sciendo
DOI 10.2478/jazcas-2023-0048

TEXT VECTORIZATION TECHNIQUES BASED ON WORDNET

DÁVID DRŽÍK– KIRSTEN ŠTEFLOVIČ
Department of Informatics, Faculty of Natural Sciences and Informatics,
Constantine the Philosopher University in Nitra, Nitra, Slovakia

DRŽÍK, Dávid – ŠTEFLOVIČ, Kirsten: Text Vectorization Techniques Based on Wordnet. Journal of Linguistics, 2023, Vol. 74, No 1, pp. 310 – 322.

Abstract: The utilization of text vectorization techniques has become essential for numerous classification tasks in present-day natural language processing. Word embedding methods commonly used today, such as Word2Vec, GloVe, etc., are based on the semantic similarity of words. WordNet, as a lexical database of words, provides a rich source of semantic information. In our article, we propose a text vectorization technique using extended text data with the data augmentation method, specifically by replacing words with their synonyms obtained from WordNet. The results obtained from text classification tasks using multiple classifiers demonstrate that expanding the corpus with this method leads to improved vector representations of words.

Keywords: word embedding, Word2Vec, Glove, synsets, text data augmentation, semantic similarity

1 INTRODUCTION

The use of text vectorization techniques is nowadays a necessity for many classification tasks in the field of natural language processing. Modern word embedding models such as Word2Vec, Doc2Vec, Glove, etc. are based on the semantic similarity of words. On the other hand, the semantic relations of individual words are also available in the form of synsets. Synsets capture information about words that are similar in meaning. Unlike embedding models, these relationships are not created automatically by a computational algorithm, but by human annotators.

In the article, we propose a text vectorization method based on synsets. This method will represent an improvement in the training of modern word embedding models. Semantically similar words will not be automatically inferred only from the corpus, but we will also use information about semantically similar words from synsets.

We hypothesize that incorporating synset information can help improve the quality of word vectors by capturing more semantic relationships between words. Synsets will thus provide models with additional context. However, the question how to include these synsets in word embedding models, remains. For individual word embedding models, there are already used libraries that ensure training as well

as the actual use of the created vectors. Our effort is to offer the simplest possible intervention in the process of training embedding models using the available corpus.

With our approach, we want to use information about synsets in the training stage of these models. Synsets will therefore be used to modify and especially supplement the corpus used for model training. In each sentence of the corpus, we will analyse its individual words. For each word of the selected sentence, we find synonyms using synsets. And we will add new sentences to the corpus, in which the analysed word will be replaced by a synonym. For example, for the sentence "Dad is cooking lunch." and the word "dad" we add the following sentences:

- Dad is cooking lunch.
- Daddy is cooking lunch.
- Papa is cooking lunch.
- Father is cooking lunch.
- Male parent is cooking lunch.

We apply a similar procedure for the other words „cooking" and „lunch".

There are several approaches to compare the success of the proposed method. Nazir et al. (2022) evaluate success using datasets wordsim-353 and Lexsim-999, which contain calculated similarities of selected pairs. Calculating the similarities between two words is often presented in educational examples focused on word vectors. However, from a practical point of view, word vectors are used in different practical tasks. Classification is one of them. For this reason, we will verify the suitability of our method by using the trained methods to create vectors in the classification task. Subsequently, we will evaluate the performance measures of the classification itself, i.e., we will not evaluate the word vectors themselves directly, but we will use them to solve the classification task and evaluate the success of the classification.

We will proceed according to the following methodology (Fig. 1):

1. Preprocessing of the corpus for training Word2Vec and GloVe models (corpus 1).
2. Creation of the second corpus using synsets (corpus 2).
3. Training Word2Vec (Skip-gram, CBOW) and GloVe with two separate corpora – in this way, 6 word embedding models will be created:
   - **Word2Vec Skip-gram (C1)** – Word2Vec Skip-gram trained using Corpus 1
   - **Word2Vec CBOW (C1)** – Word2Vec CBOW trained using Corpus 1
   - **GloVe (C1)** – GloVe trained using Corpus 1
   - **Word2Vec Skip-gram (C2)** – Word2Vec Skip-gram trained using Corpus 2
   - **Word2Vec CBOW (C2)** – Word2Vec CBOW trained using Corpus 2
   - **GloVe (C2)** – GloVe trained using Corpus 2.
4. Preprocessing of the dataset for the classification task.
5. Creation of input sequences of vectors to classifiers from the dataset.
6. Creating and verifying the quality of classification models using k-fold validation.
7. Identification and comparison of the performance of the created models.

