A word embedding trained on South African news data

Martin Canaan Mafunda
Doctoral Candidate, Department of Physics, University of KwaZulu-Natal, Westville Campus, Durban
https://orcid.org/0000-0001-9008-5834

Maria Schuld
Senior Researcher and Software Developer, Xanadu Quantum Technologies, Toronto; and Researcher, University of KwaZulu-Natal, Westville Campus, Durban
https://orcid.org/0000-0001-8626-168X

Kevin Durrheim
Distinguished Professor, Department of Psychology, University of Johannesburg
https://orcid.org/0000-0003-2926-5953

Sindisiwe Mazibuko
Doctoral Candidate, Department of Psychology, University of KwaZulu-Natal, Pietermaritzburg Campus, South Africa
https://orcid.org/0000-0003-4376-4230

Abstract

This article presents results from a study that developed and tested a word embedding trained on a dataset of South African news articles. A word embedding is an algorithm-generated word representation that can be used to analyse the corpus of words that the embedding is trained on. The embedding on which this article is based was generated using the Word2Vec algorithm, which was trained on a dataset of 1.3 million African news articles published between January 2018 and March 2021, containing a vocabulary of approximately 124,000 unique words. The efficacy of this Word2Vec South African news embedding was then tested, and compared to the efficacy provided by the globally used GloVe algorithm. The testing of the local Word2Vec embedding showed that it performed well, with similar efficacy to that provided by GloVe. The South African news word embedding generated by this study is freely available for public use.

Keywords

natural language processing (NLP), word embedding, Word2Vec, GloVe, news data, South Africa

Acknowledgements
The authors thank Media Monitoring Africa (MMA) for making the data used for this project freely available. We are also grateful for support from the University of KwaZulu-Natal's Big Data and Informatics' Research Flagship, South Africa's National Research Foundation (NRF-Grant UID: 137755), and the South African Centre for Digital Language Resources (SADiLaR-Grant #OR-AAALV). SADiLaR is a national centre supported by the South African Department of Science and Innovation (DSI).

DOI: https://doi.org/10.23962/ajic.i30.13906

Recommended citation
Mafunda, M. C., Schuld, M., Durrheim, K., Mazibuko, S. (2022). A word embedding trained on South African news data. *The African Journal of Information and Communication (AJIC), 30*, 1-24. https://doi.org/10.23962/ajic.i30.13906

1. Introduction
Word embeddings are finding increasing use in the social sciences as tools to analyse social groups through the language they produce. They are computer models that use machine learning to develop representations of words as vectors or points in a high-dimensional space. The points are constructed so that relations between words, such as the use of two words in a similar semantic or grammatical context, can be measured as a distance between two points in the space. This gives rise to an "arithmetic of meaning".

The use of word embeddings as tools for studying culture and language is acknowledged as a new, emerging field of research (Arseniev-Koehler & Foster, 2020; Kozlowski et al., 2019). The ever-growing existence of large but "messy" pools of textual data harvested from social and traditional media is driving interest in word embeddings as key mechanisms for natural language processing (NLP). Word embeddings are used by companies such as Facebook to tag harmful posts, e.g., content written with the sole purpose of spreading false or misleading information on COVID-19 vaccination programmes. Badri et al. (2022) have demonstrated the role of word embeddings in text tagging or text detection. Their study uses fastText and GloVe word embeddings to detect offensive and hate speech in social media content.

The meaning captured by word embeddings is specific to the data that the machine-learning algorithm (model) is trained on. The development, training, and evaluation of word embedding models must therefore be context-specific. Examples of word embeddings linked to a certain domain are: the NukeBERT model (Jain et al., 2020) that is trained on texts from the nuclear and atomic energy section; specialised embeddings for finance (Theil et al., 2020); and embeddings trained on certain languages, such as Setswana and Sepedi (Marivate et al., 2020) or Croatian (Svoboda & Beliga, 2017).

Even when a widely spoken language such as English is used in a dataset, geographic contexts will induce specific terms or relationships between words that are of critical importance to researchers in fields such as social and political sciences. For instance, political scientists have used word embeddings to recover rich knowledge, through semantic projections, about the behaviour of the main political parties in South Africa in respect of illegal foreign nationals (Grand et al., 2022). In addition, Durrheim et al. (2022) have demonstrated how word embeddings provide a useful tool to study cultural bias, showing that calculating the difference between two bipolar bias vectors (centroids) gives rise to another vector which represents a bias dimension. Other researchers have used the bias dimension to study stereotypes in word embeddings (Kozlowski et al., 2019).

The need to gather knowledge that is unique to a specific field or research area is what motivated us to carry out this study, which developed and evaluated a new word embedding trained on a large corpus of online South African news articles from outlets including *Daily Maverick*, *News24* and *Independent Online (IOL)*. The embedding was trained using Word2Vec's Skip-Gram algorithm (Mikolov et al., 2013), and the dataset used was provided by Media Monitoring Africa (MMA). The word embedding we generated is publicly available via a github repository.[1] It is, to the best of our knowledge, the first publicly available word embedding trained on South Africa news article data, and thus forms a valuable addition to the field of NLP in African contexts (Marivate et al., 2020). The embedding will allow researchers to investigate the meanings of numerous words from within a South African context and to seek answers to culturally or politically oriented South African research questions—such as, to give but one small example, how the African National Congress (ANC) and Democratic Alliance (DA) relate to terms such as "corruption" and "white monopoly capital".

This article introduces the word embedding and explains the choices we made in data preprocessing and in training of the Word2Vec algorithm that generated the embedding. We also present results from extensive validation testing of the embedding, and comparative testing between the performance of our locally generated Word2Vec

1 https://github.com/Mafunda/SouthAfricanNewsEmbeddings

embedding and an embedding generated by the internationally recognised GloVe algorithm. We conducted the comparison using 14 standard analogy benchmark tasks, and found that our local South African Word2Vec embedding scored very competitively with the GloVe embedding—and in some cases scored better.

Section 2 of this article describes the Word2Vec and GloVe algorithms; section 3 sets out approaches to evaluating word embeddings; section 4 describes the preparation and configuration of the dataset; section 5 describes the implementation, evaluation, and refinement of the word embedding; section 6 describes our work to maximise robustness of the embedding through determining variances and testing ensembles of embeddings; section 7 provides results from our comparative evaluation of the performance of our South African Word2Vec embedding against the performance of a GloVe embedding; section 8 provides findings from validation of our local embedding against South African benchmarks; and section 9 provides conclusions.

2. The Word2Vec and GloVe algorithms

Word2Vec

Word2Vec is a common algorithm for training word embeddings and is powered by the statistical power of neural network models. It was first introduced in 2013 by Tomas Mikolov and his research collaborators from Google. In our study, the Word2Vec algorithm was used to learn a word embedding from a South African news articles database.

