Evaluation of BERT outperforming Context-free models for Text Classification Tasks

Deepak Rastogi
rastog01@ads.uni-passau.de
Universität Passau
Passau, Germany

ABSTRACT

A language model needs to care about the context of a sentence before making any predictions. As a state-of-the-art language model, BERT (Bidirectional Encoder Representations from Transformers) has shown incredible results in almost all known Natural Language Processing tasks due to its ability to learn representations bidirectionally. Thus, we call it a contextual model and compare it with a context-free model like Word2Vec for the News analysis task. In this paper, we train the Word2Vec model using 2225 news articles and use its embedding in deep learning model for classification. On the other hand, we fine-tune a BERT model with 110 million parameters for the same task. The BERT model outperforms the Word2Vec model and reaches the weighted F1 score of 97.31%.

KEYWORDS

Deep Learning, neural networks, News analysis, BERT, Word embedding, context-free models, Word2Vec, GRU, Multi-class classification

1 INTRODUCTION

The English language has numerous words that are spelled out the same. However, their meanings sometimes, are unrelated. For example, *"he went on a date with someone"*, *"the date was July 24th"* and *"they ate date brownie after the dinner"* have a common word date which has distinct meanings in each sentence and it is impossible to differentiate without their context.

Word embedding in Natural Language Processing, in general, can be differentiated based on various parameters like frequency-based [10], prediction-based[10], context-based, etc. but here we focus towards context-based and context-free language models. Although the models we are going to discuss in this paper might satisfy two or more above mentioned parameters.

In Natural Language Processing, models that take the context of the word into their account are known as contextual models and those treat such words as same in every case are context-free models. Context-free models generate the same word embedding for such words which might confuse our classification model. Some of the context-free models are Word2Vec [11], Glove [14], and all frequency-based models. On the other hand, examples of contextual models are ELMo [15], OpenAI GPT [16], BERT [4], etc. These could be further differentiated based on being Unidirectional, which uses left-to-right architecture and Bidirectional which uses both left-to-right and right-to-left architecture.

A very important aspect of this paper is to know whether BERT is really a context dependent model. [23] answers this question briefly. The authors test contextual behavior of BERT word embeddings. They placed contextualized embeddings in a shared vector space and investigate their properties on an example basis using k-nearest neighbour. They observed that these embeddings really captures sense when placed in different regions in the vector space.

In this paper, we will compare a context-free model i.e. Word2Vec with a contextual model like BERT. We will train Word2Vec with our data and use the learnt word representation in a classification model. Same thing will be repeated with a pre-trained Word2Vec model. Late, we will load a pre-trained BERT model and fine-tune it for our text classification problem. The results will be reported in terms of precision, recall and F1 score. This comparison of two models help us understand that how important it is for a model to be context aware. Also, We will see how a pre-trained model is beneficial to use rather than training them from scratch.

2 RELATED WORK

In the early works, [5] gave a hypothesis that the meaning of a word depends upon the company it keeps. This hypothesis led to a very simple representation of words. [10] explain that a word can be represented as a vector of N unique values. Then the count of words taken into account based on a given distance. The problem with these embedding method was that it produces a very high dimensional vectors. In the later advancements, [11] introduced the idea of obtaining word embedding by training a large number of corpus. The idea came from the concept of distributed representation of words [12].

Simlar to our work, [19] compares Bag-of-words with Word2Vec model on text classification tasks. The authors numerical representations of each model in a Random Forest with softmax classifier to make predictions. They observed that Word2Vec model significantly outperformed the vanilla bag-of-words model. They also reported that by slightly varying Word2Vec paramters resulted in surprising results.

[9] discuss various types of language representation models on clinical text corpora including pre-trained language models including BERT and Word2Vec. The authors perform two types of evaluation methods i.e. intrinsic evaluation where they test the quality of word representations of different language models and extrinsic evaluation where the quality of results of the application task were discussed. However, they emphasize more upon the clinical data. Furthermore, they talk about standard processes involved to train (or fine-tune) the language models rather reporting which model performs better on their data.