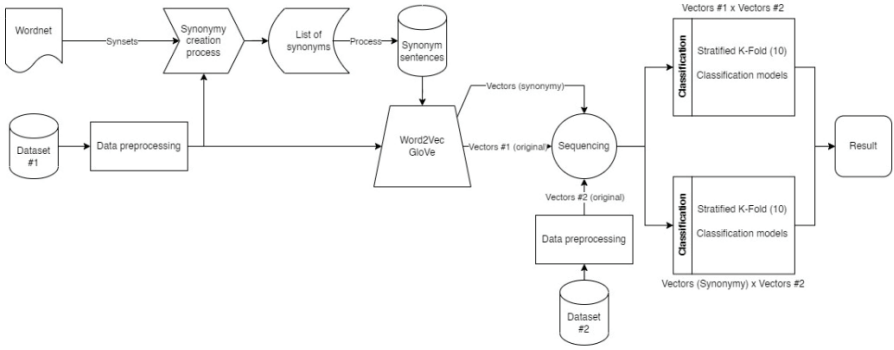


Fig. 1. Workflow of our proposed method

The article has the following structure. The second part summarizes the current state of research. The text corpus, but also the classification dataset and their preprocessing, the creation of the second corpus using synsets and the used word embedding models are described in the third part. The most important results are summarized in the fourth section. Discussion and conclusions form the content of the last part of the article.

2 RELATED WORK

Data augmentation is a technique used to increase the size and diversity of a training dataset by creating new examples from existing ones through various operations such as random rotations, flips, zooms, and translations (Pellicer – Ferreira – Costa 2023). The goal of data augmentation is to enhance the performance and robustness of machine learning models by exposing them to a wider range of variations and situations. Data augmentation has been successfully applied in various domains such as computer vision, natural language processing, and speech recognition. Moreover, data augmentation can also help mitigate overfitting by introducing more variation into the training data and preventing the model from memorizing the training examples.

There are few popular techniques for text data augmentation:

- **Back Translation**: Translating sentences to a foreign language and then back to the original language to create new variations of the original text.
- **Synonym replacement**: Replacing words in a sentence with their synonyms to increase the diversity of training data.
- **Paraphrasing**: Restating the same idea using different words and structures to generate new variations of a sentence or passage.
- **Random Insertion**: Adding randomly chosen words to a sentence to create new variations and increase the length of the sentence.

- **Random Swap**: Swapping two words in a sentence to create new variations while preserving the overall structure of the sentence.
- **Random Deletion**: Removing randomly chosen words from a sentence to create new variations and increase the focus on the most important words.

Data augmentation techniques, such as easy data augmentation (EDA), have been proposed to boost the performance of text classification tasks Wei et al. (2019). EDA consists of four of these simple techniques including synonym replacement, random insertion, random swap, and random deletion. Through experiments on five text classification tasks, EDA has shown to improve performance for both convolutional and recurrent neural networks. EDA also demonstrated particularly strong results for smaller datasets. On the other hand, Haralabopoulos et al. (2021) propose reverse classification of each augmented example on antonyms and negation.

Traditional synonym recommendations often include ill-suited suggestions for the writer's specific contexts. To address this issue, Glenski et al. (2021) proposed a simple approach for contextual synonym recommendation. This method combines existing human-curated thesauri, such as WordNet (Miller 1995), with pre-trained language models. Wang et al. (2015) is using the same thesauri to enhance computational behavioral analysis using social media text. To further improve the effectiveness of data augmentation through synonym replacement, Kobayashi (2018) proposes a novel method called contextual augmentation using BERT transformer (2019).

