Comparison between feature engineering and classification algorithms for text classification

Marcela Ribeiro de Oliveira[1], Thiago Jorge Abdo[2]
Programa de Pós Graduação em Informática
{mro15[1], tja14[2]}@inf.ufpr.br

I. Introduction

One classical problem in natural language processing (NLP) is the problem of assign one or more labels to a textual document, known as **Text Classification Problem**. A few examples of textual documents are: emails, reviews and news articles. The labels commonly depends of the type of textual documents, for instance, reviews can be labeled as positive or negative and emails as spam and not spam.

Text classification is a problem found a wide variety of domains in text mining. News filtering and organization, document organization and retrieval, opinion mining, email classification and spam filtering are a few examples of domains where text classification is required [1].

Text classification research goes from designing the best features to choosing the best possible machine learning classifiers [2].

To solve a text classification problem two main tasks should be performed: feature engineering and classification using machine learning algorithms. In the feature engineering step each text document is processed to get a set of features that represents this text in a vector space. This features will be used as input of the classification algorithm, which will map each vector (text representation) to a label.

In this work we reproduce and compare some techniques found in the literature for text classification. For feature engineering we compare Bag of Words (BOW), Term Frequency - Inverse Document Frequency (TF-IDF) and Word2Vec, a Word Embedding method . The classifiers use are K-Nearest Neighbors (K-NN), Support Vector Machine (SVM), Multi Layer Perceptron (MLP), Naive Bayes and Random Forest. To evaluate these classifiers, the accuracy and F1 measures are used.

The remainder of the paper is organized as follows Section II presents the related works, Section III briefly overviews the theoretical foundations, Section IV presents the experiments and finally Section V presents the results from the experiments.

II. Related work

In [3], Korde et al brings a survey about text classification and classifiers. The authors define the classification process in six stages: document collection, pre-processing, indexing, feature selection, classification and performance evaluation. Each of these stages are explained in the survey. In addition, the authors present commonly used classifiers, how these classifiers work including a discussion of their parameters. Among the discussed classifiers are: neural network, K-NN, SVM, and naive bayes.

Another survey about text classification have been done by Ikonomakis et al [4], which illustrates text classification process using machine learning techniques. According to the authors, the task of constructing a classifier for documents does not differ too much from other Machine Learning tasks. The main issue is the document representation. In this survey, besides discussing classification and evaluation algorithms, the authors present the most common metrics for feature subset selection, the goal is the reduction of the dimensionality curse to yield improved classification accuracy.

III. Theoretical Foundation

In this session are presented a brief explanation about each feature engineering technique and classifier evaluated in this work.

A. Feature engineering

Bag of words: is a simple representation used in natural language processing and information retrieval (IR). In this model, a text is represented by a vector (array) of the size of the vocabulary. Each position represents a word, and its value (weight) is the number of occurrences of that word in the text. As a consequence this representation does not preserve grammar nor word order [5].

TF-IDF: works by determining the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus. Intuitively, this calculation determines how relevant a given word is in a particular document [6].

Word2vec: is a word embedding method proposed by google engineers, this method tries to learn language structures and generates a vector representation of words with syntactic and semantic relationships [7]. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are close to one another in the space, it was successfully applied in language translation [8].

B. Classification algorithms

K-NN: (K-Nearest Neighbors) is a classifier that assigns to an unlabeled example the most frequent label in its **k** nearest neighbors, where **k** is a parameter of the algorithm. To find the nearest neighbors, an euclidean distance function is used. However the classification time is long and is difficult to find optimal value of **k** [3].

SVM: (Support Vector Machine) is a classifier that constructs hyperplanes in a multidimensional space that separates examples of different class labels. The labels assigned to an unlabeled example depends of the "side" of the hyperplane that the example is found. SVM performs excellent and outstanding results in text classification [9].

MLP: (Multi Layer Perceptron) is a kind of artificial neural network. In this classifier, the artificial neurons (perceptrons) are organized in layers, where the output of each perceptron of a layer creates the input vector of each perceptron of the next layer. Each perceptron performs a dot product with the input vector and a weight vector, exclusive for each perceptron, and use the result of the dot product in a activation function, that determines the output of the perceptron. Mathematically, this classifier implements a nonlinear function that maps from the "representation space" to the "label space" which lets classify data that is not linearly separable (in the "representation space").

Naive Bayes: is a probabilistic classifier based on the Bayes theorem. This classifier is often used in text classification applications and experiments because of its simplicity and effectiveness [10].