3 DISCUSSION

3.1 Word2Vec

Word2Vec [11] is computationally efficient predictive model [10] used to produce word vectors from raw text. The model first constructs vocabulary out of the training data and use them to find the vector representations for the words in the documents [7]. The learned word vectors can later be used for various Natural language processing tasks. The word2vec model [11] has two ways to find the word vectors. First is, CBOW (Continuous bag of words) and Skip Gram. The architecture of both the models more or less similar but they work differently. The architecture includes one hidden layer known as the projection layer which is shared for all words in the context and all words get projected into the same position. This is known as CBOW. Similarly, in the Skip-Gram model, the current word is treated as the input to a log-linear classifier with the projection layer and predicts words within a certain range around the input word. To investigate the model, we can find the closest words for a specific word [7]. The word2Vec model produces a vocabulary of 10997 words and phrases. Note that, instead of finding vectors for individual words of our articles, we extract phrases for each article in the dataset in prior. We discussed this in the introduction part that sometimes a group of words captures more information than the individual words alone. In our case, we use two-word sequence (also called bigram model) [6].

In [10], shows that embedding those are obtained by this method for similar words has similar vectors. They are correlated using cosine similarity between those words. This is however important to note that the opposite words also lie in the same context. This is where Word2Vec lacks in maintaining the context. In [11] shows that the score of opposite words (accept,reject) is 0.73 [10] which depicts both words has very less semantic gap, but in reality this is not the case.

3.2 BERT

BERT (Bidirectional Encoder Representations from Transformer) is based on a multi-layer bidirectional transformer architecture which is trained on plain text and predicts masked word and the next sentence. [22] The masked language model masks 15% of the words randomly in the input text and it predicts the vocabulary id of the masked words based on its context.

In order to fine-tune BERT, we provide input texts and output labels to the pre-trained model. It takes a input of sequences not more than 512 token (for BERT base). The sequence has always a first token as [CLS] which has a special classification embedding and other token [SEP] which is used for separating the input segments. For text classification, BERT takes final hidden state h of the first token i.e. [CLS] for representing the whole sequence. A softmax classifier at the top predicts the probability of label c. [21]

$$p(c|h) = softmax(Wh) \tag{1}$$

where W is the task specific parameters.

4 DATASET

The dataset we use here is origination from BBC News, provided for use as a benchmark for machine learning research. The dataset consists of 2225 documents from the BBC news website corresponding to stories in five topical areas from 2004 – 2005. The data is distributed among 5 class labels i.e. business, entertainment, politics, sport, and tech [8]. The distribution of the news articles per class label can be seen in fig. 1.

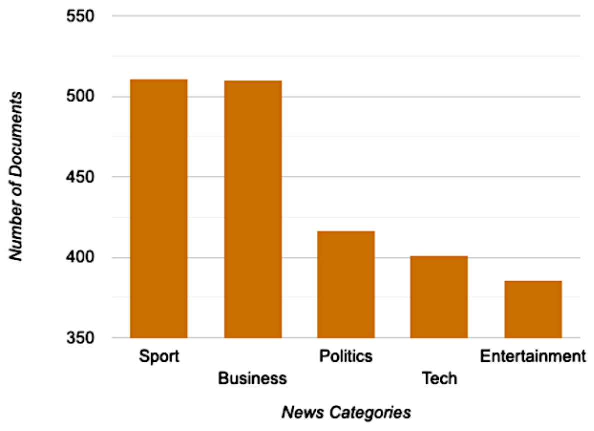

Figure 1: Number of news articles per class

5 METHODOLOGY

In this paper, at an abstract level, we can divide our experiment into two tasks - task I and task II. Task I can also be into two subtasks. In task I, first we prepare a Word2Vec model with a variety of parameter configurations and choose the best ones. This task would result in a list of vocabulary and their vectors. Then we use those vectors, prepare them in a two-dimensional matrix, and provide it to the embedding layer of our deep learning model for classification along with the input articles. Task II includes loading a pre-trained BERT model [4], prepare the data into a specific format (discussed in upcoming sections), and train the model for the domain-specific tasks.

5.1 TASK I - Classification using Word2Vec Embedding

Initially, we split our data into 1780 training sets and 445 into test sets. Then, both sets are separately pre-processed according to the model's requirements.

Preprocessing

Since the textual data contains a lot of information which might not be much useful to our language model. Hence, we need to clean our data as much as possible. In order to clean our data, we remove the newline characters because a news article is shaped in multiple paragraphs. Second we remove the punctuation marks [1] from both training and test documents. Numerical values also would not be of so much use, hence we remove them as well. There is a full list of stop words provided along with the dataset [8] which is useful for cleaning the stop words from the articles. Later we convert the documents into lists of lowercase token. Now our documents are cleaned and ready for training our Word2Vec model.