Marivate et al. (2020) explores different text augmentation methods on three datasets. The goal is to provide insights on making choices for classification use cases. Word2Vec-based augmentation works well without a formal synonym model. Mixup further improves performance and reduces overfitting.

3 MATERIALS AND METHODS

3.1 Datasets and their preprocessing

In our article, we used two freely available data files. The first text dataset or corpus (Risdal 2016) contains 143,000 English articles from 15 American journals. Before using this text, it is necessary to preprocess it as precisely as possible. We started by tokenizing the text into sentences using the NLTK library (Bird – Klein – Loper 2009). This dataset contains a total of 704,357 sentences. Subsequently, we converted the text in each sentence to lowercase letters, cleaned the text from hyperlinks, from white characters and from special characters that are not numbers or letters. We split the sentences into words. We then removed words that contained numbers as well as numbers themselves, ensuring that only words with linguistic meaning appeared in the dictionary. We have also removed stop words that appear too often in the text but have no meaning in themselves. And the last step of text

preprocessing was lemmatization using WordNetLemmatizer, which reduced the number of words in the dictionary, since they can appear in the dictionary in different forms, but carry the same meaning. For further work, we did not use all this text, but only a part of it, namely the first 10,000 sentences.

The second dataset, WELFake (Verma – Agrawal – Amorim – Prodan 2021), contains a total of 72,134 articles correctly labelled as fake news (35,028) and real news (37,106). We also preprocessed this file in the same way as the first one. We will use this classification dataset to verify the quality of the vector model.

3.2 Creating a list of synonyms from wordnet synsets

In this article, we want to experiment with word vectors and see if replacing words with their synonyms improves their quality. Therefore, we need to create a list of synonyms for the words in our examined text.

To create a list of synonyms, we used the sentences from the first data set, because in this text we will replace words with their synonyms, so it does not make sense to look for synonyms for words that we will not need later. The process of creating synonyms was as follows. We gradually went through the individual words of all preprocessed sentences. Then, for each word, we searched for synsets (list of synonyms) using the function synsets through the WordNet library (Miller 1995) from the NLTK library (Bird – Klein – Loper 2009). If no synonyms were found for the specified word in the WordNet dictionary, we tried to lemmatize the word again, this time using the WordNet morphy function, which tries to find its lemmatized form, which is listed in the WordNet dictionary. If the word is found, a list of synonyms for the modified word is searched.

After finding a synset for our word, we went through each synonym in turn and filtered the list based on semantic similarity (by using path similarity, result value is between 0 and 1, where a value of 1 indicates maximum similarity) to our word. If this similarity was higher than 0.5, we kept this synonym, otherwise we removed this synonym from the synset. In the end, we lemmatized all valid synonyms and stored a list with a maximum length of 5 synonyms per word. We saved the list as a binary file using the Pickle library (Rossum 1999).

3.3 Replacing words in the original sentences with their synonyms

After creating a list of synonyms (a word and a list of synonyms for this word), of which there were 9,704, we decided to gradually replace each word for which we register a list of synonyms in all sentences. We used a technique known as Data Augmentation – replacement by synonyms (Wei – Zou 2019). In our experiment, we go through all words in all sentences and check for each word whether it contains a list of synonyms. If so, then we will duplicate the whole sentence and replace the one specific word with each synonym for this word. As a result, we will have several times more sentences than the original ones. Just to give an idea, from the original

10,000 sentences, we created 175,354 sentences with this method, which is more than 17 times.

3.4 Creation of word vectors

Word vectors is one of the ways to represent words using numbers. They are n-dimensional vectors of real numbers, which are placed in the vector space in such a way that they capture the meaning or relationships between individual words. In our article, we used word models Word2Vec and GloVe to create word vectors. Using the Gensim (Řehurek – Sojka 2010) and GloVe (Pennington – Socher – Manning 2014) libraries, we implemented these 3 vector models in Python by creating the corresponding models.

Word2Vec

The Word2Vec model works on the principle that the very meaning of a word is represented by its context, i.e., the words that come before and after it. As a result, this model creates one multidimensional vector for each word. Syntactic or semantic relations between words are preserved, and the vector distances of individual words correspond to the human idea of the relation between words (Wei – Zou 2019; Mikolov – Chen – Corrado – Dean 2013).