This Word2Vec algorithm consists of two model architectures and two training methods. The two model architectures are Skip-Gram and CBOW (continuous bag of words), while the two training methods are the hierarchical softmax and negative sampling. The Skip-Gram model aims to predict context from a given word. Skip-Gram is slow, and good at learning infrequent words. On the other hand, the CBOW aims to predict a word from a given context of words. CBOW is fast, and is good at learning common words. The hierarchical softmax is good at training with infrequent words, and negative sampling is good at training with common words and low-dimension vectors.

Word2Vec is similar to other commonly used approaches for learning word embeddings such as GloVe (global vectors for word representation) (Pennington et al., 2014), BERT (bi-directional encoder representations from transformers) (Devlin et al., 2018), GPT (generative pre-trained transformer) (Radford et al., 2018), fastText (Bojanowski et al., 2017; Santos et al., 2017), and ELMo (embeddings from language model) (Peters et al., 2018), to name just a few. Since one goal of this study was to compare the performance of the Word2Vec and GloVe algorithms, we now briefly review the GloVe model.

GloVe (global vectors for word representation)

GloVe, like Word2Vec, is an unsupervised learning algorithm for generating word embeddings. According to the model's developers, Pennington et al. (2014), GloVe is a count-based, global log bilinear regression model that combines two embedding methods, namely global matrix factorisation and local context window. The model is based on the observation that the most appropriate starting point for word vector learning is the ratios of co-occurrence probabilities rather than the probabilities themselves. In other words, the GloVe model is built on the intuition that the ratios of co-occurrence probabilities among words potentially encode some kind of a relation among words.

3. Evaluation of word embeddings

To ensure that word embeddings are useful and can be deployed to solve downstream NLP tasks, the quality and reliability of a word embedding needs to be assured through validation tests. Several approaches to evaluating the quality of word embeddings have been reported. Bakarov (2018) divides the methods of evaluation into two categories, namely: (1) extrinsic; and (2) intrinsic.

According to Bakarov (2018), methods of *extrinsic evaluation* are anchored on the idea that every downstream NLP task is a form of word embedding evaluation. In other words, methods of extrinsic evaluation entail leveraging the potential of word embeddings to be used as feature or input vectors when training supervised machine-learning algorithms (like the maximum entropy model). Therefore, a rule of thumb for methods of extrinsic evaluation is that any downstream NLP task can be considered as an evaluation method, e.g., for the task of sentiment analysis, text classification, or part-of-speech tagging, to mention only a few (see Bakarov (2018) for more examples).

The methods of *intrinsic evaluation*, on the other hand, involve experiments which are designed to compare word embeddings with human judgments on word relations. This was of particular interest to our study because we made use of locally inspired analogy tasks—e.g., matching politicians to political parties—for model evaluation based on South African news article data. According to Bakarov (2018), methods of intrinsic evaluation are divided into four sub-categories: (1) methods of conscious evaluation; (2) methods of subconscious evaluation; (3) thesaurus-based methods; and (4) language-driven methods. In this study, we used methods of conscious evaluation to evaluate the South African news word embedding and therefore we now limit our discussion to describing those methods.

According to Bakarov (2018), the core methods of conscious evaluation are (1) word semantic similarity, (2) word analogy, (3) thematic fits, and (4) synonym detection. The *word semantic similarity* method is based on the idea that distances between words in an embedding space can be evaluated through the human heuristic judgments on the actual semantic distances between these words. For example, we would expect the distance between *cup* and *mug* defined by a number from the interval [0, 1] to be in the region of 0.8 since these words are nearly synonymous, that is, they are used similarly in language.

The *word analogy* method is the second most popular method for evaluating word embeddings (Bakarov, 2018). First introduced by Mikolov et al. (2013), word analogies are based on the idea that arithmetic operations in a word vector space can be predicted by humans. For instance, given a set of three words or word pairs—e.g., the two politicians "Julius Malema" and "Jacob Zuma", as well as the party "EFF" (Economic Freedom Fighters, founded by Malema)—the task would be to predict the word D such that the relation Julius_Malema : EFF is the same as the relation Jacob Zuma : D (Pereira et al., 2016; Turian et al., 2010). In this case, the target word would be "ANC" (African National Congress), which is the party of ex-President Jacob Zuma. Word analogies are also known as "analogical reasoning", "linguistic regularities", and/or "word semantic coherence". In this study, we used both word semantic similarity and word analogy methods to evaluate the quality of our South African news embedding.

4. Dataset preparation and configuration

Data

This study used a text corpus of 1,312,125 news articles, which were provided, upon request, by MMA from its news database. The text dataset consisted of news articles that were published between 1 January 2018 and 17 March 2021. It should be noted that the database was not in the public domain, and access was granted in response to our individual request.

Data preparation

Raw texts are by nature "noisy" and therefore require some text preprocessing before they can be used to train machine-learning algorithms such as the Word2Vec model. Text preprocessing for this study was done with the help of several open source

Python software packages, including the natural language toolkit (NLTK) (Loper & Bird, 2002), beautifulsoup (Richardson, 2007), and gensim (Řehůřek &, Sojka, 2011a). The sequence of preprocessing steps included: splitting documents (multi-sentences) into single sentences (also known as sentence tokenisation); removing all words containing single uppercase letters surrounded by lowercase letters in order to remove JavaScript; and converting all words to lowercase letters. Further, preprocessing included the removal of: html tags; expressions such as "\xad" and "displayad"; words that contained substrings ("windowtextcolor"), and punctuation and digits.

We did not remove stopwords, following a growing trend in the machine-learning literature. Rahimi and Homayounpour (2022) recommend the retention of stopwords when learning word representations for solving sentiment classification problems. This is because the removal of stopwords such as "no" and "don't" can potentially change the polarity of words in documents.

Data preparation also included the creation of n-grams (bigrams and trigrams) using the Phraser model of the gensim package. Bigrams are pairs of words that are repeatedly mentioned together in a given text corpus. For example, during our data preparation, words such as Jacob and Zuma were joined to produce a bigram Jacob_Zuma because they occurred together more than our determined minimum threshold of collocations. Similarly, we joined three words together into a trigram if they consecutively and consistently occurred together within the news articles corpus. For example, the word combination President Jacob Zuma was joined to produce a President_Jacob_Zuma trigram.

We conclude this section with an example of a "messy" text followed by its "clean" version after data preprocesing:

- *Before preprocessing:* PRESIDENT Jacob Zuma has declared a special official funeral for renowned author and poet, Prof. William Keorapetse Kgositsile, a renowned veteran activist and a giant of the liberation struggle who died on Wednesday.
- *After preprocessing:* president jacob zuma has declared a special official funeral for renowned author and poet prof william keorapetse kgositsile a renowned veteran activist and a giant of the liberation struggle who died on Wednesday.