Here we carry out another experiment Where we use pre-trained word embedding trained over Google news dataset of roughly 100 billion words. It has a vocabulary of 3 million words and phrases. The length of the vector of each word is 300 features [7].

Parameter tuning and Training

To obtain the best parameters for our model, we choose different possible settings of parameters and observe the impact of each of our predictions. For observing the variations in the result, we have set static parameters for our classification model. Later, we will check different parameter settings for our deep learning model also. The results of the test of the different parameters will be discussed in the further sections.

Minimum Count. The minimum count is the absolute frequency of a word in document [17]. It ignores all the words with a total absolute frequency lower than the provided value [17]. We experimented with three values of minimum count that is 1, 2 and 3.

Window. The window size is the distance of current the current word and the predicted word within a sentence [17]. In the experiment, we choose 3, 5 and 7 for the value of the window.

Epochs. We experimented with three values of epochs i.e. 8, 16 and 32. Although, all the results showed the highest accuracy value when the epoch value was 32.

Embedding Matrix

After training the Word2Vec model on the news dataset, we get a vocabulary of 10997 words and phrases. Once we have found the word vectors for our documents, we have to prepare them for use in our deep learning model for classification. So we create a two dimensional matrix having 300 features of each word (or phrase) in a single row. This embedding matrix will later be used by our embedding layer of deep learning model for classification. In the case of the pre-trained Word2Vec model, we discard the vocabulary that does not belong to our dataset and keep the rest of the words and phrases. This reduces the process reduces the actual vocab count, which is 3 million words to around 28223 words and phrases.

Sequence Generation

Before we start to train our classification model, we need to represent the documents numerically. Therefore, we first tokenize the each document and turn them into sequence of integers (as required by embedding layer) where these integers are the index position of the token in the dictionary of words [3].

Encoding Labels

The labels also have to been encoded before training out, deep learning models. Therefore, we encode our labels between 0 and number of classes − 1 [2] [13].

Deep Learning Model

The model we use here consists of an input layer. The second layer is an embedding layer where we provide our embedding matrix of word vectors. Since these weights are previously learned by our word2vec model, we would stop our optimizer (we use Adagrad optimizer) training them further. The next layer, we use is a recurrent layer to learn the correlations between the words. In our case, we use GRU (Gated Recurrent Unit) with of 64 units. We will discuss more about recurrent layer in the further sections. We will also have a look at the performance of our model with and without this layer. Further, we use 3 Dense layers of [64,32,16] units respectively followed by dropout layers with dropout values 0.2. The activation function for each dense layer we use is the Rectified linear unit (ReLU)[3] and at the output layer, we use Softmax for getting the probability of classes predicted by our model.

For our deep learning model, we tried various hyperparameters, for example, batch size as [32, 64, 128], learning rates as [0.005,0.001], optimizers [SGD, Adam, Adagrad]. However, we achieved the best results at batch size 128, the number of epochs 22, optimizer Adagrad, and learning rate 0.005 Therefore, we finalize these parameters for our analysis. Moreover, we add dropout layers after each dense layers to tackle overfitting the model. The summary of our finalized model can be seen in fig 2.

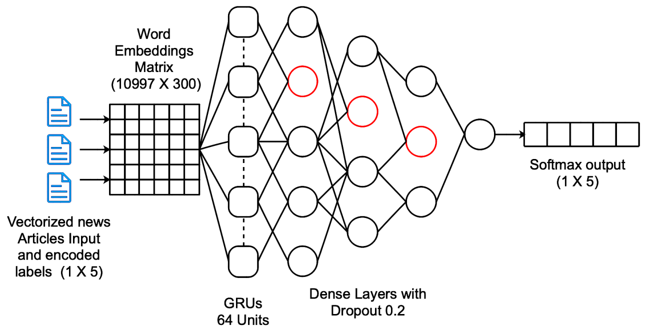
Word Embeddings Matrix (10997 X 300)
Vectorized news Articles Input and encoded labels (1 X 5)
GRUs 64 Units
Dense Layers with Dropout 0.2
Softmax output (1 X 5)

Figure 2: news classification model

GRU Layer

Gated Recurrent Unit layer is used for remembering the output of previous cells onto further units [3]. In our news analysis task, we need our classification model to remember the context for a word in the sentence. While experimenting, in our deep learning model when we did not add an GRU layer, we observed the problem of Vanishing gradient which resulted in a higher loss at each epoch and consistent lower accuracy. Therefore, the GRU layer helps overcome this problem.

