# Classifying recognised speech with deep neural networks

Rhyno A. Strydom[0000−0002−4364−2148] and Etienne Barnard[0000−0003−2202−2369]

Multilingual Speech Technologies, North-West University, South Africa;
and CAIR, South Africa.
{rhyno.strydom02, etienne.barnard}@gmail.com

**Abstract.** We investigate whether word embeddings using deep neural networks can assist in the analysis of text produced by a speech-recognition system. In particular, we develop algorithms to identify which words are incorrectly detected by a speech-recognition system in broadcast news. The multilingual corpus used in this investigation contains speech from the eleven official South African languages, as well as Hindi. Popular word embedding algorithms such as Word2Vec and fastText are investigated and compared with context-specific embedding representations such as Doc2Vec and non-context specific statistical sentence embedding methods such as term frequency-inverse document frequency (TFIDF), which is used as our baseline method. These various embedding methods are then used as fixed length input representations for a logistic regression and feed forward neural network classifier. The output is used as an additional categorical input feature to a CatBoost classifier to determine whether the words were correctly recognised. Other methods are also investigated, including a method that uses the word embedding itself and cosine similarity between specific keywords to identify whether a specific keyword was correctly detected. When relying only on the speech-text data, the best result was obtained using the TFIDF document embeddings as input features to a feed forward neural network. Adding the output from the feed forward neural network as an additional feature to the CatBoost classifier did not enhance the classifier's performance compared to using the non-textual information provided, although adding the output from a weaker classifier was somewhat beneficial.

**Keywords:** word embeddings · Word2Vec · fastText · Doc2Vec · TFIDF · Deep Neural Networks · CatBoost

## 1 Introduction

In recent years traditional statistical text representation methods in Natural Language Processing (NLP) are increasingly overshadowed by methods that incorporate Deep Neural Networks (DNNs) to encode information within words, sentences and paragraphs. Where statistical methods have failed at capturing

the regularities such as the linguistic structure within languages by representing text as sparse handcrafted feature vectors, DNNs [1] are able to represent text as dense feature vector representations. These representations are able to capture the semantic relationship between words or even different contexts between paragraphs and sentences. Mikolov's "Word2Vec" [2] showed how by representing words as continuous feature representations (word embeddings), it is possible to capture the semantics of words outperforming traditional statistical language modelling techniques such as term-frequency inverse document frequency (TFIDF) [3]. This idea is extended upon with Facebook's "fastText" [4], which learns each character in a word's orthographic representation, helping to generate better word embedding for out-of-corpus words. All these methods, however, are related in that they depend on the same distributional hypothesis which states: "words that appear in the same context share semantic meaning with each other". Mikolov expanded this idea to paragraphs and sentences with "Doc2Vec" [5], representing each document as a dense vector which is trained to predict words in a document.

In this paper, our goal is to investigate these embedding techniques to analyse text produced by a speech recognition system for all the official South African languages and Hindi. Word embeddings are used to determine whether certain keywords of interest in the news have been mentioned falsely or not, due to keywords being mistaken for a similar sounding word in one of the other languages. Identifying these false positives is particularly relevant as it can help advertising agencies know if their advertisements were aired at the right time. This capability can help ensure that these agencies maximise exposure of their clients' products at the most appropriate time of day, and also that media organisations abide by their contractual obligations to broadcast advertisements at agreed upon specified times.

Training word embeddings is an unsupervised learning task, requiring a clearly identified objective in order to evaluate how successful word embeddings are at identifying false positives. Also alterations must be made to the unsupervised learning algorithm based on how well it performs on the task at hand, which is known as extrinsic evaluation [6]. In this paper, we employ the word embeddings as input features to both a linear and non-linear classifier using logistic regression and a DNN classifier, trained to predict whether or not false positive keywords were encountered by the speech recognition system.

We also investigate an alternative method of identifying false positive keywords, where different word embeddings are trained for falsely mentioned keywords (i.e. false positives) and keywords that were correctly identified (i.e. true positives), comparing the cosine similarity between the different occurrences of these keywords.

