Does a Hybrid Neural Network based Feature Selection Model Improve Text Classification?

Suman Dowlagar
LTRC
IIIT-Hyderabad
suman.dowlagar@
research.iiit.ac.in

Radhika Mamidi
LTRC
IIIT-Hyderabad
radhika.mamidi@
iiit.ac.in

Abstract

Text classification is a fundamental problem in the field of natural language processing. Text classification mainly focuses on giving more importance to all the relevant features that help classify the textual data. Apart from these, the text can have redundant or highly correlated features. These features increase the complexity of the classification algorithm. Thus, many dimensionality reduction methods were proposed with the traditional machine learning classifiers. The use of dimensionality reduction methods with machine learning classifiers has achieved good results. In this paper, we propose a hybrid feature selection method for obtaining relevant features by combining various filter-based feature selection methods and fastText classifier. We then present three ways of implementing a feature selection and neural network pipeline. We observed a reduction in training time when feature selection methods are used along with neural networks. We also observed a slight increase in accuracy on some datasets.

1 Introduction

Text classification assigns one or more class labels from a predefined set to a document based on its content. Text classification has broad applications in real-world scenarios such as document categorization, news filtering, spam detection, Optical character recognition (OCR), and intent recognition. Giving high weights to relevant features is the objective of text classification.

The field of text classification has gained more interest during the machine learning (ML) era. Many discriminative and generative machine learning classifiers have achieved excellent results in the field of text classification (Deng et al., 2019). Feature selection and feature extraction methods are often used to reduce high dimensionality (Bharti and Singh, 2015). Feature extraction generates features from text (Agarwal and Mittal, 2014). Feature selection (FS) selects the most prominent features (Saleh and El-Sonbaty, 2007).

These feature selection and extraction methods are used along with traditional classification algorithms. These methods reduced the curse of dimensionality and increased the classification accuracy (Deng et al., 2019).

Recently, deep learning models are used to learn better text representations and to classify the text (Minaee et al., 2020). Such models include convolutional neural networks (CNN) (Kim, 2014), recurrent neural networks (RNN) (Hochreiter and Schmidhuber, 1997), Transformer models (Adhikari et al., 2019), and graph convolutional networks (GCN) (Yao et al., 2019). These NN models capture semantic and syntactic information in local and global word sequences.

Even though the neural networks capture a complex and dense representation of data, the set of words introducing noise in the classifier is still present. Such words add the burden of increased vocabulary, which results in increased textual representation and an increase in the training time of the classifiers (Song et al., 2011).

Similar to the traditional approaches, we want to understand the effects of using statistical feature selection algorithms beforehand to calculate the features' relevance and then train a fastText text-classification algorithm on those relevant features. Using this feature selection and neural network pipeline, we assume that the complexity of dealing with larger vocabulary decreases. Including feature selection with fastText text-classification helps reduce the classifier's training time and helps the classifier reach better local optima, showing a significant increase in classification accuracy.

In this work, we analyzed a feature selection and neural network pipeline for text classification.

Proceedings of the 17th International Conference on Natural Language Processing, pages 272–280
Patna, India, December 18 - 21, 2020.

We used a hybrid feature selection method to get a score on relevant features. Using this score, we formulated three methods. The first and second methods deal with modifying the original text by extracting the relevant features. The third method deals with using the feature selection scores and pass it along with the word embeddings. We then observed the effect of feature selection on various neural networks.

The rest of the paper is organized as follows. Section 2 gives a brief review of previous works in the field of feature selection and text classification. Section 3 presents a detailed procedure of the proposed pipeline and presents the experiments and datasets used for our study. Section 4 reports the performance of text classifiers with and without feature selection methods. Section 5 concludes the paper.

2 Literature Survey

This section presents a brief description of the neural network (NN) classification algorithms and various feature selection methods.

2.1 Deep Learning for Text Classification

Nowadays, various NNs such as CNN, RNN, BERT, and Text GCN achieve state-of-the-art results on text classification. CNN uses 1d convolutions (Zhang et al., 2015) and character level convolutions (Conneau et al., 2016) to learn the semantic similarity of words or characters, which helps in classifying the text. RNN models such as GRU, LSTM, and BiLSTM (Liu et al., 2016) take word to word sequences to learn a better textual representation of a document that helps in text classification. Attention mechanisms have been introduced in these LSTM models, which increased the representativeness of the text for better classification (Yang et al., 2016). Transformer models such as BERT (Devlin et al., 2018) uses the attention mechanism that learns contextual relations between words or sub-words in a text (Adhikari et al., 2019). Text GCN (Yao et al., 2019) uses a graph-convolutional network to learn a heterogeneous word document graph on the whole corpus. Text GCN can capture global word co-occurrence information and use graph convolutions to learn a global representation, which helps classify the documents.