Hyperparameter settings

Table 1 shows the hyperparameter names and values used to train the embedding (with Python's gensim package). As mentioned in section 2, the Word2Vec algorithm learns word embeddings using one of its two model architectures: Skip-Gram or CBOW. In our study, we used Skip-Gram. Also as mentioned above, Word2Vec uses two training methods to learn word embeddings: hierarchical softmax (Goodman, 2001) and negative sampling (Mikolov et al., 2013). We adopted negative sampling, and a hyperparameter negative value of 10. The role of the hyperparameter "negative" is to specify the number of "noise words" that the model is allowed to draw on during model training.

Table 1: Hyperparameter settings used to train the embedding

Parameter name	Value
minimum word count (m)	50
window size	10
architecture	Skip-Gram (s1)
training method	negative sampling (h0)
negative	10
vector dimension size (d)	250

For two of the hyperparameters—minimum word count and vector dimension—the hyperparameters seen in Table 1 (50 and 250, respectively)—were only finalised through experiments conducted on the initial embedding (see section 5).

5. Implementation, evaluation, and refinement of the word embedding

Implementation of the embedding

The gensim package (Řehůřek & Sojka, 2011b), implementable in the Python environment, was used to build and train the Word2Vec algorithm. The popularity and convenience of implementing the Word2Vec algorithm with gensim influenced our decision to select this implementation framework.

We used Google Colaboratory, an online environment for Python programming, to implement the Word2Vec algorithm with gensim. It took approximately eight hours to implement the embedding, starting from data preparation until the model finished training. Due to the large dataset size, we used a procedure in which data was read in chunks of 10,000 sentences into a buffer holding 100,000 sentences, and after each read-in, the buffer was shuffled. This introduced a pseudo-randomness in which the first sentence in the corpus had a greater chance of being fed to the training procedure early on.

Performance evaluation measures

We measured the performance of the embedding using both "similarity" and "analogy" measures.

Similarity measure

The similarity measure probes the extent to which words are similar or dissimilar by measuring the distance between their respective vector representations in an embedding. More precisely, this measure typically uses the cosine similarity, or the size of the angle between two vectors belonging to any two given words, as a proxy for measuring the degree to which the two words are related. Given any two word vectors vec(word$_1$) and vec(word$_2$), where vec(word$_i$) is the vector corresponding to a given word, the similarity value is computed as follows:

$$similarity(word_1, word_2) = \frac{vec(word_1)vec(word_2)}{||vec(word_1)||||vec(word_2)||} \quad \textbf{(1)}$$

Note that we normalised word vectors in the embedding, so that their norm is always 1. Equation (1) implies that highly similar words (or synonyms) have similarity values that are closer to 1, while highly dissimilar words have similarity values closer to –1. We used the WordSim353 dataset to evaluate our embedding. WordSim is a test dataset for measuring word similarity or relatedness (Agirre et al., 2009). The WordSim dataset consists of word pairs such as soccer and football, baseball and netball, etc. and their similarity scores. (The WordSim353 dataset is freely available for public use.[2])

Analogy measure

Analogy measurement tasks ask the embedding to predict the fourth word in a relational equation of the form "ANC is to Jacob_Zuma as EFF is to Julius_Malema". In a relational task, the model is given the first three words and asked to predict the fourth word that will solve the relational equation, i.e., Jacob_Zuma – ANC = [predict word] – EFF. A prediction is computed by retrieving the 10 nearest neighbours to the vector vec(Jacob_Zuma) - vec(ANC) + vec(EFF). If the correct word is found to be among these 10 nearest neighbours, the model is given a score of 1 (correct prediction). Otherwise, a score of 0 (incorrect prediction) is given. This

2 The WordSim353 dataset is at http://alfonseca.org/pubs/ws353simrel.tar.gz

method is commonly known as the accuracy@k method (Xu, 2018). It is called the accuracy@k method because the value of k is arbitrarily chosen and it measures the extent to which the model is penalised for producing k nearest neighbours.

The precision score for a set of analogies is then computed as follows:

$$\text{precision score} = \frac{\text{number of analogies correctly predicted by the model}}{\text{total number of tasks evaluated}} \times 100\% \qquad \textbf{(2)}$$

For analogy measurement, we used the GloVe word analogy dataset. Publicly accessible via the GloVe website,[3] the dataset is made up of 14 analogy tasks, which are named as follows:

- capital-common-countries;
- capital-world;
- city-in-state;
- currency;
- family;
- adjective-to-adverb;
- opposite;
- comparative;
- superlative;
- present participle;
- nationality adjective;
- past tense;
- plural; and
- plural verbs.

Word analogies are relational equations of the form word1:word2::word3:word4 (translated verbally as word1 is to word2 as word3 is to word4). To restate, our goal in testing the South African news embedding's ability to solve analogy tasks was to measure how well the embedding predicted the fourth word ("word4").

In reporting the experiment results in this article, we use the following notations:

- "p" stands for %;
- "d" stands for dimension size; and
- "m" stands for "minimum word count".

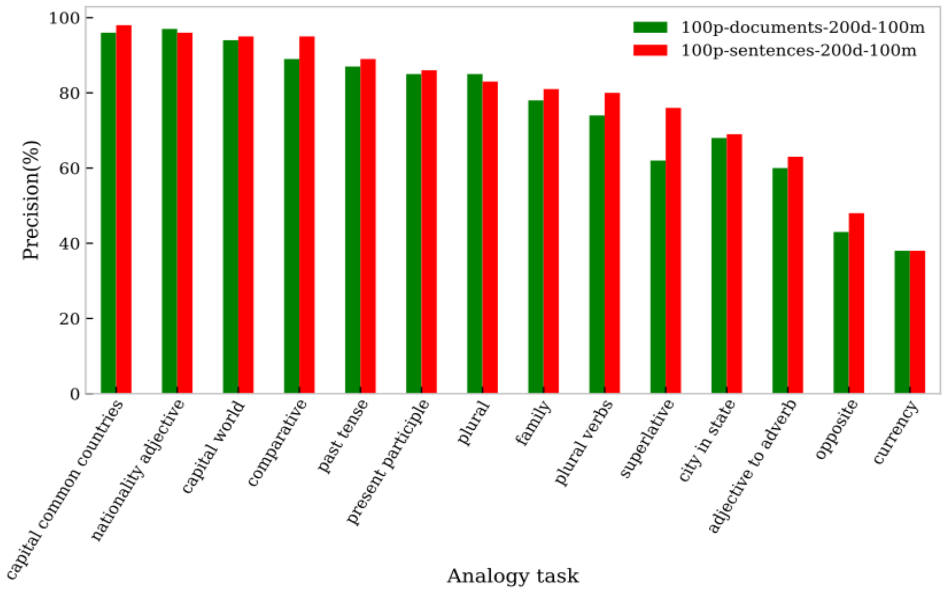
3 https://github.com/stanfordnlp/GloVe/tree/master/eval/question-data

For example, the notation "100p 100d 50m" denotes our word embedding trained on 100% (p) of the training dataset, with a word vector dimension size (d) of 100, and with words (tokens) with a minimum word count (m) of 50 (meaning that words not appearing 50 or more times were ignored during training). The reason for adopting 50 as the minimum word count is given below in the "determining a suitable minimum word count" sub-section.