The creators of the Word2Vec model themselves state two basic architectures: CBOW (Continuous Bag-of-Words) and Skip-gram. CBOW focuses on predicting a target word based on its context words within a context window, while Skip-gram focuses on predicting context words based on the target word within a context window. Both architectures are actually just simple neural networks with input, hidden and output layers. The result of the network training is not the output of the output layer, but the weights of the hidden layer, which represent the word vectors. A Softmax activation function is used for the output layer, but Word2Vec also uses algorithmic improvements such as hierarchical Softmax and negative sampling to reduce the computational complexity (Řehurek – Sojka 2010; Mikolov – Chen – Corrado – Dean 2013).

In Python, we used the Word2Vec model implementation using the Gensim library (Řehurek – Sojka 2010). The input to the model was preprocessed sentences, we set the size of the dimension to 150 and the size of the context window to 7. The reason for choosing these values was our previous research, where we had been investigating the appropriate setting of these parameters. We used both the skip-gram and CBOW architectures, so we created two models for unedited sentences and two models for edited sentences (words in sentences replaced synonyms). By training the model defined in this way, we obtain word vectors for all available words. We saved the individual word vectors and their corresponding words using the Gensim library (Řehurek – Sojka 2010) and the KeyedVectors class in the form of a complex dictionary in a binary file.

GloVe (Global Vectors)

The GloVe (Global Vectors) model is another important model for creating word vectors, which is similar to the Word2Vec model. Its advantage is that it can consider the statistical occurrences of words in the context in which they appear, thus capturing finer semantic relationships between words. The GloVe model is based on matrix factorization of the word context matrix. That is, first a large matrix is created that contains information about the common occurrence of words in context. For each word, its frequency of occurrence in a context in the corpus is calculated. Then this matrix is factorized to obtain a lower-dimensional matrix (Pennington – Socher – Manning 2014). We also implemented the GloVe model in Python using the GloVe library (Pennington – Socher – Manning 2014), with all parameter settings identical to those of the Word2Vec model, except for a learning_ rate of 0.05 and a number of iterations of 30.

3.5 Verification of the quality of the vector model on the classification task

It is generally stated that intrinsic and extrinsic evaluations are used to evaluate the performance of language models. The basic difference between them is that with Intrinsic evaluations, the performance of the system is evaluated directly on the specific task for which it was designed. For example, if a language model is designed to generate coherent sentences, such an evaluation would include measuring the coherence of the sentences generated. On the other hand, extrinsic evaluation involves evaluating the system's performance on a subsequent task for which the system was not specifically designed. For example, if a language model is designed to generate coherent sentences, extrinsic evaluation would involve measuring the model's ability to improve the performance of a downstream task such as machine translation or sentiment analysis (Zhai – Tan – Choi 2016; Shi – Zheng – Guo – Zhu – Qu 2018).

Since in our article we want to find out whether word vectors created on modified sentences (words replaced by their synonyms) or on original sentences are of better quality, we will use extrinsic evaluation. In particular, we will solve the classification of fake news using our created word vectors and we will compare how well we can classify fake news with vectors created over the original and modified sentences.

3.6 Creating sequences of word vectors

Even before creating the sequences of word vectors, we need to modify the classification dataset of fake news. We have already done the basic preprocessing, but this dataset also contains words for which we have not created word vectors. Such words will not help us in the classification of fake news and thus also in verifying the correctness of our word vectors, so we will remove them. After the removal, we noticed that some records remained completely empty of content, or the

number of words in the records decreased. To improve the quality of our results, we removed 5% of extreme entries from the top and bottom based on the number of words in each entry.

Another problem we had to solve was that the number of words in each record was not the same. Some entries had few words and some had too many words. Since we will want to create sequences of word vectors for each word in the record, and one vector has dimension 150, with very many words we would end up with a sequence of huge dimension. Therefore, we decided to take only the first 10 words from each record to classify fake news.