2.2 Feature Selection on Text Data

The text classification often involves extensive data with thousands of features. Although tens of thousands of words are in a typical text collection, most of them contain little or no information to predict the text label. These features introduce complexity and increase the training time of an ML classifier. Feature selection is one method for giving high scores to relevant features (Deng et al., 2019). The goal of feature selection is to select highly-relevant features with minimum redundancy. The relevance of a feature indicates that the feature is always necessary to predict the class label.

There are various text feature selection methods in the literature, each being filter, wrapper, hybrid, and embedded methods. The filter method evaluates the quality of a feature using a scoring function. Some filter methods evaluate the goodness of a term based on how frequently it appears in a text corpus. Document Frequency (DF) (Lam and Lee, 1999) and Term Frequency - Inverse Document Frequency (TFIDF) (Rajaraman and Ullman, 2011) comes under this category. Other filter methods that originate from information theory are, Mutual Information (MI) (Taira and Haruno, 1999; Tang et al., 2019), Information Gain (IG) (Yang and Pedersen, 1997), CHI (Rogati and Yang, 2002), ANOVA F measure (Elssied et al., 2014), Bi-Normal Separation (BNS) (Forman, 2003) and the GINI method (Shang et al., 2013). They use hypothesis testing, contingency tables, mean and variance scores, conditional and posterior probabilities for selecting the features.

The wrapper method (Maldonado and Weber, 2009) use a search strategy to construct each possible subset, feeds each subset to the chosen classifier, and then evaluates the classifier's performance. These two steps are repeated until the desired quality of the feature subset is reached. The wrapper approach achieves better classification accuracy than filter methods. However, the time taken by the wrapper method is very high when compared to filter methods.

Embedded methods complete the FS process within the construction of the machine learning algorithm itself. In other words, they perform feature selection during the model training. An embedded method is Decision Tree (DT) (Quinlan, 1986). In DT, while constructing the classifier, DT selects the best features/attributes that may give the best discriminative power.

Hybrid methods are robust and take less time when compared to the wrapper and embedded methods. They combine a filter method with a wrapper method during the feature selection process. The HYBRID model (Günal, 2012) employs a combination of filter methods to select to rank the features and then a wrapper method to the obtained final features set. Our FS method is similar to the HYBRID model.

A detailed report on the benefits of using the feature selection methods in the pipeline with traditional classifiers is presented in Deng et al. (2019); Forman (2003).

Apart from using traditional classification methods, deep feature selection using neural networks were also proposed. These models use deep neural network autoencoders for the feature set reduction and text generation (Mirzaei et al., 2019; Han et al., 2018).

Lam and Lee (1999) studies the effect of feature set reduction before applying the neural network classifiers. The paper uses a multi-layer perceptron (MLP) classifier in combination with filter-based FS method. Alkhatib et al. (2017) proposes the use of neural network-based feature selection and text classification. Our work comes under this category.

3 Proposed Pipeline

In this section, we present our feature selection and neural network pipeline.

The feature selection and neural network pipeline start with selecting a good tokenizer to tokenize the data and create a feature set. The tokenizer used for our feature selection is the Sentencepiece tokenizer (Kudo and Richardson, 2018). Sentencepiece tokenizer implements subword units by using byte-pair-encoding (BPE) (Sennrich et al., 2015) and unigram language model (Kudo, 2018). In the feature subset generation, we considered a hybrid feature selection method known as HYBRID (Günal, 2012). It has proved that a combination of the features selected by various methods is more effective and computationally faster than the features selected by individual filter and wrapper methods. Similar to the HYBRID model, we used three filters to obtain the relevancy score. The filters we considered were CHI2, ANOVA-F, and MI. These filters calculate the relevancy between the word and the class labels.

Before feature selection, we used the Bag-of-Words(BoW) model to vectorize the data. In the BoW model, each feature vector is represented by TF_IDF scores.

Then we used statistical measures such as χ^2, ANOVA-F, and MI for obtaining feature scores.