Determining whether to train with sentences or documents

We conducted an experiment in order to determine whether the optimal training approach for our embedding was: (1) training based on data split into documents of news articles; or (2) training based on data split into sentences. As seen in Figure 1, we found that the precision of the word embedding trained on sentences was always the same or better than that of the word embedding trained on documents, with the sentence-contexts outperforming document-contexts in 11 (almost 80%) of the 14 analogy tasks. This finding was consistent with emerging best practices in the NLP literature (Gu et al., 2018).

Figure 1: Training with sentences versus documents (evaluated via 14 analogy tasks)



Refinement of hyperparameters through testing

As mentioned above, two of the hyperparameters could only be finalised once the embedding had been generated—allowing testing of the influence of different parameter settings on the embedding. We conducted experiments that measured the embedding's precision in conducting 14 analogy tasks when the value of a certain hyperparameter was varied.

Determining a suitable vector dimension size

We conducted a second experiment in order to determine the optimal dimension size, i.e., the size or dimensionality of the word vectors in the trained embedding. Identifying optimal dimensionality is important since it influences the space available to "encode meaning": a low dimension may result in under-fitting, a situation where there is not enough space to reflect the subtle levels of meaning, while a dimension that is too large may lead to model over-fitting, where all words are positioned far from each other and relational meaning is lost (see also Yin & Shen, 2018). To understand the impact of the vector dimension, we trained and compared four versions of our word embedding, with each version having the same training settings except for the dimension size, which was varied for the values 100, 200, 300 and 400.

Figure 2: Training with 4 different vector dimension sizes (evaluated via 14 analogy tasks)

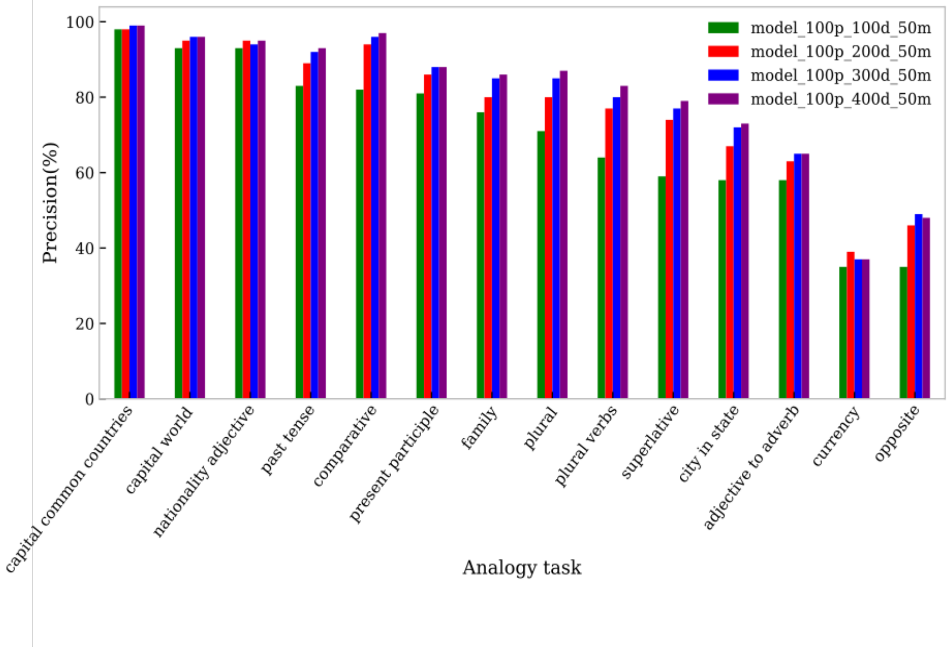


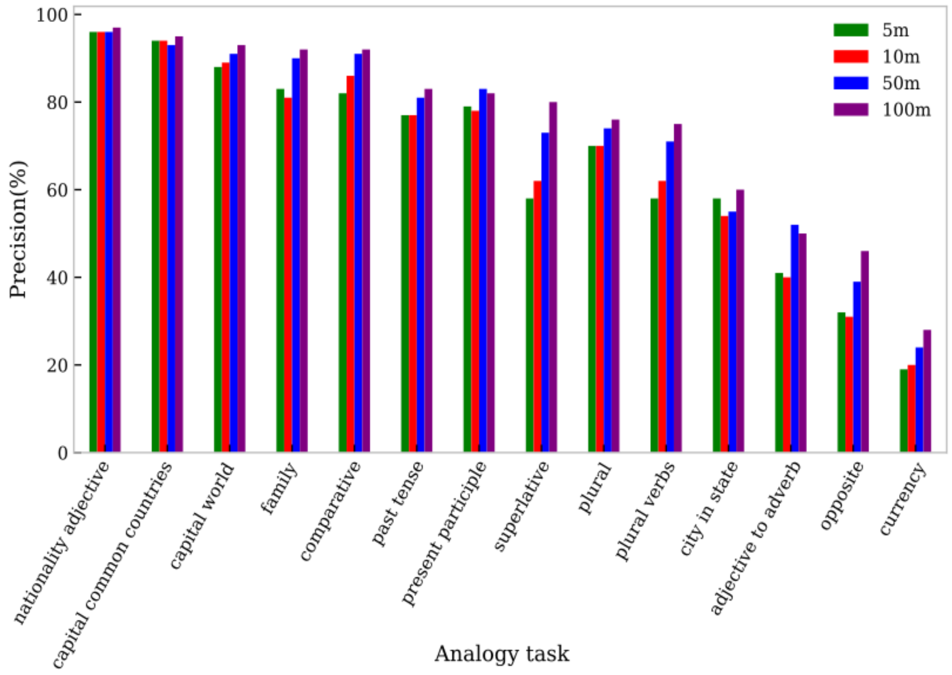
Figure 2 shows that while the precision consistently increased with higher dimensions, there was only a small improvement between 200 and 300, as well as a negligible improvement between 300 and 400. At the same time, due to the large vocabulary, 100 additional dimensions translated to 1.24 x 10^7 additional values (for a vocabulary of 124,000 words) that would have to be stored to describe the word vectors. We therefore decided to fix the vector dimension size at 250 in order to balance the physical size of the model with performance needs. This decision was justified on the grounds that we were reliant on the free version of Google's Colaboratory (Colab) platform to train and evaluate our models, and thus fixing the vector dimension size at 250 was necessary in order to reduce computational and time resources required to train and evaluate embeddings. (The free version of Colab is limited in that one cannot leave the code running without being monitored, i.e., the code has to be constantly monitored to avoid premature termination of the task being executed.)