Creating the sequences was already easy because we went through the individual records and replaced the words in them with their word vectors. The result is a list of sequences, where one sequence consists of ten vectors of dimension 150.

3.7 Classification of fake news

Since our classification dataset contains too many records, we decided to continue working only with the first 15,000 records. And since we wanted to have a balanced dataset, in terms of the balance of false and true message classes, we used the undersampling method to balance the dataset by randomly reducing the number of entries from the larger class to the number of entries from the smaller class. As a result, we will work with 7,265 records for each of the classes. Thus, the input to the classification model will be a list of sequences of word vectors and the corresponding indication of whether it is a false message or not.

To classify fake news, we use several different classifiers from the Scikit-learn library (Shi – Zheng – Guo – Zhu – Qu 2018), which we will introduce in the next section. For classification, we used the Stratified K-fold cross-validation method, which works on the principle that the entire set of data is divided into the same number of K sets, one of which is used for testing and the rest are used for training the classification model, while the entire process on K-repeats times. Moreover, all sets will contain the same representation of individual classes.

In our evaluation we set k = 10, so the dataset was split into 10 subsamples. To assess the performance of the model in each part of the cross-validation, we used the basic performance measure metrics of the model: accuracy, precision, recall and F1 score. After obtaining the results from all parts of the cross-validation, we averaged the results of the individual metrics and thus obtained the average results of the correctness of the classification of fake news.

4 RESULTS AND DISCUSSION

After creating the Word2Vec and GloVe models, we used these models to create word vectors for fake news classification. Word2Vec and Glove were trained on a text corpus of English articles with a dimension of 150 and a window size of

7. According to the methodology presented in the Introduction section, we created classification models for identifying fake news based on word vectors. The classification models were created from the WELFake dataset presented in Section 3.1. The following classification algorithms were used to create classification models:

- Logistic Regression,
- SGD Classifier,
- K-Neighbours,
- BernoulliNB,
- SVC.

All algorithms are well known in the community. For the comparability of the results, we used them with basic settings without hyperparameter optimization. We used Stratified k-fold validation for evaluating our classification models. The evaluation metrics in our experiments is the classification accuracy. Accuracy is the ratio of correct predictions to the total number of samples and is computed as (1):

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} ¿¿ \quad (1)$$

where TP represents the number of True Positive results, FP represents the number of False Positive results, TN represents the number of True Negative results, and FN represents the number of False Negative results. To evaluate, analyse and describe the results of our model we also calculated precision (2), recall (3) and F1 score (4) as follows:

$$precision = \frac{TP}{TP + FP} ¿¿ \quad (2)$$

$$recall = \frac{TP}{TP + FN} ¿¿ \quad (3)$$

$$F1\ score = 2 * \frac{precision * recall}{precision + recall} ¿¿ \quad (4)$$

Precision is the ratio of correctly predicted positive observations of the total predicted positive observations. Recall shows the ratio of correctly predicted positive observations to all observations in the actual class. F1 score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

In our results, we present the average values of the stratified k-fold measurements. In the case of fake news classification models created from vectors generated by Word2Vec Skip-gram models (Tab. 1), better results can be seen in the accuracy metric for all used classification methods in the case of the C2 corpus. So it

means that we can achieve better results with our proposed method of preparing Word2Vec models using synsets (corpus C2). In the table (Tab. 1), for the sake of clarity, we have marked only pairs of performance measures in which we did not achieve an increase in the case of the C2 corpus. From the results, it can be seen that for most performance measures, the results improved in the case of applying our method (corpus C2). In the case of recall, the results worsened only for the K Neighbors classification method, in the case of precision, again for the BernoulliNB and SVC classification methods. Here it is interesting that the Precision results decreased, but the Recall values increased quite significantly. Also, for this reason, an increase in the results for the F1 score was also observed.

