using a smaller embedding size and vice versa. Caselles-Dupré [7] tested their embedding size on three different datasets varying in 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 to 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 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. [7] 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 like text classification. Some researchers have stated that the window size does not make a difference when using a larger dataset.

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 than the minimum count. For large datasets this is less important.

Learning rate Dillenberger [6] 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 and thus speeds up the process [6].

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 more than high-frequency words [4]. The default value of α=0.75 is set according to experiments in the original Word2Vec paper of Mikolov et al. [20] where the parameters were tested on intrinsic tasks. Caselles-Dupré et al. [4] suggest that other values may perform better for recommendation applications.

3.2 Experimental setup

Data For our South African code-switching dataset, we used speech-recognition outputs produced by a commercial speech-recognition system developed by Saigen[1], on South African radio and television broadcasts obtained from 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. 77 Of these stations are in English, 8 in Afrikaans, 6 in Setswana, 4 in isiZulu and the rest of the languages each have 1 dedicated station. For the purpose of this study only the English radio station data was used. This data however still includes much code switching to the other languages and is, therefore, a useful representation of spoken South African English. Simple pre-processing was done using Gensim's pre-processing library. The data was lowercased, tokenised, lemmatised and stop words were removed. After preprocessing the dataset consisted of 100K words, which were split into a training and test set of 70K and 30K respectively.

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

Rijhwani et al.[22] 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 that implies the tweet is usually only a translation of the same content and not an example of true code switching. Gambäck and Das [10] 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. In this study we used the M-index and the I-index.

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

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

   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 range 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. I-index: Introduced by Guzman et al.[12], the Integration 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 [10]. A valuable benefit of the I-Index is that it does not require dividing the dataset into utterances or for it to 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:

$$I - index = \frac{1}{n-1} \sum_{1 \leq i < i+1 \leq n} S(l_i, l_{i+1}) \tag{2}$$

   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.)

To measure the M-index and the I-index, each word in the corpus has to be tagged with their language. Whatlang is an extension of fastText's language identifier model that can identify languages on a 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. The number of different languages that was identified is summed. The number of times a language switch occurred in the data was calculated using the I-index. 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 indicate an uneven representation of the different languages

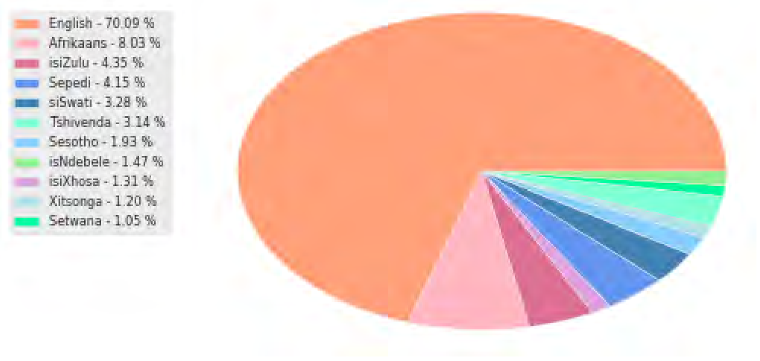


Fig. 1: The number of words in each language included in the code-switched dataset.

Models Word2Vec is based on the distributional hypothesis that words used in the same context, will have similar meanings. There are two methods proposed by Mikolov et al. [20] when training Word2Vec models; CBOW and Skip-gram, both of which consist of a single-layer architecture where the inner product of two-word vectors are calculated. A Skip-gram model aims to predict the context, given a word, while CBOW's aim is to predict a word, given the context. According to Mikolov et al. [19], the CBOW model performs better with news data and less data is needed than with skip-gram. Kim et al. [14] also claims that the accuracy of a CBOW model is higher and more stable compared to that of a Skip-gram model when the model is evaluated by classifying news articles. A CBOW method was used to train the custom Word2Vec models for which different hyper-parameter settings were experimented with. Word embeddings cannot be directly used to train a text classifier as documents have words with various lengths. That is why a weighted average of all the words in the document has to be used. Each of these models was therefore averaged with Mean Square Error (MSE) and Term Frequency–Inverse Document Frequency (TF-IDF) respectively.

For a baseline model, we implement a simple BOW model for which the parameters have been optimised. The baseline model ignores the ordering of relationships within the text. The document is translated to a vector of term weights where each dimension corresponds to a term and each value represents the relevance. We will also be comparing our Word2Vec models with a pre-trained GloVe model that is again averaged with MSE and TF-IDF respectively.

Hyper-parameter selection The hyper-parameters we explore in this study are listed in Table 1.