χ^2 [1] is a statistic to measure a relationship between feature vectors and a label vector.

Analysis of Variance (ANOVA[2]) is a statistical method used to check the means of two or more groups that are significantly different from each other.

Mutual Information (MI[3]) is frequently used to measure the mutual dependency between two variables.

Using different statistical methods, we measured the relevance of each feature. We then aggregate the relevance scores of all satistical methods for each feature. The relevance of a feature x_i is given by,

$$Relevancy(x_i) = \chi^2(x_i) + ANOVA(x_i) + MI(x_i) \quad (1)$$

Instead of an LR classifier given in the HYBRID model, we used the fastText classifier (Joulin et al., 2016) for the feature selection. We used the fastText classifier as it is often on par with deep learning classifiers in terms of accuracy and performs faster computations. The fastText classifier treats the average of word embeddings as document embeddings, then feeds document embeddings into a feed-forward NN or a multinomial LR classifier. We used pre-trained fastText word embeddings (Grave et al., 2018) while training a classifier.

To get the final features list, we sorted the normalized, aggregated value in descending order and divided the entire feature space into k sets. In our model, we divided the sorted feature space into 20 sets. The value of k is fixed to 20 using a trial and error basis. We take the first set as the vocabulary of the classifier. We then trained the classifier and noted its accuracy. In the second iteration, we considered the vocabulary as the combination of first

[1] A detailed explanation and a simple example of χ^2 is given at https://www.mathsisfun.com/data/chi-square-test.html

[2] A detailed explanation for ANOVA is given in https://towardsdatascience.com/anova-for-feature-selection-in-machine-learning-d9305e228476.

[3] A simple explanation and working python example of MI is available at https://machinelearningmastery.com/information-gain-and-mutual-information/

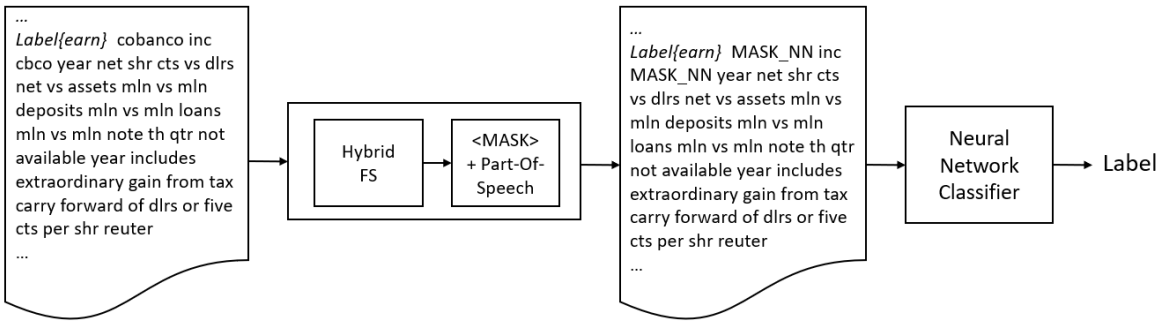

Figure 1: Modifying text by masking the low ranked words

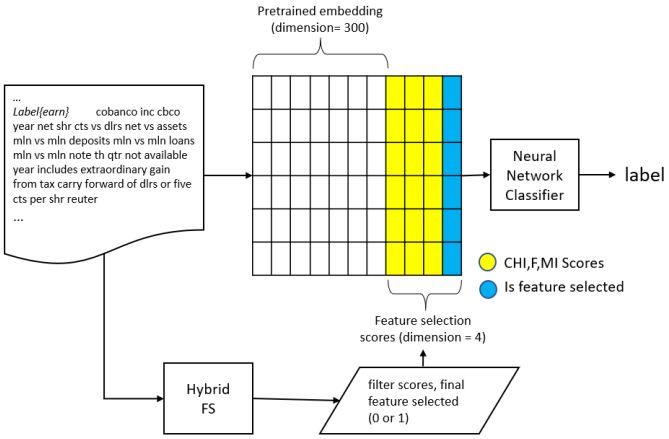

Figure 2: Meta-Embeddings, including feature scores along with word embeddings

and second sets. Similarly, the third set has the vocabulary of the first three sets combined. We repeated the process until all the lists are exhausted. The set of features that resulted in a better classification metric is considered as the final feature set.