Determining a suitable minimum word count

We conducted a third experiment in order to determine the most suitable word count for the embedding. The minimum word count sets the minimum word frequency for pruning the vocabulary available for model training. All words with minimum word count below a given specified threshold are ignored during training (Řehůřek & Sojka, 2011b). A high minimum word count leads to embeddings with a smaller but more robust vocabulary. However, such small models may not contain the words that researchers require in applications, and we found that even words used in the analogy test set (such as "policewoman") quickly became ignored if the minimum word count was too high.

Figure 3 shows the precision results of the four word embeddings, which were trained using the same hyperparameter settings except for the minimum word counts (which were set to 5, 10, 50 and 100, respectively).

Figure 3: Training with 4 different minimum word counts (evaluated via 14 analogy tasks)



It is evident in Figure 3 that the quality of the word embedding increased with increasing minimum word count. Nonetheless, since we intended to make the embedding publicly available for research, we decided to fix the minimum word count threshold at 50, which ensured that most of the words found in the vocabulary were associated with word vectors after building of the Word2Vec embedding. While the results show that a value of 100 was the optimal value for the minimum word count hyperparameter setting, the usefulness of our word embedding would be highly compromised if 100 were used—as only words whose frequency of occurrence reached 100 or more would appear in the vocabulary.

6. Maximising robustness: Determining variances and testing ensembles of embeddings

When developing and deploying a word embedding, one must seek to maximise its robustness. For example, one does not want the distance between word vectors in an embedding to significantly depend on random initialisation of weights in the training algorithm, or to depend on random permutations of the data when it is used during training of the algorithm. Likewise, the results should be robust against bootstrapping, or subsampling of the data (as long as the overall size or quality of the dataset—and therefore the information available—does not change). If large variances are produced by small changes in the training set, then this is evidence that the embedding does not generalise well (Antoniak & Mimno, 2018).

In order to maximise robustness of our embedding, we conducted tests, as described below, in order to:

- determine the variances produced by data shuffling, random initialisation and bootstrapping; and
- determine the degree to which generation of ensembles of embeddings would reduce variance and improve robustness.

Determining the variances

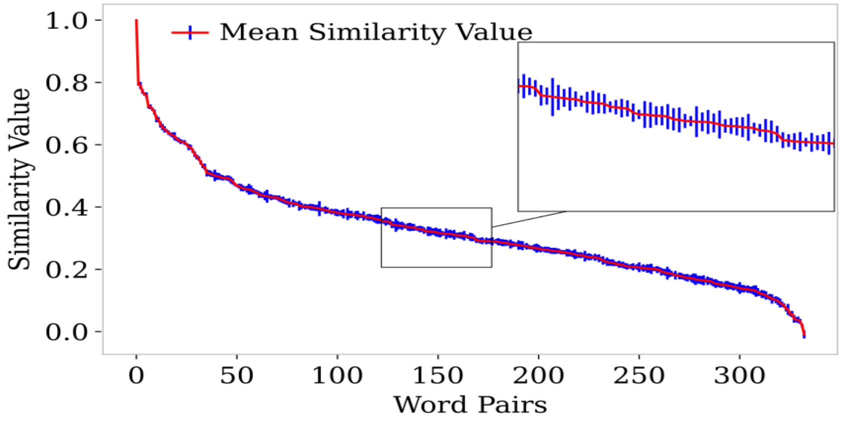
To understand how training stochasticity (data shuffling and random initialisation) and subsampling (bootstrapping) influenced the distances between word vectors in our embedding, we generated three ensemble word embeddings:

- 10p subsampled: This first ensemble consisted of 10 word embeddings trained on 10% of the sentences in the data corpus that were randomly subsampled for each word embedding. The resulting word embeddings therefore had different vocabularies.

- 10p shuffled: This second ensemble consisted of 10 word embeddings trained on the same subset of 10% randomly sampled sentences and the resulting embeddings therefore shared the same vocabulary. (The differences between the embeddings stemmed only from the differences in training procedures.)
- 100p shuffled: This third ensemble consisted of five word embeddings (a smaller number of embeddings, due to their size) that were trained on the entire training dataset. (Again, the difference in the embeddings stemmed only from the differences in training procedures.)

In addition to showing the variance of word similarities between different instances of the embedding, these three ensembles allowed us to study the effect of bootstrapping (when comparing 10p_subsampled vs 10p_shuffled), as well as the effect of the size of the training dataset (when comparing 10p_shuffled vs 100p_shuffled). It should be noted that the vocabulary of the smaller datasets was necessarily smaller as well, and we ignored analogies if one of the words (or the solution) was not part of the word embeddings' vocabulary. We calculated the 360 similarities of word pairs in the WordSim353 dataset (Agirre et al., 2009)[4] for all word embeddings in a set, and plotted the mean and variance of the results, as shown in Figures 4(a), 4(b), and 4(c).

Figure 4(a): Variance of embeddings based on a 10% subsampled set of 10 embeddings



4 http://alfonseca.org/pubs/ws353simrel.tar.gz

Figure 4(b): Variance of embeddings based on a set of 10 embeddings trained on shuffled 10% of the total training dataset

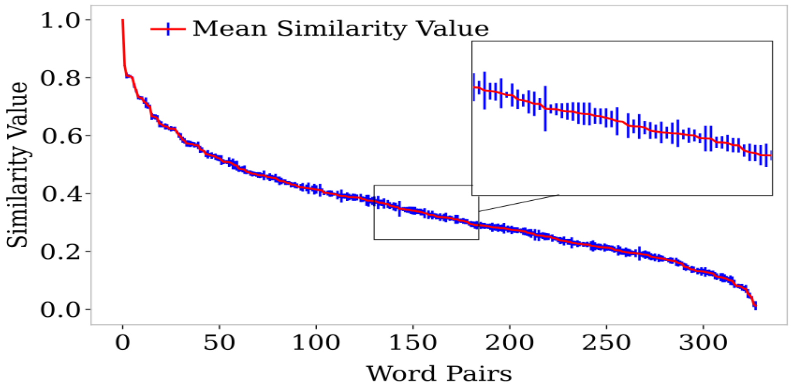
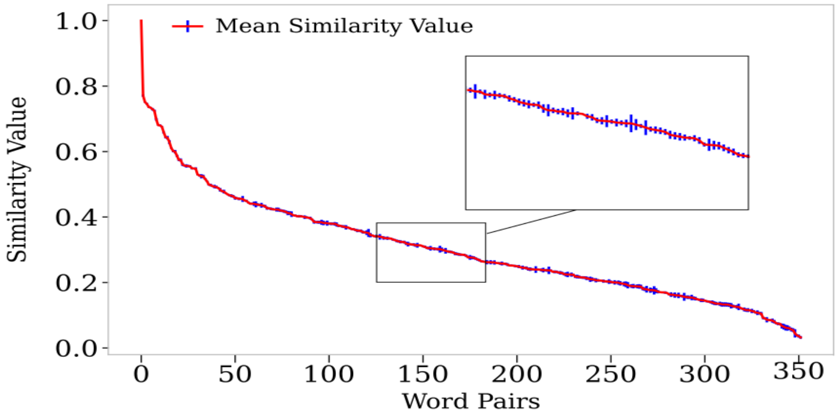