Embedding Layer

Embedding Layer. The embedding layer in Keras [3] has been specially designed to handle textual data. The layer requires the data to be integer encoded, which we have done in the sequence generation section. Generally, the layer is initialized with random weights and later they are changed during backpropagation. But in our case, since we have already learned the weights using the Word2Vec model, we do not want them to change anymore. Generally, this is the first hidden layer of the deep learning model. In our case, we provide 4 parameters to this layer. Input dimension which is [size of vocabulary + 1] i.e. 10998, output dimension that is the size of vector space which is same as max sequence length of Word2Vec (i.e. 300), weights which have been already learned and trainable flag (i.e. False) to allow or stop the effect of backpropagation.

Callbacks

There could be a case when the evaluation metrics (accuracy in our case) stops to improve. In that case reducing the learning might help. Therefore, we monitor this issue while training our classification model. We monitor if the accuracy shows no improvement, after waiting for 2 epochs, we reduce the learning rate by factor 0.1 [3].

5.2 TASK II - Fine tuning BERT

Similar to previous task, we split the data into 1780 training and 445 test documents. There is no pre-processing required for fine-tuning BERT model in our case. However, BERT expects a fixed structure of the data in order to make predictions. It requires data to be tab separated. For training and validation set, we include columns ID, label, column with one same letter and last the text documents. On the other hand, test set is expected to have columns ID and the text documents. At last, the training and validation is not have any headers but test set should.

To fine-tune BERT for our data, we use a pre-trained case insensitive model, since we do not care much about the casing of words in our documents [4]. The model is called the BERT base which has 12 layers, 768 hidden and 12 heads having around 110 million pre-trained parameters [18]. Since the BERT model is pre-trained over the general domain corpus, we need it to be trained further on our domain-specific data [21]. The model has been trained on Wikipedia and book corpus. Fine-tuning for BERT is quite straight-forward because of its self-attention mechanism to model our news analysis task [4]. We simply provide our news documents to our pre-trained BERT model and fine-tune all the parameters end to end. Into the parameters, similar to what we have done in the case of Word2Vec, we will try to find out the best parameter amongst the given parameter sets.

Max Sequence Length. We keep max sequence length as we kept for training our Word2Vec model i.e. 300.

Train batch size. We select the best amongst 8,16 and 32.

Learning Rate. As given in original BERT paper, we choose the best learning rate amongst 5e-5, 4e-5, 3e-5, and 2e-5 [4]

Epochs. We keep the number of epochs 3.

6 RESULTS

While training Word2Vec with different parameter configurations as mentioned in the above sections, we found out

Table 1: Accuracy observed on validation set by switching the parameters while training Word2Vec model. CBOW-A represents CBOW accuracy and SG-A represents skip-gram model accuracy.

Min Count	Window	CBOW-A(%)	SG-A(%)
1	3	89.88	85.58
	5	89.43	83.78
	7	**86.03**	85.80
2	3	83.33	85.58
	5	85.11	82.88
	7	83.56	85.13
3	3	82.21	86.03
	5	84.90	85.80
	7	82.88	85.35
4	3	**86.03**	**90.25**
	5	89.33	89.68
	7	82.78	85.35

that when we kept minimum count as 4 and window size 3 we reached maximum accuracy using our news classification model in both CBOW and skip-gram. However, we get the same accuracy for CBOW when we had min count as 1 and window size 7. But in our case, we chose those parameters for further predictions which provides the best results for both CBOW and skip-gram. The performance can be seen in table 1.

The model provides almost similar results for all Word2Vec approaches. All of them results a lot of false positives and false negatives. The confusion matrix for the pre-trained model in terms of accuracy, can be seen in fig 3. The Word2Vec model achieves the highest 87.19% accuracy among all approaches. If we dig down deeper into the results, we observe that we get 87.48% precision for and 87.19% recall for the best approach. F1 score (micro averaged) for the classification 87.11%.

While fine-tuning the BERT model, the model is trained using various parameter settings. The maximum sequence is fixed that we discussed earlier. Although, we changed the batch size and the learning rate and fine-tuned the BERT model. The accuracy on the validation set can be seen in table 2 when fine-tuned with different parameter settings. In this case also we choose the best ones i.e batch size 8 and learning rate 2e-5.