Classification method	Word2Vec Skip-gram (C1)				Word2Vec Skip-gram (C2)			
	Acc.	Prec.	Recall	F1	Acc.	Prec.	Recall	F1
Logistic Regression	80.95	80.24	82.19	81.19	81.77	80.81	83.34	82.05
SGD Classifier	79.99	77.87	84.76	80.86	81.54	79.40	85.37	82.18
K Neighbours	72.94	74.50	**69.94**	**72.11**	75.15	83.84	**62.37**	**71.43**
BernoulliNB	79.37	**75.69**	86.65	80.78	80.91	**72.60**	99.34	83.88
SVC	81.99	**80.41**	84.66	82.47	84.91	**79.12**	94.88	86.28

Tab. 1. Average k-fold values for selected performance measures of models created using Word2Vec Skip-gram C1 and C2

In the case of models created over vectors from Word2Vec CBOW (Tab. 2), the average values of the Accuracy metric and also the F1 Score improved in all classification models. The results for the Precision and Recall metrics are mixed.

Classification method	Word2Vec CBOW (C1)				Word2Vec CBOW (C2)			
	Acc.	Prec.	Recall	F1	Acc.	Prec.	Recall	F1
Logistic Regression	79.80	78.10	**82.90**	80.41	81.42	80.73	**82.57**	81.63
SGD Classifier	76.80	74.92	**83.81**	77.95	77.87	77.21	**79.19**	78.15
K Neighbours	67.77	**67.31**	66.36	67.31	70.74	**65.09**	89.90	75.44
BernoulliNB	74.14	**73.91**	74.72	74.29	80.67	**72.54**	98.75	83.64
SVC	75.81	74.10	79.42	76.66	84.20	80.35	90.59	85.15

Tab. 2. Average k-fold values for selected performance measures of models created using Word2Vec CBOW C1 and C2

Perhaps the best results in favor of our proposed method of corpus editing according to synsets were recorded for GloVe (Tab. 3). In addition to the average

Recall value for the SGDClassifier classifier, an improvement in results was observed in all classification methods in favor of GloVe for the C2 corpus. At the same time, the largest increase in the value of Accuracy was recorded here in all monitored models and classification methods (value increase by 12.53 for the BernoulliNB classifier, from 67.42 to 79.95). Also, the largest increase in the value of F1 Score was observed here in all monitored models and classification methods (increase in value by 14.96 for the BernoulliNB classifier, from 68.06 to 83.02).

Classification method	GloVe (C1)				GloVe (C2)			
	Acc.	**Prec.**	**Recall**	**F1**	**Acc.**	**Prec.**	**Recall**	**F1**
Logistic Regression	76.62	74.81	80.29	77.45	80.56	79.91	81.68	80.77
SGD Classifier	76.60	72.80	**85.81**	78.53	80.05	78.95	**82.00**	80.42
K Neighbors	66.11	66.99	63.57	65.22	70.36	73.46	64.46	68.29
BernoulliNB	67.42	66.80	69.40	68.06	79.95	72.03	97.98	83.02
SVC	73.75	71.78	78.34	74.90	83.90	79.64	91.11	84.98

Tab. 3. Average k-fold values for selected performance measures of models created using GloVe C1 and C2

By comparing the average values of the monitored performance measures, we pointed out the fact that by using our method it is possible to improve the Word2Vec and GloVe models. We evaluated their improvement for subsequent classification tasks. Our method significantly improved the GloVe model.

In addition to presenting our improvement of Word2Vec and GloVe models using synsets, in the article we also wanted to present a methodology for verifying the suitability of methods for improving these models. From the analysis of the available literature, we discovered the use of the methodology presented in the article (Nazir et al. 2022), by calculating the similarity of pairs of words. This approach is justified in the case of using the Word2Vec and GloVe models for the identification of semantically similar words. However, we think that in practical applications these models will be used mainly for the preparation of word vectors for classification tasks. For this reason, we wanted to evaluate our improvements to the Word2Vec and GloVe models based on improved performance measures in classification tasks.

5 CONCLUSION

New methods of representing word vectors (Word2Vec, GloVe) represent a significant leap forward in the ability to analyze relationships between words, sentences, and documents. Even though natural language processing is becoming something of a black box that usually works well, we have pointed out other possibilities for improving models for word embedding with our proposed method.