The main contributions of this work are twofold: on the one hand, we show that text obtained from a speech recognition system provides an interesting and important use case for NLP techniques. On the other, we demonstrate the relative capabilities of various NLP solutions, including embeddings and more traditional methods on such a task in the multilingual South African context.

## 2 Related work

Text classification is the task of classifying a document to a predefined category. Examples of such task include sentiment analysis, document categorisation and language detection. Linear classifiers such as logistic regression are typically used as a baseline method for NLP problems these classifiers might be very simplistic but can sometimes yield strong baselines for comparing different text classification methods, as shown by Joachims [7] and Fan [8]. Neural networks and support vector machines (SVM) on the other hand are non-linear classifiers and able to distinguish between more complex patterns present in data. Malhar [9] shows how twitter data can be categorised using support vector machines (SVM) outperforming other machine learning methods. Manevitz et al. [10] show how they are able obtain superior results using a simple feed-forward neural network to classify documents on the standard Reuters data-base, comparing it to various traditional machine learning techniques such as Nearest Neighbour, Naive-Bayes and One-Class SVM's. Typically in automatic speech recognition systems (ASR) token identification is used as a means of identifying specific words uttered [11]. Throughout this study we apply more NLP techniques applied to plain text as a mean to recognise text after it has been identified by an ASR system.

Boosting is a method commonly used by machine learning practitioners to reduce both bias and variance of the machine learning algorithm by combining multiple weak learners (i.e. decision tree's) to make one strong classifier. CatBoost [12] is a gradient boosting algorithm developed by Yandex. They show how CatBoost is able to outperform existing state of art boosting algorithms such as XGBoost and LightGBM on various machine learning tasks. We use the output of the neural network as an additional input feature to a CatBoost [12] classifier, comparing how the CatBoost classifier performs with and without the corresponding metadata associated with the text (i.e. keyword or target word, broadcast station name, total duration of the recorded clip in minutes, station language and day of the week). TFIDF is used as a baseline method to evaluate how effectively the embeddings are able to represent the text data

## 3 Data analysis and exploration

In this section, we investigate the available data and the preprocessing techniques used to clean the text.

### 3.1 Preprocessing data

Before the text is converted to a numerical representation for the various machine learning algorithms, preprocessing is done on the corpus, shown in fig. 1.
The preprocessing steps are listed as follows:

1. Extract speech-text data from JSON files.
2. Clear the data of any unnecessary symbols and change all the words in the corpus to lower case.

3. Remove English stop-words, i.e. words that do not contribute to the overall meaning in the text, such as 'a', 'the', 'is' and 'are'.
4. Lemmatise the words in text, converting all English words to their base form, for example "geese" becomes "goose" and "caring" becomes "care".
5. Identify common phrases in text such as "New York" or "Net income", instead of having a numerical representation for each word separately, which would effectively increase the vocabulary size.
6. Extract data with different window sizes around individual keywords.

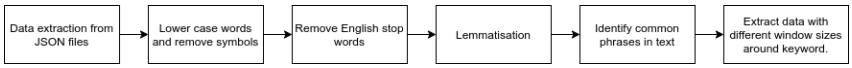



**Fig. 1:** Flow diagram detailing data extraction and preprocessing pipeline.

### 3.2 General corpus information

The speech-text data was acquired from local South African broadcasting stations and is an amalgamation of radio news and television channel data the data were manually labelled to indicate false positives by people working for the Novus Group.

The text in the data set is predominantly English; there are almost 10 times more English sentences in comparison with the second biggest language in the data set, Afrikaans, as shown in Fig. 2. There are 885,930 speech-text examples (many examples contain more than one keyword) in the corpus of which 675,425 are examples of true positives and 210,505 are true negatives (false positives from the speech-text classifier). This means that 76.24% of the data consists of true positives and 23.76% of true negatives.