According to the proposed FS method, the final feature set is considered relevant, and they are necessary to perform the text classification. In contrast, the other features have little to no effect on the text classification or might degrade the classifier's performance.

After feature subset generation, we propose three methods for including the feature selection information before training the neural network classifiers.

1. **Method 1 (Selecting only the relevant features)**[4]**:** Like traditional classification algorithms, we select only the relevant features that are estimated to be important by the feature selection method before training a neural network classifier.

2. **Method 2 (Masking the features that were given low importance by our FS method):** We felt that removing the features given low rank by our FS method might disturb the original data's grammatical structure, thus disturbing the word to word dependencies. We masked the low ranked words with the help of $< MASK > +POS(word)$ tag. $< MASK >$ word masks the low ranked word, and POS preserves the word's part of speech. The visual representation of method 2 is shown in figure 1

3. **Method 3 (Meta Embeddings):** As shown in figure 2, we pass the relevancy and feature selection information along with embeddings in this method. Each slot holds the filter scores, i.e., CHI, ANOVA, MI scores of each feature. The last slot holds a 1 or 0 value. 1 is used for the selected features, and 0 is used for low ranked features that were not selected by our hybrid feature selection approach.

We analyzed and evaluated the above methods with various state-of-the-art NN classifiers on the benchmark datasets.

3.1 Experiment

In this section, we evaluated our feature selection and neural network pipeline on two tasks. We wanted to determine:

- If the pipeline decreases the training time of the classifier
- If it helps in obtaining better local optima, thus improving the classification accuracy.

We tested our pipeline across multiple state-of-the-art text classification algorithms.

[4]This method is already used while selecting the final features set by the fastText classifier.

1. **CNN:** (Kim, 2014) This convolutional neural network-based text classifier is trained by considering pre-trained word vectors.

2. **Bi-LSTM:** (Liu et al., 2016) A two-layer, bi-directional LSTM text classifier with pre-trained word embeddings as input was considered for the task of text classification.

3. **fastText:** (Joulin et al., 2016) This is a simple, efficient, and the fastest text classification method. It treats the average of word/n-grams embeddings as document embeddings, then feeds document embeddings into a linear classifier.

4. **Text GCN:** (Yao et al., 2019) Builds a heterogeneous word document graph for a whole corpus and turns document classification into a node classification problem. It uses GCN (Kipf and Welling, 2017) to learn word and document embeddings.

5. **DocBERT:** (Adhikari et al., 2019) A fine-tuned BERT model for document classification. The BERT model (Devlin et al., 2018) uses a series of multiheaded attention and feedforward networks for various NLP tasks.

3.2 Datasets

We ran our experiments on three widely used benchmark corpora and multilingual corpora. They are 20Newsgroups(20NG), R8, and R52 of Reuters 21578 and MLMRD.

- The 20NG dataset contains 18,846 documents divided into 20 different categories. 11,314 documents were used for training, and 7,532 documents were used for testing.

- R52 and R8 are two subsets of the Reuters 21578 dataset. R8 has 8 categories of the top eight document classes. It was split into 5,485 training and 2,189 test documents. R52 has 52 categories and was split into 6,532 training and 2,568 test documents.

- MLMRD is a Multilingual Movie Review Dataset. It consists of the genre and synopsis of movies across multiple languages, namely Hindi, Telugu, Tamil, Malayalam, Korean, French, and Japanese. The data set is minimal and unbalanced. It has 9 classes and a total of 14,997 documents. The data was split into 10,493 training and 4,504 test documents.

We first preprocessed all the datasets by cleaning and tokenizing. The tokenizer used is the fastText tokenizer.

For baseline 1 models, we used multilingual fastText embeddings (Grave et al., 2018) of dimensionality 300, and baseline 2 models had the dimensionality of 304. We used default parameter settings as in their original papers for implementations. For calculating TFIDF, CHI2, ANOVA-F, MI scores, we used the scikit-learn library (Pedregosa et al., 2011). For POS tagging, we used the NLTK (Bird et al., 2009) pos tagger.

All the neural network models were run on the GPU processor on the Windows platform with NVIDIA RTX 2070 graphics card.

4 Performance

Datasets	Our FS	HYBRID FS
20Newsgroups	**81.27%**	77.34%
R8	**96.94%**	93.79%
R52	**92.72%**	86.43%
MLMRD	**47.09%**	42.98%

Table 1: The classification accuracy of our FS model when compared to the HYBRID model.