In the case of Finetuning with BERT, we get far better results than the Word2vec. We observed the accuracy of 97.30%, 97.34%, and 97.30% weighted averaged precision and recall respectively. The confusion matrix for BERT can be seen in fig. 4.

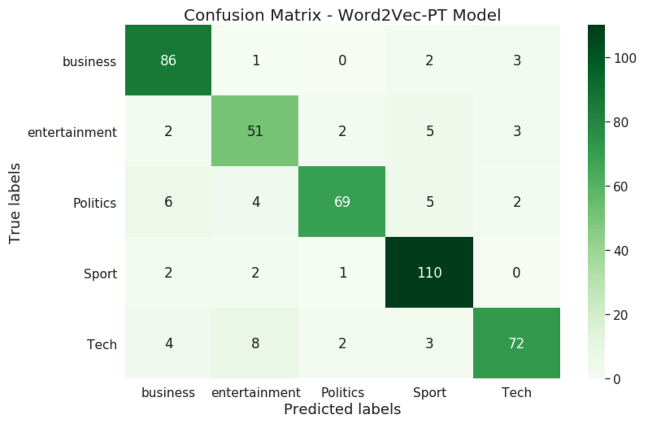

Figure 3: Confusion Matrix for Word2Vec Pretrained Model

Table 2: Accuracy observed on validation set by switching the parameters while Fine tuning BERT

Batch Size	Learning Rate	Accuracy (%)
8	2e-5	**98.19**
	3e-5	97.07
	4e-5	97.75
	5e-5	97.52
16	2e-5	97.30
	3e-5	97.07
	4e-5	97.52
	5e-5	96.85
32	2e-5	97.97
	3e-5	97.30
	4e-5	97.30
	5e-5	96.40

Upon a deeper look at the confusion matrices produced by Word2Vec and BERT models, we observe that while training the both models, we had 91 documents of a class business, 80 articles of entertainment, 88 articles related to politics, 104 of sport and 82 of class tech. To decide, which model tends to perform better, we calculate error rate [20] using the formula (1) for each class, and plot a graph in fig. 5 to see which model has performed the best amongst all.

$$ErrorRate = \frac{FP + FN}{Positives + Negatives} \tag{2}$$

From Fig. 5, it is clear that the results of Fine-tuned should be preferred amongst all of our models. It has a very small number of misclassified articles compare to all other models. Although, the pre-trained Word2Vec model could not make a huge difference in making predictions for our news articles.

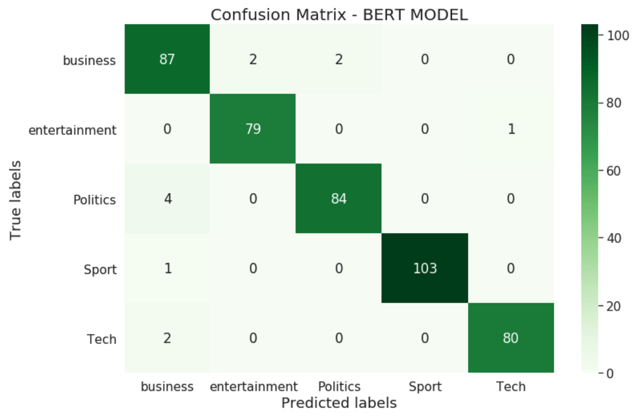

Figure 4: Confusion Matrix for Fine tuning using BERT

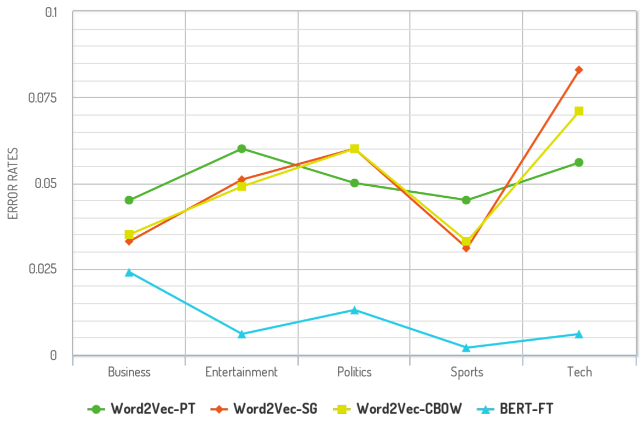

Figure 5: Error rates for each class made by all models

It performs worst for 3 classes than the skip-gram model which we trained from scratch. But it performs better for other classes than the other Word2Vec models.