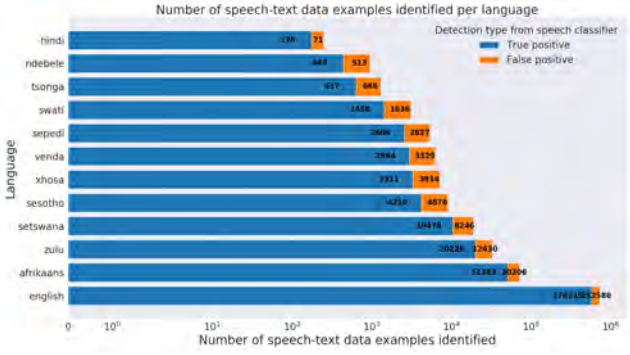



**Fig. 2:** Frequency of occurrence for different languages in the corpus.

The raw corpus consists of 248,069,944 words with a vocabulary size of 69,052. After pre-processing the word count is significantly reduced to 104,565,491 and the vocabulary size increased to 237,548. There are 4,186 unique keywords, of which 2,729 keywords have examples of where they appear in text as both true and false detection examples. 1,457 examples were only being detected once or more as either true or false.

Keyword pairs are broken into individual keywords with different word window sizes around specific keywords of interest (window size refers to the words to the left and right of the keyword). A new corpus is thus created where each keyword has its own pseudo sentence. This is done to investigate how many context words around specific keywords are needed to achieve good classification. Information regarding this newly generated corpus can be seen in table 1. It is clear from this table that as the window size increases, so does the vocabulary and word count, which is to be expected.

Table 1: Corpus information, showing window sizes, word count and vocabulary size.

| **Window size around keyword** | **Number of words in corpus** | **Vocabulary size** |
|---|---|---|
| window 5 | 7,562,980 | **105,685** |
| window 11 | 15,521,514 | **150,951** |
| window 21 | 27,244,705 | **184,033** |
| window 41 | 46,840,613 | **208,273** |
| window 61 | 63,174,235 | **217,431** |
| window 121 | 99,466,572 | **226,408** |

Fig.3 shows the most frequent keywords found in a corpus with a window size of 121. Splitting the data in this way resulted in some common nouns appearing more frequently, due to the way the data was captured. These common nouns do not contribute much to the identification of specific words and are only a result of how the data was split.

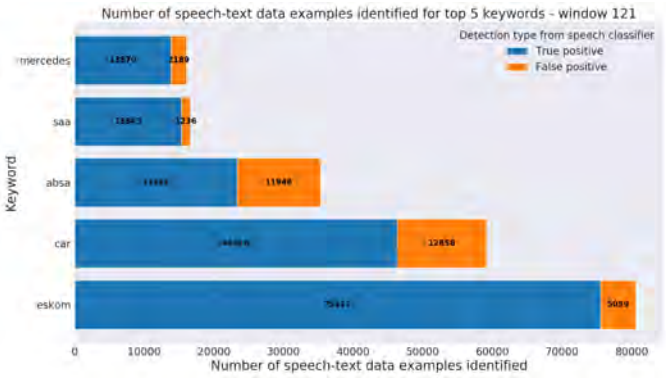



**Fig. 3:** Top 5 words found in the corpus, with window size of 121.

## 4 Experimental procedure

In this section, we investigate how word and document embeddings are trained and evaluated with the various classifiers.

### 4.1 Training, validation and test set

The data is split into a train, validation and test set by randomly splitting the data in accordance to how frequently a specific keyword appears in the corpus — 10% of the data is used for the test and validation set respectively, thus 80% of the data remains for training. We report the accuracy achieved on the test set, and do not attempt to perform statistical significance testing since our goal is to give an indication of the relative performance for various choices of algorithms and parameters, rather than a definite ranking.

### 4.2 Representing text with word and document vectors

**Word2Vec & fastText** Word2Vec is used to represent words as continuous vector representations. The algorithm has two variants, namely Continuous Bag of Words (CBOW) and Skip-Gram (SG). CBOW maximises the probability of the target word by looking at the context words, thus given a set of words at its input it aims to predict a specific word at its output. SG follows the inverse rule to this; the technique is designed to predict context given a specific target word. fastText is a technique related to Word2Vec, in that it follows the same distributional hypothesis, and can also be trained using either the CBOW or SG method. fastText claims to be able to generate better word embeddings for out-of-corpus words and morphologically rich languages as it takes into account each character in a word's orthographic representation. This enriches the word embedding, given that orthographic N-grams are able to capture some key structural components of words.