Table 1: The 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

4 Evaluation

An extrinsic evaluation method is used to evaluate the word embedding models. The word embedding models are used to create a text classifier that classifies the text into the categories news, sport, traffic, advertisement and weather. A logistic regression classifier is used to classify the text. The performance of a classification model depends on the quality of the training data and the quality of the representation model. The dataset was tagged with the appropriate categories. The classes are very imbalanced; 38,02% of the data is sports data, 36,49% of the data is news data, 14,74% is advertisements, 6,96% is traffic reports and 3,78% is weather data. The embeddings are expected to perform better with the high resource classes.

4.1 Comparing the feature selection models

Each of the Word2Vec models trained with different hyper-parameters was classified using a logistic classifier. Precision-recall (PR) curves are used to assess performance given the unbalanced datasets [3]. Micro-averaging considers each element of the class indicator matrix as a binary prediction, whereas macro-averaging gives equal weight to the classification of each class. because our dataset is imbalanced, we do not want to give equal weight to each class and will therefore be looking at the micro average PR curve of each model. For the initial test, each hyper-parameter was changed in isolation so that the effect that it has on the default model can be observed. The biggest improvement in the model's performance was seen when negative sampling was used. A small

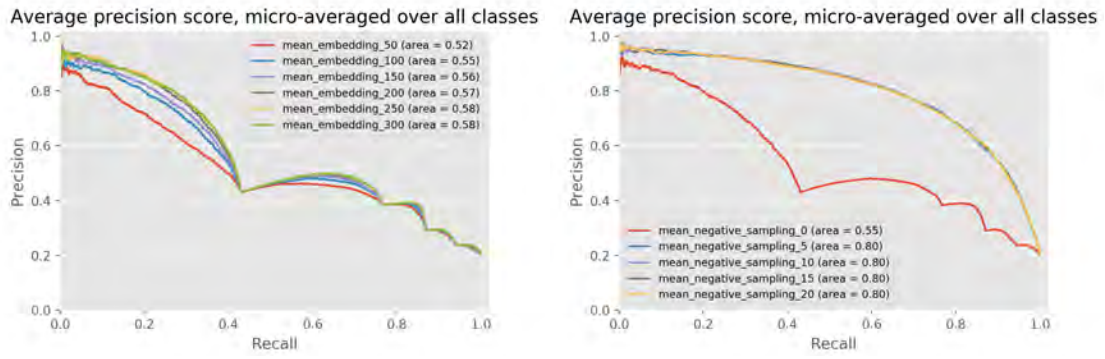
Table 2: Examples of sentences in the dataset tagged with their appropriate catagories.

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

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. [7], 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 2 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.

4.2 Comparing the interactions of multiple parameters

Krebs and Paperno [15] showed the importance of the interaction of different hyper-parameters. Their research focused on three hyper-parameters; subsampling, shifted Pointwise Mutual Information (PMI) 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 have on each other. 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 exhaustively test all interactions, we discuss five combinations that were found



(a) Changes in the embedding size

(b) Changes in negative sampling

Fig. 2: Micro-averaged precision-recall curves for different embedding size and negative sampling for feature selection models

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 when the learning rate is changed in combination with negative sampling, a difference of up-to 10% is reached.

Learning rate + negative sampling distribution 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 with an additional 2%. This combination continued showing the best performance independent of how other hyper-parameters were changed.

Training time + negative sampling 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 When the window size is 30 or higher, all minimum count values achieve the same performance. Overall, changing the minimum count and window size do not show significant performance differences in the model.

4.3 Evaluating the code-switched model model

The values that performed the best for each hyper-parameter combination were used to train the code-switched model. Table 3 compares the default Word2Vec hyper-parameters with the optimised values.

Table 3: The 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

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 the best of all learning rates and changing the minimum count does not have any significant impact on the model, and the default value can be kept. Figure 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.

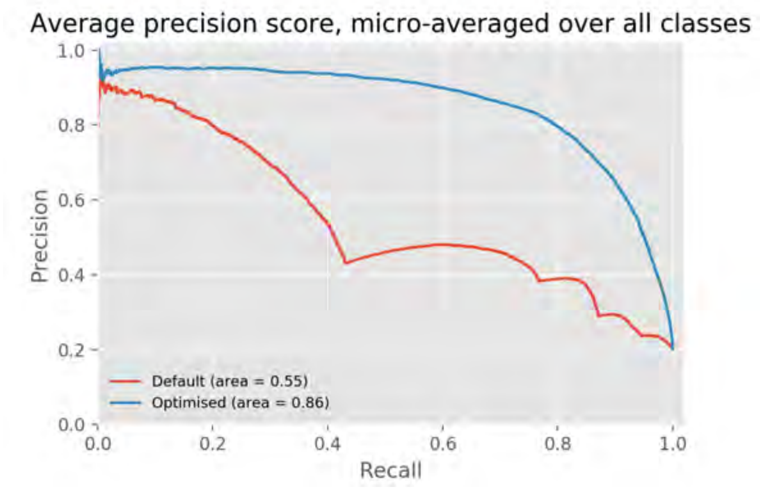


Fig. 3: Micro-averaged precision-recall curve for the default values vs the optimised values