In our work, we modified the HYBRID (2012) feature selection model by changing the LR classifier to the fastText classifier. We selected the fastText classifier in the feature selection process because of its fast learning ability of a NN model compared to the traditional ML classifiers and other neural network classifiers (Joulin et al., 2016) without any decrease in classification accuracy. The neural network classifiers such as MLP, CNN, RNN, transformer, and GCN models achieve better classification accuracy when compared to traditional ML classifiers, but their training time is very high.

Using a fastText classifier during feature selection, we observed that our model performed better on all the benchmark datasets than the HYBRID model. The results are shown in table 1. The fastText classifier's use helped the model obtain better relevant features, increasing the current feature selection model's accuracy compared to the HYBRID model.

As mentioned above, we used the training time-taken and test accuracy as the metrics to evaluate our approach. The accuracy and training time are recorded by running the model 10 times, and the

Datasets	20Newsgroups	R8	R52	MLMRD
Baseline 1 & 2	1,01,631 (V)	19,956 (V)	26287 (V)	94073 (V)
Method 1	25732 (0.25V)	17364 (0.87V)	22372 (0.85V)	52015 (0.55V)
Method 2	25732+30 (0.25V)	17364+30 (0.87V)	22372+30 (0.85V)	52015+143 (0.55V)
Method 3	1,01,631 (V)	19,956 (V)	26287 (V)	94073 (V)

Table 2: The vocabulary size in all the FS inclusion methods when compared to the baselines. "V" is denoted as the vocabulary size of the actual data. Baselines 1,2, and method 3 have no change in vocabulary. However, using our FS method, the vocabulary is reduced to a maximum of 75% (for 20Newsgroups data). Other datasets have seen a 13% to 45% decrease in vocabulary size. We can see an increase in vocabulary from method 1 to method 2. It is due to the additional vocabulary resulted from the mask words when they are accompanied by pos tags. Here Penn Treebank POS tagset is used.

Datasets	Method	Classifier(s)				
		CNN	**Bi-LSTM**	**fastText**	**DocBERT**	**Text GCN**
20Newsgroups	Baseline 1	79.31%	73.60%	81.04%	**90.19%**	86.13%
	Baseline 2	79.46%	74.25%	82.44%	NA	86.23%
	Method 1	78.27%	73.44%	81.27%	89.37%	**86.25%**
	Method 2	77.29%	70.48%	80.14%	88.43%	85.65%
	Method 3	**80.59%**	**76.57%**	**84.48%**	NA	86.15%
R8	Baseline 1	97.24%	92.70%	96.13%	**97.62%**	96.80%
	Baseline 2	97.37%	93.82%	96.50%	NA	**96.94%**
	Method 1	**97.39%**	93.74%	96.94%	97.44%	96.28%
	Method 2	96.57%	94.34%	96.07%	97.44%	96.85%
	Method 3	**97.39%**	**96.74%**	**97.18%**	NA	**96.94%**
R52	Baseline 1	94.78%	87.53%	92.02%	92.95%	93.56%
	Baseline 2	**94.84%**	90.79%	92.76%	NA	93.64%
	Method 1	94.29%	87.47%	92.72%	**93.10%**	92.97%
	Method 2	91.71%	**91.90%**	90.30%	92.10%	93.19%
	Method 3	**94.84%**	91.48%	**92.83%**	NA	**93.74%**
MLMRD	Baseline 1	47.63%	46.43%	46.92%	**53.11%**	47.62%
	Baseline 2	47.79%	47.43%	48.92%	NA	49.62%
	Method 1	44.98%	44.82%	47.09%	51.90%	46.58%
	Method 2	44.63%	44.05%	46.61%	50.90%	46.98%
	Method 3	**48.44%**	**49.13%**	**49.55%**	NA	**51.50%**

Table 3: Test accuracy on various neural network classifiers for the task of document classification. As the BERT model used is a fine-tuned one, we did not modify the model.

average of the metrics was presented.

4.1 Effects of our methods on classification accuracy

Table 3 demonstrates the accuracy of feature selection methods on NN classifiers.

When methods 1 and 2 were used, there is a slight decrease in classification accuracy because the first two methods lost semantic connection among words. Thus, the classification performance is degraded. Also, some words which were relevant to the classifier were masked out during the FS method. Whereas in method 3, including the feature selection scores with word-embeddings, has shown a significant improvement in accuracy on all the datasets.