Figure 4(c): Variance of embeddings based on a set of five embeddings trained on shuffled full training dataset



The results confirmed our expectation that a variance in the training itself, as introduced by subsampling, would lead to an even larger variance of the word similarities. Furthermore, the results showed that a larger dataset led to a much lower variance. In the 100p model, for example, the variance was in fact low enough to reliably distinguish distances between words on an order of 0 to $2x10^{-4}$. These results suggested a strategy for how to make our word embedding more robust: build an ensemble model that united the prediction of several models (which is in fact standard practice to decrease variance (Antoniak & Mimno, 2018)).

Determining efficacy of ensemble embeddings

We investigated the efficacy of ensemble embeddings using the aforementioned 14 analogy benchmarks. For these benchmarks, averaging over models was not an option since the result of every analogy was a correct/incorrect answer. We therefore needed more elaborate rules for turning the decision of a single embedding into the ensemble decision. The rule we developed to combine analogy decisions into an ensemble was a rule that (1) computed the list of 10 closest neighbours to the vector in the analogy equation (such as vec(ANC) – vec(EFF) + vec(Zuma)) for every word embedding, and then (2) concatenated (joined together) the lists. We then computed the 10 words that appeared in the largest number of ensembles, and checked whether the desired result was in the final list (positive outcome) or not in the list (negative outcome).

Figure 5: Precision values for ensemble word embeddings evaluated on analogy tasks

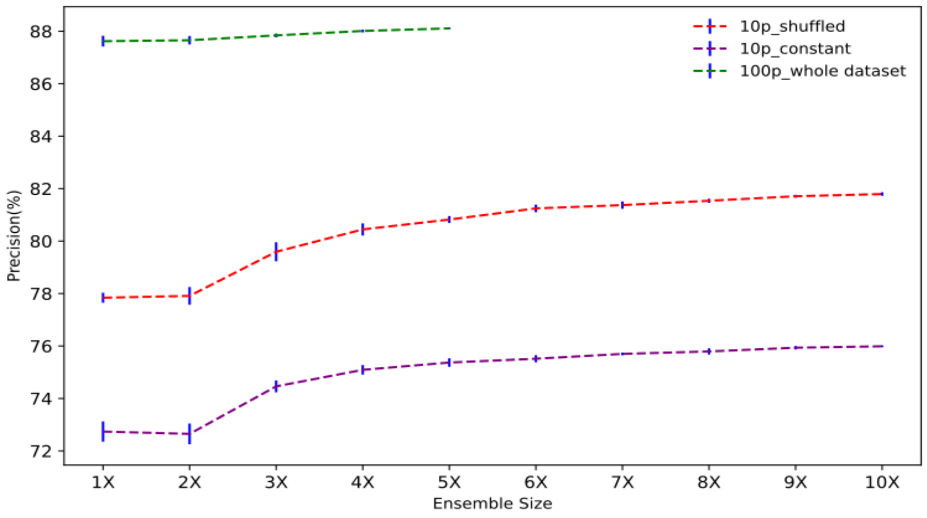


Figure 5 shows the average performance of ensembles of varying sizes on the analogy tasks, using the three different strategies mentioned above to create the training sets for each member of the ensemble. The benchmarks were calculated by randomly sampling, 10 times, ensemble members from the set of trained models. For example, the 2X ensemble word embedding in Figure 5 represents an ensemble consisting of two members, created by randomly sampling two models from a set of 10 word embeddings. This ensured that the results did not depend on the model used to make up the ensemble. This evaluation generated four important observations:

- Generation of ensemble embeddings improved the precision of the model's results.

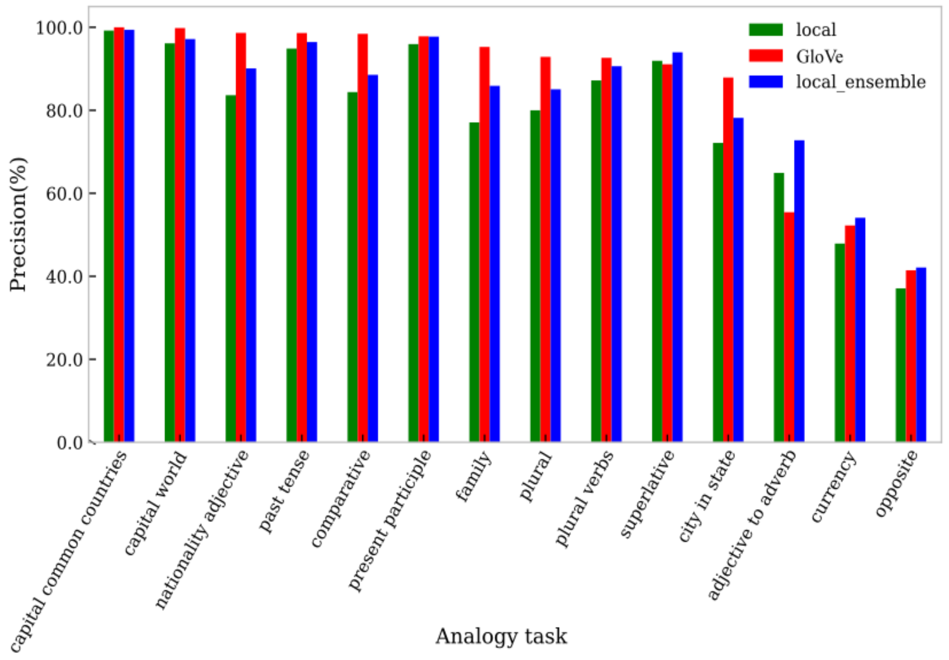
- The ensembles of embeddings performed better as they increased in size.
- This trend (improvement with increased ensemble size) was particularly strong for the 10p models (ensembles trained with smaller datasets), and not so prevalent for the ensembles trained on the full dataset.
- Significant improvements (over single models) occurred even in ensembles composed of three or more embeddings, an important finding given that smaller ensembles are easier to store and use.