ACKNOWLEDGEMENTS

This work was supported by the Scientific Grant Agency of the Ministry of Education of the Slovak Republic and Slovak Academy of Sciences under Contract VEGA-1/0490/22, also supported by the Slovak Research and Development Agency under the contract no. APVV-18-0473.

References

Bird, S., Klein, E., and Loper, E. (2009). Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit, 1st ed. Beijing; Cambridge Mass. O'Reilly Media.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv, May 24, 2019. Accessed: Mar. 26, 2023. [Online]. Accessible at: http://arxiv.org/abs/1810.04805.

Glenski, M., Sealy, W. I., Miller, K., and Arendt, D. (2021). Improving Synonym Recommendation Using Sentence Context. In Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing, pages 74–78. Accessible at: doi 10.18653/v1/2021.sustainlp-1.9.

Haralabopoulos, G., Torres, M. T., Anagnostopoulos, I., and McAuley, D. (2021). Text data augmentations: Permutation, antonyms and negation. Expert Systems with Applications, Vol. 177. Accessible at: doi 10.1016/j.eswa.2021.114769.

Kobayashi, S. (2018). Contextual Augmentation: Data Augmentation by Words with Paradigmatic Relations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 452–457. New Orleans, Louisiana. Accessible at: doi 10.18653/v1/N18-2072.

Marivate, V., and Sefara, T. (20202). Improving Short Text Classification Through Global Augmentation Methods. In Machine Learning and Knowledge Extraction, pages 385–399. Accessible at: doi 10.1007/978-3-030-57321-8_21.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. ArXiv. Accessible at: doi 10.48550/arXiv.1301.3781.

Miller, G. A. (1995). WordNet: A lexical database for English, Commun. ACM, 38(11), pages 39–41. Accessible at: doi 10.1145/219717.219748.

Nazir, S, Asif, M., Sahi, S. A., Ahmad, S., Ghadi, Y. Y., and Aziz, M. H. (2022). Toward the Development of Large-Scale Word Embedding for Low-Resourced Language. IEEE Access, Vol. 10, pages 54091–54097. Accessible at: doi 10.1109/ACCESS.2022.3173259.

Pellicer, L. F. A. O., Ferreira, T. M., and Costa, A. H. R. (2023). Data augmentation techniques in natural language processing. Applied Soft Computing, Vol. 132. Accessible at: doi 10.1016/j.asoc.2022.109803.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543. Doha, Qatar. Accessible at: doi 10.3115/v1/D14-1162.

Risdal, M. (2016). Getting Real about Fake News, Kaggle. Accessible at: doi 10.34740/KAGGLE/DSV/911.

Rossum, G. V. (1999). Python Library Reference. To Excel Inc.

Řehurek, R., and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora, pages 45–50. Accessible at: doi 10.13140/2.1.2393.1847.

Shi, Y., Zheng, Y., Guo, K., Zhu, L., and Qu, Y. (2018). Intrinsic or Extrinsic Evaluation: An Overview of Word Embedding Evaluation. In 2018 IEEE International Conference on Data Mining Workshops (ICDMW), pages 1255–1262. Singapore. Accessible at: doi 10.1109/ICDMW.2018.00179.

Verma, P. K., Agrawal, P., Amorim, I., and Prodan, R. (2021). WELFake: Word Embedding Over Linguistic Features for Fake News Detection. IEEE Trans. Comput. Soc. Syst., 8(4), pages 881–893. Accessible at: doi 10.1109/TCSS.2021.3068519.

Wang, W. Y., and Yang, D. (2015). That's So Annoying!!!: A Lexical and Frame-Semantic Embedding Based Data Augmentation Approach to Automatic Categorization of Annoying Behaviors using #petpeeve Tweets. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2557–2563. Lisbon, Portugal. Accessible at: doi 10.18653/v1/D15-1306.

Wei, J., and Zou, K. (2019). EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. arXiv, Aug. 25, 2019. Accessible at: doi 10.48550/arXiv.1901.11196.

Zhai, M., Tan, J., and Choi, J. (2016). Intrinsic and Extrinsic Evaluations of Word Embeddings. AAAI, 30(1). Accessible at: doi 10.1609/aaai.v30i1.9959.