Compared to the other datasets, the 20NG dataset has seen a significant decrease in vocabulary size. The vocabulary was decreased by 75%. However, eliminating those features did not affect the accuracy of the classifier for methods 1 and 2.

Introducing the masked features in method 2 shown an increase in accuracy only in the Bi-LSTM method as this method considers word dependencies while training a classifier.

Including the feature selection scores along with

the word-embeddings improved the classification accuracy on all the datasets. The feature selection metadata helped the neural network classifier learn a better relationship between the words and classes and improve the classifier's accuracy by reaching better local optima.

In R8 and R52 datasets, we have seen an increase in accuracy using method 1 because our hybrid FS method worked better on these datasets by removing the noisy words without disturbing the relevant words. The maximum improvement in accuracy is shown in the R8 dataset, with a +4% increase in classification accuracy.

Our approach did not show any better results on MLMRD datasets as this dataset has a limited number of documents to train and test the data for some languages (Telugu, Tamil, Malayalam, Korean). Reducing vocabulary size by the FS method decreased the classification accuracy.

4.2 Effects of our methods on training time

The pictorial representation of time taken by the classifiers for all the datasets is given in appendix B of the supplementary material.

The time taken by method 1 is lower than in all baseline models. In method 1, as the text is modified by considering only relevant features, the vocabulary size is reduced, and the sentence length is reduced. It resulted in the more accelerated training of the neural network.

The time taken by baseline 2 and method 3 is similar because of the same embedding dimensionality of 304, but method 3 has achieved local optima a few epochs before compared to baseline 2, resulting in a time decrease of a few seconds. This phenomenon is attributed to the use of feature selection scores along with word embeddings.

Method 2 has shown an increase in training time even though the vocabulary is decreased because of 2 factors.

1. The masking of features created unknown words in the data, and the classifier has to be trained to learn the representation of masked words, whereas the other words had pre-trained embeddings.

2. Apart from vocabulary, the neural network training time also depends on the input batch size given to the network and the length of the sentence in each batch. Because of the masked words, there is no decrease in either batch size or the sentence length. So the masking of data did not decrease the training time of the classifier.

On the contrary, the Text GCN model has shown a decrease in training time because the classifier computes heterogeneous graph embeddings of each word based on the textual data before classification. It did not use any pre-trained embeddings.

In method 3, there is a slight increase in training time because of increased vocabulary size due to the inclusion of feature selection metadata.

Of all the NN classifiers, the Text GCN model had shown a maximum decrease in training time by 488 sec when method 1 was used on 20NG data. As the Text GCN operates on building a graph on the complete vocabulary of data, the time taken by the method to build the graph is reduced significantly by reducing the vocabulary size. It is followed by the DocBERT and Bi-LSTM on 20NG data with a decrease in training time by 480 and 394 sec. Text GCN and Bi-LSTM have shown a significant decrease in training time on all the datasets. On the contrary, fastText and CNN are very fast while training the NN model. The training time of such models was unchanged when our method 1 was used.

When compared to all the classifiers, DocBERT achieved better results because of its evolutionary multi-headed attention and transformer models. As the Text GCN captures both local and word embeddings by constructing a heterogeneous graph, their results were better than those of the CNN and Bi-LSTM models, which work only on local word dependencies. As we increased the size of the embedding in FS method 2, this increased the dimensionality of vocabulary, resulting in the classifier's increased training time.

5 Conclusion

In our work, "Does a Hybrid NN FS Model Improve Text Classification?", we used the NN based hybrid FS method to extract relevant features and used NN classifiers for text classification. We extracted the relevant or high ranked features using filter-based methods and a fastText classifier. We then proposed three methods on how the feature selection can be included in the NN classification process. First, modifying the corpus by considering only relevant features. Second, modifying the data by masking the low ranked features, and

the third method introduces feature selection information along with word embeddings. We observed that method 1 had shown a significant reduction in training time when large datasets or slower models are used, accompanied by a minimal change in classification accuracy. By introducing $MASK + P0S(word)$, we inferred that the masked word was a burden to the classifier, and it always tried to adjust the word embeddings, which resulted in increased epoch time during training and a slightly negative effect on classification accuracy. Whereas method 3 has shown no effect on decreasing the training time, it has shown a maximum of 4% increase in the classification accuracy compared to baseline. It proved that introducing feature scores along with pre-trained word embeddings while training the NN classifier is beneficial.