4.4 Comparison between baseline and feature selection models

We compare the default as well as the optimised Word2Vec model (code-switch model) with a BOW baseline model as well as a pre-trained GloVe model. Figure 4 shows the precision-recall curve for the 4 different models. The code-switch model shows competitive results, whereas the default model is outperformed by both the pre-trained GloVe model, as well as the BOW model. The performance of the pre-trained GloVe model demonstrates the importance of a) optimising the model's hyper-parameters to suite the task, and b)the dataset should be appropriate for the application environment. The pre-trained GloVe model was trained using the Google News dataset, a dataset that only includes English examples. The fact that our relatively small code-switched dataset performs better than a similar embedding pre-trained on a much larger monolingual dataset suggests that training embeddings on a code-switched dataset when it will be used in a code-switched environment is more important than the size of the dataset the embedding will be trained on.

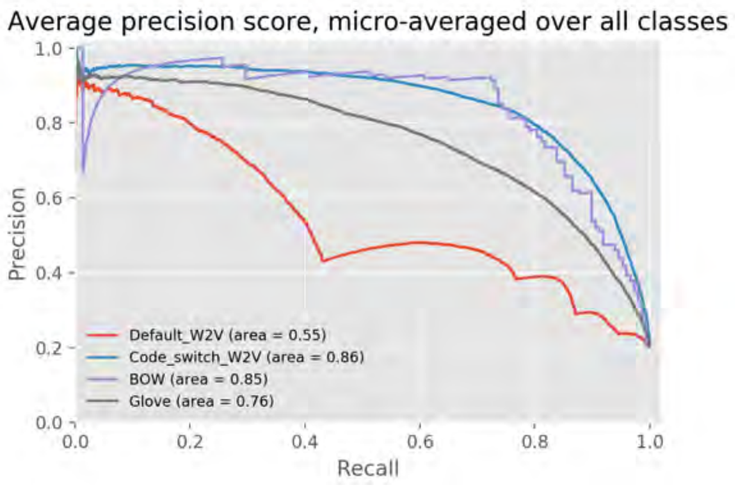


Fig. 4: Comparison between baseline and feature selection models.

5 Conclusion

Word embeddings are popularly 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 [20], where the embeddings were used to derive analogies for word pairs. Each NLP task, however, has it's own characteristics and challenges. This study evaluated the effect that fine-tuning certain hyper-parameters has on the model's performance when evaluated on an extrinsic text classification task. 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 are very important. Other 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. 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 the task. When comparing the code-switched model to a BOW baseline model and a pre-trained GloVe model, we found that the optimised Word2Vec model outperformed the pre-trained models and showed results that are competitive to 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 Acknowledgements

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

References

1. Barnett, R., Codó, E., Eppler, E., Forcadell, M., Gardner-Chloros, P., van Hout, R., Moyer, M., Torras, M.C., Turell, M.T., Sebba, M., Starren, M., Wensing, S.: The lides coding manual: A document for preparing and analyzing language interaction data version 1.1—july, 1999. International Journal of Bilingualism **4**(2), 131–132 (2000). https://doi.org/10.1177/13670069000040020101, `https://doi.org/10.1177/13670069000040020101`
2. Bojanowski, P., Celebi, O., Mikolov, T., Grave, E., Joulin, A.: Updating pre-trained word vectors and text classifiers using monolingual alignment (2019)
3. Boyd, K., Eng, K.H., Page, C.D.: Area under the precision-recall curve: Point estimates and confidence intervals. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 451–466. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
4. Caselles-Dupré, H., Lesaint, F., Royo-Letelier, J.: Word2vec applied to recommendation: Hyperparameters matter. CoRR **abs/1804.04212** (2018), `http://arxiv.org/abs/1804.04212`
5. Chiu, B., Korhonen, A., Pyysalo, S.: Intrinsic evaluation of word vectors fails to predict extrinsic performance. In: Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP. pp. 1–6. Association for Computational Linguistics, Berlin, Germany (Aug 2016). https://doi.org/10.18653/v1/W16-2501, `https://www.aclweb.org/anthology/W16-2501`