As a result, we are able to recommend the use of ensembles of three or more embeddings when datasets are relatively small, which is common in geographic areas, such as on the African continent, that are under-represented in terms of data representation.

7. Performance comparison between South African Word2Vec embedding and a GloVe embedding

Figure 6 shows a comparison between our South African news Word2Vec embedding and the GloVe model in performing the 14 analogy tasks. We trained a single embedding (that is, the "100p_250d_50m" embedding) and a five-member ensemble of the South African news word embedding for comparison, and, consistent with the findings above, the ensemble showed a better performance.

Figure 6: Results from comparison between Word2Vec and GloVe embeddings



Overall, as seen in Figure 6, our Word2Vec word embedding was competitive on most tasks when compared to the GloVe embedding. And, interestingly, our Word2Vec ensemble of embeddings outperformed the GloVe model on four analogy tasks—remarkable given that the latter has the advantage of a significantly larger training dataset size and a global training set domain. This finding was surprising for two reasons. First, the GloVe algorithm is an international benchmark that has been acknowledged as one of the most stable models for production of word embeddings (Wendlandt et al., 2018). Second, the GloVe embedding we compared ours with was trained on a much larger corpus than our local word embedding. The GloVe embedding had approximately 300,000 unique word tokens in each embedding—300,000 for the GloVe embedding versus 124,000 for our embedding. The importance of vocabulary size in ensuring high quality word embeddings is well-documented (Rodman, 2020). The results are therefore encouraging evidence of the robustness and performance of our South African news word embedding.

We observed no differences of statistical significance, between the performance of our Word2Vec word embedding model and the performance of the GloVe model, on the largely universal grammar tasks among the 14 analogy tasks (such as the present participle and superlative tasks), or on semantic analogy tasks (such as capital-common-countries task). (Because we were not able to check the word frequencies for the corpus used to train the GloVe model, we suspect that the equivalence in performance for the two models showed that the words used in the evaluation tasks were sufficiently represented in the corpuses used to train both the GloVe embedding and our local Word2Vec embedding.)

However, we found that for the capital-world analogy task, the South African news Word2Vec embedding did not perform well in analogy examples that involved tokens such as Brussels and Belgium. The results suggested that the news embedding appeared to have known Brussels more in the context of the European Union (EU). This inference followed from noting that the EU was predicted with high likelihood amongst a list of top 10 nearest neighbours as predicted by the news embedding to solve given relational tasks. Also, the South African news embedding failed to solve the relational task which involved Madrid and Spain. We observed that the South African news embedding appeared to know Madrid in the context of Real Madrid, the Spanish football team. This observation points to the need to build situation-specific word embeddings whose learned word vectors fully capture and represent the original views and standpoints expressed by news content creators at the time of writing.

It should also be noted that, during the comparison, we found that there was a subtlety that may have influenced the results to a small extent—a subtlety introduced by our rule that excluded analogies in a task if the embedding (or all of the embeddings in an ensemble) did not contain a word in the analogy. For example, if the local embedding did not contain the word "nursultan" (representing "Nur-Sultan"), the capital-common-countries analogy of guessing Kazakhstan's capital city did not have to be solved, while the global embedding had to solve it, even if the global embedding contained very few examples mentioning the city. Accordingly, we resolved this issue by evaluating the models using only those relational tasks with words that were common to both models.

8. Validation of the embedding against local benchmarks

In addition to testing our South African embedding against the international benchmarks provided by the GloVe algorithm, it was also necessary to validate the embedding against local benchmarks that represented local contexts. For this purpose, we created two local analogy tasks. The first local analogy task involved matching politicians to political parties—for example, EFF is to Julius_Malema as ANC is to Jacob_Zuma. This politician-to-political party relational task consisted of 398 analogy tasks, of which our local Word2Vec South African news embedding model successfully solved 212 (53%)—according to the performance evaluation metrics described above. The second local analogy task involved matching cities to provinces—for example, KZN is to Durban as Western_Cape is to Cape_Town. This city-to-province analogy consisted of 586 relational tasks, of which our local Word2Vec model successfully solved 582 (a model performance of 99%). Due to the fluidity of political affiliations, we were not surprised that the politician-to-political party task proved to be more difficult than the city-to-province task.

However, we observed that, in certain circumstances, our South African news embedding failed to solve certain analogy tasks simply because text preprocessing did not include stemming, a practice that reduces all related tokens to their root word. As a result, the embedding was penalised for predicting DAS (which stands for DA's) instead of DA. When we implemented stemming, that increased the predictive power of the word embedding for the task of matching politicians with their respective political parties (predictive accuracy increased from 53% to 59%). However, stemming proved to be counterproductive for the second task: matching provinces with their cities (predictive power dropped from 99% to 69%).

The negative effects from stemming arose from the fact that a word stemmer is a model that is trained to reduce English words to original root words, and therefore any token that is given to the model will be reduced to the root word that is known to the model. The work of Al-Shammari and Lin (2008) supports our finding regarding the drop in model performance, for certain analogy tasks, following word stemmatisation. Consequently, we recommend development of a full range of localised NLP tools, including word stemmers and lemmatisers, that are optimised to handle local contexts, so as to facilitate training of word embeddings whose learned word vectors truly resemble local contexts.

9. Conclusions

In this article, we have presented a word embedding that was trained, using the Word2Vec algorithm, on South African news article data collated and stored by MMA. The full corpus consisted of news articles that were published between 1 January 2018 and 17 March 2021. We have presented results from testing of the impact of varied hyperparameters, changes in the training set, and ensemble-building, on the performance of the embedding. We have also presented results from comparison of the performance of our local Word2Vec embedding against the performance of the GloVe algorithm—results which showed competitive performance, and even superior performance in some instances, by our local South African embedding. Furthermore, we have provided results from two tests used to check the performance of our embedding against South African benchmarks.

We hope that the embedding and benchmarks we have presented promote further research in South African social sciences, and will help researchers who lack the resources required to train vast machine learning models for NLP. In particular, the word embedding contributed by this study presents researchers with an opportunity to use the word vectors as text encoders. For instance, researchers can use our word embedding in ways similar to how pre-trained word vectors produced by algorithms such as Word2Vec, GloVe, BERT and fastText are used to vectorise texts without the need to train word embeddings from scratch. Consequently, we hope and anticipate that our word embedding will play a significant role as the "embedding layer" in similar South African text analysis studies. We believe that contributions such as these are crucial to unlocking the potential of big data analysis in localised African contexts.

Disclosure statement

The authors report that there are no competing interests to declare.