In preliminary investigations the following hyperparameters, shown in Table 2, exhibited good overall performance in extrinsic tasks. These values were empirically selected and correspond with some of the practical values suggested by Miklolov [13]. The window size was changed from 5 to 10 as it showed slightly better results and minimum word count set to 2 to neglect words being removed from South African languages with less detection examples.

Table 2: Word2Vec and fastText hyperparameters.

| **Parameter** | **Value** |
|---|---|
| Embedding dimension | 300 |
| Window size | 10 |
| Minimum word count | 2 |
| Negative sampling rate | 5 |
| Sub-sampling rate | 1e-5 |

**Doc2Vec** Doc2Vec shows it is possible to represent documents of variable lengths as fixed-length paragraph vectors. These document embeddings can then be used for text classification tasks such as sentiment analysis or predicting falsely detected words given a certain context. Document embedding also come in two variants, namely Distributed Bag of Words of Paragraph Vectors (PV-DBOW) and Distributed Memory Model of Paragraph Vectors (PV-DM). PV-DM averages or concatenates the word and paragraph vectors to predict the next word in context, and the paragraph vector that is learned acts as a memory that remembers what is missing from the current context. PV-DBOW, on the other hand, forces the model to randomly predict sampled words from the paragraph at its output; this method thus ignores word ordering at its output. Both these methods are explored and training done in a semi-supervised fashion, where each document of interest is tagged as either a true or false detection regarding a specific keyword of interest in the news.

In preliminary investigations the following document vector hyperparameters, shown in Table 3, exhibited good overall performance in extrinsic tasks. Hyperparmaters suggested by Lau [14] are used the minimum word count is set to 2 for the same reasons as the word embeddings. The document vectors were trained using a smaller embedding dimension than the trained word vectors, as no improvements were seen in classification increasing this value beyond 100 dimension.

Table 3: Doc2Vec hyperparameters.

| **Parameter** | **Value** |
|---|---|
| Embedding dimension | 100 |
| Window size | 15 |
| Minimum word count | 2 |
| Negative sampling rate | 5 |
| sub-sampling rate | 1e-5 |

**Term frequency-inverse document frequency** TFIDF is a numerical statistic used to express the importance of a word in a document when considering the whole corpus; the statistic is calculated using equation 1. Term frequency refers to the number of times a specific word appears in a document, while inverse document frequency indicates the importance of the word and is calculated by taking the number of documents in the corpus and dividing it by the number of times a specific word appeared in the corpus. This has the effect of rarer words having a larger numerical value assigned to them, the intuition being that rarer words carry more meaning in a document and should therefore be assigned a larger numerical value than words that appear more frequently and carry no meaning. This method, however, results in a sparse document representation neglecting word order in the sentence. The best results for TFIDF were obtained constraining the vocabulary to 60,000 words for the top N TFIDF scores.

$$w_{i,j} = tf_{i,j} \times \log(\frac{N}{df_i}) \tag{1}$$

where: $w_{i,j}$ = the numerical statistic of a specific word $i$ in document $j$
$tf_{i,j}$ = term frequency of word $i$ in document $j$
$N$ = the number of documents
$df_i$ = number of times word $i$ occurs in the document