In table 1, we observe a summary of precision, recall and F1 scores of all different experiments that we carried out in the paper. We use abbreviations like Word2Vec CBOW for the continuous bag of words, Word2Vec SG for skip-gram, Word2Vec PT for the pre-trained model. Even here, When it comes to making correct predictions, BERT outperforms the rest. Note that, in our results, we consider weighted precision and recall, since it can be noticed that there is an imbalance of documents in our test data and we want all the classes not to be equally weighted in our predictions.

7 CONCLUSION

A language model needs to retain the context of the words and make predictions as precise as possible because use cases of Natural language processing are increasing day-by-day and this technology is not limited to engineers anymore.

Table 3: Weighted Precision, Recall and F1 Score resulted by various models on test set.

Model	Precision	Recall	F1 Score
Word2Vec-CBOW	87.68	86.35	87.01
Word2Vec-SG	86.83	86.96	86.69
Word2Vec-PT	87.48	87.19	87.11
BERT-FT	**97.34**	**97.30**	**97.31**

Therefore, contextual models like BERT will be extremely helpful in the upcoming breakthroughs in this area. In this paper, we performed a multi-class classification for the news analysis dataset. We compared context-free and contextual models and their performances for making our predictions on the news analysis dataset. One model is Word2Vec, that is we call context-free model which is trained over 2225 news articles. We also use a pre-trained Word2Vec model [7]. The pre-trained Word2Vec [7] performs better for tech and politics articles than other Word2Vec models but not it is not the best in our experiment. We build a deep learning model for classification using word embeddings of all the Word2Vec models. This model includes one layer of GRUs to keep the context as much as possible, 3 dense layers followed by dropout layers with a dropout value of 0.2 to avoid overfitting. Another model we built on the top of the pre-trained language model to perform our downstream task of news classification. We evaluated our models using various matrices like accuracy, weighted precision and recall and error rates. The BERT model resulted in a great performance and outperformed our deep learning classification model with Word2Vec embeddings. However, in industrial usage, it is natural that we train over millions of documents and therefore using a pretrained model may save time and computational power. But, pretraining BERT from scratch at the production level (or with larger datasets) could be highly computationally expensive. On the other hand, Word2Vec is computation friendly, because of its single hidden layer architecture [11].

REFERENCES

[1] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".
[2] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 108–122.
[3] François Chollet et al. 2015. Keras. https://keras.io.
[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language

understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis* (1957).

[6] Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies* 10, 1 (2017), 1–309.

[7] Google Code Archive. 2013. Word2Vec. https://code.google.com/archive/p/word2vec/, Last accessed on 2020-01-13.

[8] Derek Greene and Pádraig Cunningham. 2006. Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*. ACM Press, 377–384.

[9] Faiza Khan Khattak, Serena Jeblee, Chloé Pou-Prom, Mohamed Abdalla, Christopher Meaney, and Frank Rudzicz. 2019. A survey of word embeddings for clinical text. *Journal of Biomedical Informatics: X* (2019), 100057.

[10] Amit Mandelbaum and Adi Shalev. 2016. Word embeddings and their use in sentence classification tasks. *arXiv preprint arXiv:1610.08229* (2016).

[11] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[14] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[15] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[16] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf* (2018).

[17] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50. http://is.muni.cz/publication/884893/en.

[18] Julian Risch, Anke Stoll, Marc Ziegele, and Ralf Krestel. 2019. hpiDEDIS at GermEval 2019: Offensive Language Identification using a German BERT model. In *Preliminary proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019). Erlangen, Germany: German Society for Computational Linguistics & Language Technology*. 403–408.

[19] Amir Sadeghian and Ali Reza Sharafat. 2015. Bag of words meets bags of popcorn. (2015).

[20] Takaya Saito and Marc Rehmsmeier. 2015. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one* 10, 3 (2015), e0118432.

[21] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to Fine-Tune BERT for Text Classification? *arXiv preprint arXiv:1905.05583* (2019).

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[23] Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 2019. Does BERT Make Any Sense? Interpretable Word Sense Disambiguation with Contextualized Embeddings. *arXiv preprint arXiv:1909.10430* (2019).