Data availability statement

The word embeddings that support the findings of this study are openly available in figshare.[5] The code for reproducing the results is available in a github repository.[6]

References

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, M., & Soroa, A. (2009). A study on similarity and relatedness using distributional and WordNet-based approaches. https://doi.org/10.3115/1620754.1620758

Al-Shammari, E. T., & Lin, J. (2008). Towards an error-free Arabic stemming. In *Proceedings of the 2nd ACM Workshop on Improving Non English Web Searching* (pp. 9–16). https://doi.org/10.1145/1460027.1460030

5 https://doi.org/10.6084/m9.figshare.18933728
6 https://github.com/Mafunda/SouthAfricanNewsEmbeddings

Antoniak, M., & Mimno, D., 2018. Evaluating the stability of embedding-based word similarities. *Transactions of the Association for Computational Linguistics, 6*, 107–119. https://doi.org/10.1162/tacl_a_00008

Arseniev-Koehler, A., & Foster, J. G. (2020). Sociolinguistic properties of word embeddings. https://doi.org/10.31235/osf.io/b8kud

Badri, N., Kboubi, F., & Chaibi, A. H. (2022). Combining FastText and Glove word embedding for offensive and hate speech text detection. *Procedia Computer Science*, *207*, 769–778. https://doi.org/10.1016/j.procs.2022.09.132

Bakarov, A. (2018). A survey of word embeddings evaluation methods. arXiv preprint arXiv:1801.09536.

Berardi, G., Esuli, A., & Marcheggiani, D. (2015). Word embeddings go to Italy: A comparison of models and training datasets. In *Proceedings of 6th Italian Information Retrieval Workshop.*

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146. https://doi.org/10.1162/tacl_a_00051

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Durrheim, K., Schuld, M., Mafunda, M., & Mazibuko, S. (2022). Using word embeddings to investigate cultural biases. *British Journal of Social Psychology*, *00*, 1–13. https://doi.org/10.1111/bjso.12560

Goodman, J. (2001). Classes for fast maximum entropy training. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing: Proceedings, 1* (pp. 561–564).

Grand, G., Blank, I. A., Pereira, F., & Fedorenko, E. (2022). Semantic projection recovers rich human knowledge of multiple object features from word embeddings. *Nature Human Behaviour*, *6*, 975–987. https://doi.org/10.1038/s41562-022-01316-8

Gu, Y., Leroy, G., Pettygrove, S., Galindo, M. K., & Kurzius-Spencer, M. (2018). Optimizing corpus creation for training word embedding in low resource domains: A case study in autism spectrum disorder (ASD). In *AMIA Annual Symposium Proceedings, 2018* (pp. 508–517).

Hunt, E., Janamsetty, R., Kinares, C., Koh, C., Sanchez, A., Zhan, F., Ozdemir, M., Waseem, S., Yolcu, O., Dahal, B., & Zhan, J. (2019). Machine learning models for paraphrase identification and its applications on plagiarism detection. In *2019 IEEE International Conference on Big Knowledge (ICBK)* (pp. 97–104). https://doi.org/10.1109/ICBK.2019.00021

Jain, A., Meenachi, D. N., & Venkatraman, D. B. (2020). NukeBERT: A pre-trained language model for low resource nuclear domain. arXiv preprint arXiv:2003.13821.

Kozlowski, A. C., Taddy, M., & Evans, J. A. (2019). The geometry of culture: Analyzing the meanings of class through word embeddings. *American Sociological Review*, *84*(5), 905–949. https://doi.org/10.1177/0003122419877135

Loper, E., & Bird, S. (2002). NTLK: The natural language toolkit. arXiv preprint cs/0205028.

Marivate, V., Sefara, T., Chabalala, V., Makhaya, K., Mokgonyane, T., Mokoena, R., & Modupe, A. (2020). Investigating an approach for low resource language dataset creation, curation and classification: Setswana and Sepedi. arXiv preprint arXiv:2003.04986.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, *26*. https://doi.org/10.48550/arXiv.1310.4546

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). https://doi.org/10.3115/v1/D14-1162

Pereira, F., Gershman, S., Ritter, S., & Botvinick, M. (2016). A comparative evaluation of off-the-shelf distributed semantic representations for modelling behavioural data. *Cognitive Neuropsychology*, *33*(3–4), 175–190. https://doi.org/10.1080/02643294.2016.1176907

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

Rahimi, Z., & Homayounpour, M. M. (2022). The impact of preprocessing on word embedding quality: A comparative study. *Language Resources and Evaluation*, 1–35. https://doi.org/10.1007/s10579-022-09620-5

Řehůřek, R., & Sojka, P. (2011a). Gensim – statistical semantics in Python. NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic. https://www.fi.muni.cz/usr/sojka/posters/rehurek-sojka-scipy2011.pdf

Řehůřek, R., & Sojka, P. (2011b). Gensim – Python framework for vector space modelling. NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic.

Rezaeinia, S. M., Rahmani, R., Ghodsi, A., & Veisi, H. (2019). Sentiment analysis based on improved pre-trained word embeddings. *Expert Systems with Applications*, *117*, 139–147. https://doi.org/10.1016/j.eswa.2018.08.044

Richardson, L. (2007). Beautiful soup documentation. https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Rodman, E. (2020). A timely intervention: Tracking the changing meanings of political concepts with word vectors. *Political Analysis*, *28*(1), 87–111. https://doi.org/10.1017/pan.2019.23

Santos, I., Nedjah, N., & de Macedo Mourelle, L. (2017). Sentiment analysis using convolutional neural network with fastText embeddings. In *Proceedings of the 2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)* (pp. 1–5). https://doi.org/10.1109/LA-CCI.2017.8285683

Svoboda, L., & Beliga, S. (2017). Evaluation of Croatian word embeddings. arXiv preprint arXiv:1711.01804.

Theil, C. K., Štajner, S., & Stuckenschmidt, H. (2020). Explaining financial uncertainty through specialized word embeddings. *ACM Transactions on Data Science*, *1*(1), 1–19. https://doi.org/10.1145/3343039

Turian, J., Ratinov, L., & Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th annual Meeting of the Association for Computational Linguistics* (pp. 384–394).

Wendlandt, L., Kummerfeld, J. K., & Mihalcea, R. (2018). Factors influencing the surprising instability of word embeddings. arXiv preprint arXiv:1804.09692

Xu, R., Yang, Y., Otani, N., & Wu, Y. (2018). Unsupervised cross-lingual transfer of word embedding spaces. arXiv preprint arXiv:1809.03633.

Yin, Z., & Shen, Y. (2018). On the dimensionality of word embedding. *Advances in Neural Information Processing Systems*, *31*. https://doi.org/10.48550/arXiv.1812.04224