## 5 Classifiers, optimization, and metrics used throughout study

### 5.1 Average cosine similarity method using keyword flags

Inspired by Word2Vec's distributional hypothesis, the average cosine similarity method uses the trained word vectors for classification. The premise is that keywords that are identified as false positives will be surrounded by similarly incorrect words that have nothing to do with the actual keyword. This essentially means that occurrences of words such as "Disney_t" (labelled with a flag "t" for true detection) will be surrounded by completely different words than those of "Disney_f" (labelled with a flag "f" for false detection), as illustrated in fig.4. During inference one would identify the keyword, using a dictionary of all the keywords of interest, and compare the cosine similarity, shown by equation 2, for each word in the sentence for both its true and false keyword examples by adding this flag. The average cosine similarity for both these cases could then be compared with one another, where if the average similarity for "Disney_t" is larger than "Disney_f", the detection is most likely a true detection, and vice versa. The output to this decision is thus binary (therefore a binary classifier). The investigation is done for different window sizes, comparing both fastText and Word2Vec, and investigating both SG and CBOW.

$$\cos\theta = \frac{a \cdot b}{\|a\|\|b\|} \tag{2}$$

where: $\cos\theta$ = cosine similarity between the keyword $a$ and arbitrary word $b$
$a$ = the keyword of interest for either its true or false examples
$b$ = an arbitrary word in the sentence

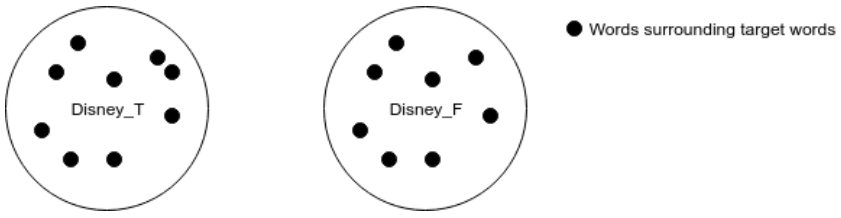



**Fig. 4:** True detection word embedding cluster for the word Disney (shown left) and false positive word embedding cluster for the word Disney (shown right).

### 5.2 Logistic regression, DNN and CatBoost classifier

Standard machine learning algorithms require that their input be represented as a fixed length feature vector. Word2Vec and fastText, however, only represent the vectors of individual words. Document vectors are therefore created by taking the average of each word vector in a sentence. Doc2Vec, Word2Vec, fastText and TFIDF are investigated using a logistic regression (linear classifier) and a one layer feed forward neural network classifier (non-linear classifier) with varying widths, also a class weighted logistic regression and feed forward neural network is used, to mitigate the effect of having an unbalanced data set. A heuristic is used to determine the weighting, as shown by equation 3. This weight value is multiplied to each classifier's loss-function, giving more weight to infrequent classes than frequent ones.

After the logistic regression and DNN classifier made its predictions based only on the information provided by text, it is combined with the metadata associated with this prediction, such as the broadcast-station name, keyword of interest, language and time of day, to enhance the classification performance of the CatBoost model, instead of training only on the metadata and ignoring the text associated with it. The same heuristic shown in equation 3 was used to counter the effect of class imbalance.

$$\text{class-weight} = \frac{\text{total number of samples}}{\text{number classes} \times \text{number of samples of specific class}} \tag{3}$$

### 5.3 Metrics used for evaluation and optimisation

The various classifiers are evaluated based on the Cohen Kappa [15] statistic. It is used to compensate for the effect of the class imbalance present in the data, and it measures the inter-rater agreement for categorical items. To evaluate feature importance in the CatBoost classifier, mean absolute SHAP [16] values are used. The shapely values are calculated by comparing what the model predicts with and without a specific categorical feature, and this is then used to determine which features are relevant and which are not.

Both the CatBoost and DNN classifier use early stopping based on the Area Under the Curve (AUC) value to counter over-fitting. The DNN uses the Adaptive Moment Estimation (Adam) [17] optimiser with a batch size 1024 and a learning rate of 0.001, including a rectified linear unit (ReLU) function before the softmax layer.

## 6 Results

Using TFIDF, Word2Vec, fastText and Doc2Vec embeddings as input features to a logistic regression (LR) and DNN classifier, the following results were obtained for different word window sizes. It should be noted that initial investigations found that Word2Vec and fastText models trained using the SG method performed better than models trained using the CBOW method. Similarly, Doc2Vec

methods trained using the PV-DBOW method performed significantly better compared to PV-DM. The results of these methods are therefore not presented in the following tables, they are, however, present for the average cosine-similarity method.