6. Dillenberger, J.: Evaluation of model and hyperparameter choices in word2vec (2019)
7. Fanaeepour, M., Makarucha, A., Lau, J.H.: Evaluating word embedding hyperparameters for similarity and analogy tasks. CoRR **abs/1804.04211** (2018), http://arxiv.org/abs/1804.04211
8. Faruqui, M., Tsvetkov, Y., Rastogi, P., Dyer, C.: Problems with evaluation of word embeddings using word similarity tasks. In: Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP. pp. 30–35. Association for Computational Linguistics, Berlin, Germany (Aug 2016). https://doi.org/10.18653/v1/W16-2506, https://www.aclweb.org/anthology/W16-2506
9. Fellbaum, C.: WordNet: An Electronic Lexical Database. Bradford Books (1998)
10. Gambäck, B., Das, A.: Comparing the level of code-switching in corpora. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16). pp. 1850–1855. European Language Resources Association (ELRA), Portorož, Slovenia (May 2016), https://www.aclweb.org/anthology/L16-1292
11. Gladkova, A., Drozd, A.: Intrinsic evaluations of word embeddings: What can we do better? In: Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP. pp. 36–42. Association for Computational Linguistics, Berlin, Germany (Aug 2016). https://doi.org/10.18653/v1/W16-2507, https://www.aclweb.org/anthology/W16-2507
12. Guzman, G.A., Serigos, J., Bullock, B.E., Toribio, A.J.: Simple tools for exploring variation in code-switching for linguists. In: Proceedings of the Second Workshop on Computational Approaches to Code Switching. pp. 12–20. Association for Computational Linguistics, Austin, Texas (Nov 2016). https://doi.org/10.18653/v1/W16-5802, https://www.aclweb.org/anthology/W16-5802
13. Huang, C., Qiu, X., Huang, X.: Text classification with document embeddings. In: Sun, M., Liu, Y., Zhao, J. (eds.) Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data. pp. 131–140. Springer International Publishing, Cham (2014)
14. Jang, B., Inhwan, K., Kim, J.W.: Word2vec convolutional neural networks for classification of news articles and tweets. PLoS ONE (2019). https://doi.org/10.1371/journal.pone.0220976
15. Krebs, A., Paperno, D.: When hyperparameters help: Beneficial parameter combinations in distributional semantic models. In: Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics. pp. 97–101. Association for Computational Linguistics, Berlin, Germany (Aug 2016). https://doi.org/10.18653/v1/S16-2011, https://www.aclweb.org/anthology/S16-2011
16. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. CoRR **abs/1405.4053** (2014), http://arxiv.org/abs/1405.4053
17. Li, P., Liu, Y., Sun, M., Izuha, T., Zhang, D.: A neural reordering model for phrase-based translation. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. pp. 1897–1907. Dublin City University and Association for Computational Linguistics, Dublin, Ireland (Aug 2014), https://www.aclweb.org/anthology/C14-1179
18. Linzen, T.: Issues in evaluating semantic spaces using word analogies. In: Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for

NLP. pp. 13–18. Association for Computational Linguistics, Berlin, Germany (Aug 2016). https://doi.org/10.18653/v1/W16-2503, https://www.aclweb.org/anthology/W16-2503

19. Mikolov, T., Chen, K., Corrado, G.S., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013)
20. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. CoRR **abs/1310.4546** (2013), http://arxiv.org/abs/1310.4546
21. Pratapa, A., Choudhury, M., Sitaram, S.: Word embeddings for code-mixed language processing. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 3067–3072. Association for Computational Linguistics, Brussels, Belgium (Oct-Nov 2018). https://doi.org/10.18653/v1/D18-1344, https://www.aclweb.org/anthology/D18-1344
22. Rijhwani, S., Sequiera, R., Choudhury, M., Bali, K., Maddila, C.S.: 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). pp. 1971–1982. Association for Computational Linguistics, Vancouver, Canada (Jul 2017). https://doi.org/10.18653/v1/P17-1180, https://www.aclweb.org/anthology/P17-1180
23. Schnabel, T., Labutov, I., Mimno, D., Joachims, T.: Evaluation methods for unsupervised word embeddings. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 298–307. Association for Computational Linguistics, Lisbon, Portugal (Sep 2015). https://doi.org/10.18653/v1/D15-1036, https://www.aclweb.org/anthology/D15-1036
24. Sinoara, R., Camacho-Collados, J., Rossi, R., Navigli, R., Rezende, S.: Knowledge-enhanced document embeddings for text classification. Knowledge-Based Systems (10 2018). https://doi.org/10.1016/j.knosys.2018.10.026