Instead of opting for random naive vocabulary reduction techniques such as using min_df and max_df (minimum and maximum document frequency) for selecting features, by using FS methods, we can calculate the relevance of the word beforehand and use that as metadata to the NN classifier. When the datasets are huge, these methods are of more significance. We can use the modified data while tuning the hyperparameters. Then we can use the real data to train and evaluate the model. Even in the critical domain datasets such as "medical", we cannot rely on removing a word based on min_df and max_df scores. Each word in those datasets should be treated with utmost significance. FS methods help in such scenarios by calculating the word's relevance and helps maintain better vocabulary before training neural network classifiers.

References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*.

Basant Agarwal and Namita Mittal. 2014. Text classification using machine learning methods-a survey. In *Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), December 28-30, 2012*, pages 701–709, New Delhi. Springer India.

Wael Alkhatib, Christoph Rensing, and Johannes Silberbauer. 2017. Multi-label text classification using semantic features and dimensionality reduction with autoencoders. In *International Conference on Language, Data and Knowledge*, pages 380–394. Springer.

Kusum Kumari Bharti and Pramod Kumar Singh. 2015. Hybrid dimension reduction by integrating feature selection with feature extraction method for text clustering. *Expert Systems with Applications*, 42(6):3105–3114.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

Xuelian Deng, Yuqing Li, Jian Weng, and Jilian Zhang. 2019. Feature selection for text classification: A review. *Multimedia Tools and Applications*, 78(3):3797–3816.

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*.

Nadir Omer Fadl Elssied, Othman Ibrahim, and Ahmed Hamza Osman. 2014. A novel feature selection based on one-way anova f-test for e-mail spam classification. *Research Journal of Applied Sciences, Engineering and Technology*, 7(3):625–638.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. *arXiv preprint arXiv:1802.06893*.

Serkan Günal. 2012. Hybrid feature selection for text classification. *Turkish Journal of Electrical Engineering and Computer Science*, 20(Sup. 2):1296–1311.

Kai Han, Yunhe Wang, Chao Zhang, Chao Li, and Chao Xu. 2018. Autoencoder inspired unsupervised feature selection. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2941–2945. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

TN Kipf and M Welling. 2017. Semi-supervised classification with graph convolutional networks iclr.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Savio LY Lam and Dik Lun Lee. 1999. Feature reduction for neural network based text categorization. In *Proceedings. 6th international conference on advanced systems for advanced applications*, pages 195–202. IEEE.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Sebastián Maldonado and Richard Weber. 2009. A wrapper method for feature selection using support vector machines. *Information Sciences*, 179(13):2208–2217.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2020. Deep learning based text classification: A comprehensive review. *arXiv preprint arXiv:2004.03705*.

Ali Mirzaei, Vahid Pourahmadi, Mehran Soltani, and Hamid Sheikhzadeh. 2019. Deep feature selection using a teacher-student network. *Neurocomputing*.

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*, 12:2825–2830.

J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning*, 1(1):81–106.

Anand Rajaraman and Jeffrey David Ullman. 2011. *Mining of massive datasets*. Cambridge University Press.

Monica Rogati and Yiming Yang. 2002. High-performing feature selection for text classification. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, page 659–661, New York, NY, USA. Association for Computing Machinery.

S. N. Saleh and Y. El-Sonbaty. 2007. A feature selection algorithm with redundancy reduction for text classification. In *2007 22nd international symposium on computer and information sciences*, pages 1–6.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Changxing Shang, Min Li, Shengzhong Feng, Qingshan Jiang, and Jianping Fan. 2013. Feature selection via maximizing global information gain for text classification. *Knowledge-Based Systems*, 54:298–309.

Qinbao Song, Jingjie Ni, and Guangtao Wang. 2011. A fast clustering-based feature subset selection algorithm for high-dimensional data. *IEEE transactions on knowledge and data engineering*, 25(1):1–14.

Hirotoshi Taira and Masahiko Haruno. 1999. Feature selection in svm text categorization. In *AAAI/IAAI*, pages 480–486.

Xiaochuan Tang, Yuanshun Dai, and Yanping Xiang. 2019. Feature selection based on feature interactions with application to text categorization. *Expert Systems with Applications*, 120:207–216.

Yiming Yang and Jan O Pedersen. 1997. A comparative study on feature selection in text categorization. In *Icml*, volume 97, page 35.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.