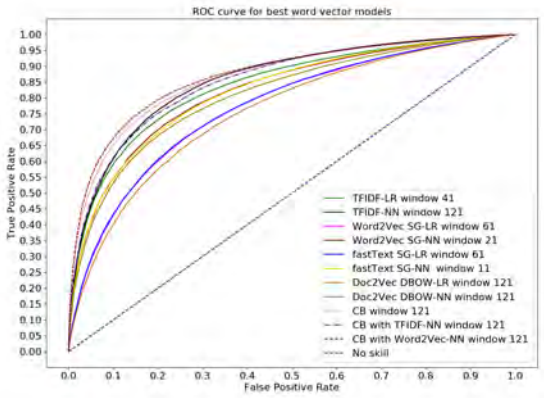
Table 4: Cohen kappa statistic on validation set using optimal window size, for various classifiers .

| | **Classifiers** | | | | | |
|---|---|---|---|---|---|---|
| Embedding method | LR | DNN-100 | DNN-200 | DNN-400 | Avg Cosine similarity method | CatBoost |
| **Window size 11** | | | | | | |
| fastText SG | - | 0.38563 | 0.39127 | 0.39540 | - | - |
| **Window size 21** | | | | | | |
| fastText SG | - | - | - | - | 0.36370 | - |
| Word2Vec SG | - | 0.38288 | 0.39344 | 0.39876 | - | - |
| **Window size 41** | | | | | | |
| TFIDF | 0.43271 | - | - | - | - | - |
| fastText CBOW | - | - | - | - | 0.22921 | - |
| **Window size 61** | | | | | | |
| fastText SG | 0.32488 | - | - | - | - | - |
| Word2Vec SG | 0.32436 | - | - | - | 0.32971 | - |
| Doc2Vec DBOW | - | 0.37808 | 0.38126 | 0.38343 | - | - |
| **Window size 121** | | | | | | |
| No embedding (Baseline) | - | - | - | - | - | **0.49639** |
| TFIDF | - | 0.45667 | 0.47026 | 0.47310 | - | **0.46679** |
| Word2Vec CBOW | - | - | - | - | 0.30704 | - |
| Word2Vec SG | - | - | - | 0.36289 | - | - |
| Doc2Vec DBOW | 0.30610 | - | - | - | - | - |

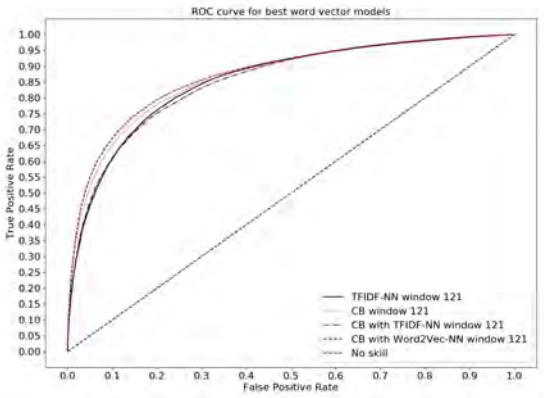
Table 5: Cohen kappa statistic on test set using optimal window size.

| | **Classifiers** | | | | | |
|---|---|---|---|---|---|---|
| Embedding method | LR | DNN-100 | DNN-200 | DNN-400 | Avg Cosine similarity method | CatBoost |
| **Window size 121** | | | | | | |
| No embedding (Baseline) | - | - | - | - | - | **0.49481** |
| TFIDF | - | - | - | 0.47209 | - | **0.46465** |
| Word2Vec SG | - | - | - | 0.36128 | - | **0.51399** |

The Receiver Operating Characteristic (ROC) curves for the logistic regression, DNN and CatBoost classifiers, are shown in fig.5(a)-(b). Note that only the 400 hidden node DNN classifier results are included in the figures.



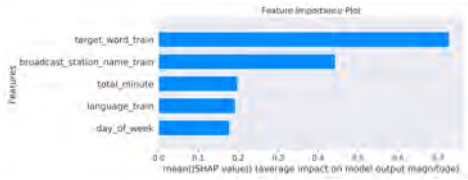

**(a)** ROC curves evaluated on validation set.



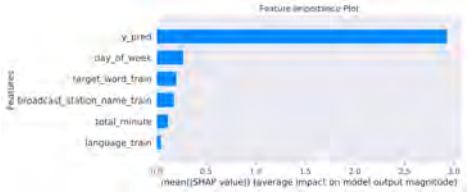

**(b)** ROC curves evaluated on test set.

**Fig. 5:** ROC plots for validation train and test set for CatBoost (shown as CB), neural network (shown as NN) and logistic regression (shown as LR) classifier using different embedding techniques.

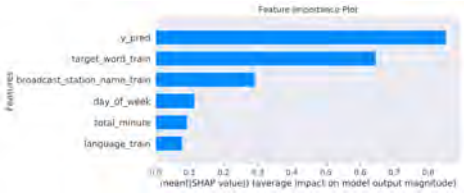
The mean absolute SHAP values were used to evaluate the contribution of the CatBoost classifier's features to the output. This is shown in fig.6.




**(a)** SHAP values for baseline Catboost classifier.




**(b)** SHAP values for Catboost classifier with added output prediction from 400 node feed forward neural network using TFIDF.




**(c)** SHAP values for Catboost classifier with added output prediction from 400 node feed forward neural network using Word2Vec.

**Fig. 6:** Mean absolute SHAP values of CatBoost classifier, evaluated on test set.

## 7 Discussion

When comparing the results of the logistic regression classifier in table 4 with the 100 hidden node feed forward neural network classifier, the logistic regression classifier shows poor performance using Word2Vec, fastText and Doc2Vec as the textual representations in comparison with the neural network. The discrepancy between the two classifiers is due to the neural network's ability to better capture the non-linearity of these embeddings compared to that of the logistic regression classifier. Generally it also seems that larger window sizes for Doc2Vec generate better embeddings for extrinsic tasks, although the same is not always true for Word2Vec and fastText: After a certain point, averaging word vectors over larger window sizes result in some information loss with regards to important words in the context window, which might indicate a false-positive or not. A window size of either 11 and 21 for Word2Vec and fastText regularly produces good results. Increasing the capacity of the feed forward neural network with more hidden nodes, increases the Cohen Kappa metric, although the performance starts to plateau after increasing the neural network size beyond 200 hidden nodes. TFIDF, used as a baseline method to evaluate the word embeddings, outperforms all other embedding methods. This is due to the fact that word and document vectors contain more latent information compared to TFIDF, thus the model might over-fit on features in the textual representations that are less important, whereas TFIDF is able to capture more clearly these important features in text. Training word vectors on a larger text corpus should result in better word embeddings. Furthermore, using larger window sizes with TFIDF (capturing more textual data) tends to improve classification performance.

The baseline CatBoost classifier achieves good results by relying on the target word and broadcast station name as its most important input features, shown using SHAP values in Fig. 6.(a). Adding the output from the 400 hidden node feed neural network classifier using TFIDF method, however, degrades the model's performance, shown in table 5. The model also becomes less reliant on the broadcast station name and target word as important input features, and sees the neural network's output as the most important indicator of whether a false positive or true detection was encountered, shown in Fig. 6.(b), over-fitting the CatBoost classifier to that specific feature. Using the output of the 400 hidden node neural network with Word2Vec (i.e. a weaker model), however, slightly improves the CatBoost classifier performance, with the output from the neural network shown as the most important of the three dominant features (i.e. "y_pred", "target_word_train" and "broadcast_station_name_train"), shown in fig. 6.(c). The broadcast station's language, day of the week and total minutes do not have much effect on the classification result, and these features can thus be removed.

The average cosine similarity method, used to classify words based on true and false flags, achieved better performance than the logistic regression classifier, but worse performance compared to the neural network. Using the CBOW method performance increased as the window size increased, but results were

never as good as using the SG method. The optimal window size for this method was 21; increasing it showed no further performance benefit.

## 8 Conclusion

In this study, different word embedding techniques were investigated employing a DNN and logistic regression classifier, adding the best textual representation output of the DNN classifier as an additional input feature to a CatBoost classifier. It was found that applying TFIDF as the textual representation with a neural network, outperformed all other embedding based methods. TFIDF is able to clearly capture important features in text that the machine learning algorithm can use to discriminate between a false and true positive word. This is due to the limited amount of data and domain specific nature (i.e. broadcast speech-text data) of the task. Training Word2Vec, Doc2Vec and fastText embeddings with a larger corpus results in better word embeddings, increasing the models' chances of detecting a falsely mentioned word. Considering the output of the neural network using TFIDF as an additional feature to the CatBoost classifier degraded its performance. The CatBoost classifier over-fitted on this feature, neglecting additional information provided such as the broadcaster's name and keyword of interest. Considering the output of the DNN using a weaker textual representation method such as Word2Vec slightly enhanced the models performance compared to the baseline CatBoost classifier relying on the metadata associated with the speech-text data. In preliminary investigations using the SG over CBOW based method for Word2Vec and fastText showed slightly better results. This is attributed to SG giving more attention to rarer words in comparison with CBOW which attempts to predict the word given the context. SG furthermore forces the model to learn the context of a specific word. CBOW will thus perform better on a larger training set compared to SG which does not average out peculiarities of a specific word given a context window. The same observations are made using PV-DM with Doc2Vec. It should also be noted that some labelling errors were found, especially with regards to car names, such as "bmw", where the data was labelled as false, but given the context and video clip it is clearly true detection. These types of examples can influence the performance of classifiers trained on such material, reducing the identification accuracy for some keywords of interest.

## 9 Acknowledgements

We thank the Novus Group for providing the data needed for this investigation. Also a special thanks must be given to Dr. Charl van Heerden at Saigen for facilitating this arrangement and producing the speech-recognition results.

## References


[1] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [Review Article]," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018, ISSN: 15566048. DOI: 10.1109/MCI.2018.2840738.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Proceedings of Workshop at ICLR*, vol. 2013, Jan. 2013.

[3] R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?" *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1270–1278, 2000.

[4] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017. DOI: 10.1162/tacl_a_00051.

[5] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," *31st International Conference on Machine Learning, ICML 2014*, vol. 4, May 2014.

[6] 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. DOI: 10.18653/v1/D15-1036. [Online]. Available: https://www.aclweb.org/anthology/D15-1036.

[7] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *ECML*, 1998.

[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, Aug. 2008. DOI: 10.1145/1390681.1442794.

[9] M. Anjaria and R. M. R. Guddeti, "Influence factor based opinion mining of twitter data using supervised learning," *2014 Sixth International Conference on Communication Systems and Networks (COMSNETS)*, pp. 1–8, 2014.

[10] L. Manevitz and M. Yousef, "One-class document classification via neural networks," *Neurocomputing*, vol. 70, pp. 1466–1481, Mar. 2007. DOI: 10.1016/j.neucom.2006.05.013.

[11] D. Ram, A. Asaei, and H. Bourlard, "Sparse subspace modeling for query by example spoken term detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 6, pp. 1130–1143, 2018. DOI: 10.1109/TASLP.2018.2815780.

[12] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: Unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Cur-

ran Associates, Inc., 2018, pp. 6638–6648. [Online]. Available: http://papers.nips.cc/paper/7898-catboost-unbiased-boosting-with-categorical-features.pdf.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2013, pp. 3111–3119. [Online]. Available: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

[14] J. H. Lau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation," in *Proceedings of the 1st Workshop on Representation Learning for NLP*, Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 78–86. DOI: 10.18653/v1/W16-1609. [Online]. Available: https://www.aclweb.org/anthology/W16-1609.

[15] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, pp. 37–46, 1960.

[16] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: https://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[17] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, Dec. 2014.