Random Forest: is an ensemble of decision trees that tries to improve the accuracy by combining each decision tree classification result. An ensemble is a set of classifiers, that is randomly generated, each classifier in the set uses only a subset of the vector positions to classify the example. The output of the ensemble is a combination of the result of each classifier in the set.

IV. Experiments

A. Datasets

To evaluate the classifiers and the feature engineering techniques 3 datasets are use, each one are detailed below:

- IMDB: proposed by [11] this dataset[1] contains 50000 labeled reviews in which 25000 are positive and 25000 are negative. We use half of them to train (12500 positives and 12500 negatives) and the other half to test.
- Polarity: proposed by [12] this dataset[2] contains 2000 labeled reviews in which 1000 are positive and 1000 are negative. We use 1000 reviews (500 positives and 500 negatives) to train and 1000 reviews to test.
- Movie reviews: proposed by [13] contains 10662 labeled one sentence movie-reviews in which 5331 ate positive and 5331 are negative. We use the split between train and test data used in [14], where 7108 reviews[3] are used for train and 3554 reviews are used for test.

[1] *https://www.kaggle.com/iarunava/imdb-movie-reviews-dataset*
[2] *http://www.cs.cornell.edu/people/pabo/movie-review-data/*

B. Baselines

For each dataset, we execute every feature engineering techniques (BOW, TF-IDF, W2V) and compare each one of them classifying the texts using all classifiers (K-NN, SVM, MLP, Naive Bayes, Random Forest).

C. Metrics

To evaluate and compare each combination of classifier and feature engineering, we used accuracy and F1 score. Both are implemented in *sklearn*[4].

Accuracy is the percentage of correctly classified examples over the total number of examples.

F1 score is the weighted average of precision and recall. Precision is the number of true positives (TP) divided by the number of true positives plus false positives (FP) and recall is the number of true positives divided by the number of true positives plus false negatives (FN). The formula for the F1 score is:

$$F1 = 2 * (precision * recall)/(precision + recall)$$

D. Parameter Settings

For K-NN, SVM and MLP is necessary to determine some parameters. In the following, we describe the values used for these parameters in the experiments.

In the K-NN classifier all the tests were done with $k = 3$.

To choose **C** and **gamma** parameters for SVM classifier we run a grid search for each feature set. Table I shows the best values found for each dataset.

TABLE I
SVM parameters

	IMDB		Polarity		Movie Reviews	
Feature Set	C	gamma	C	gamma	C	gamma
Bag of Words	32	0.0001220703125	0.5	0.03125	8192	0.0078125
TF-IDF	512	0.03125	512	0.03125	8192	0.125
Word Embedding	2	0.125	2	0.125	8192	0.0078125

On a MLP network there is no analytical method for determining the numbers of layers or the number of neurons in the hidden layer. We choose this parameter by trial and error, as suggested in [15].

The MLP topology used is formed by 3 sequential fully connected layers (input, hidden, output) with 200 hidden neurons and a dropout layer with ratio 0.5.

E. Source Code

The source code from our experiments can be found in https://gitlab.c3sl.ufpr.br/mro15/last-info7004.

[3] *https://github.com/mnqu/PTE/tree/master/data/mr*
[4] *https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics*

V. Results

The tables II and III contains the evaluation of each combination of classifier and feature engineering method. The results are the mean of 10 executions and the value of the best execution.

The best average accuracy value for each feature engineering method is highlighted in bold on Table II. The highlight make easier to identify, for each method, which was the best classifier.

In IMDB dataset, SVM were the best classifier, in all feature engineering methods. The best average accuracy found in was 0.8329 using word2vec.

In Polarity dataset, MPL was the best classifier combined with bag of words, on the other hand, when using TF-IDF and word2vec, SVM was the best classifier. The best average accuracy found was 0.73 using SVM with word2vec.

Finally, in Movie Reviews dataset, Random Forest was the best classifiers combined with bag of words and TF-IDF on the other hand SVM was the best classifier combined with word2vec. The best average accuracy found was 0.6322 using SVM with word2vec.

In terms of accuracy, it is possible to note that the best feature engineering method, for the tested datasets, is word2vec, and the best classifier for then is SVM.

TABLE II
Accuracy

IMDB Dataset						
	Bag of Words		TF-IDF		Word Embedding	
Classifier	Average	Best	Average	Best	Average	Best
KNN	0.5749	0.5749	0.6056	0.6056	0.7047	0.7047
SVM	**0.6603**	0.6603	**0.7430**	0.7430	**0.8329**	0.8329
MLP	0.6581	0.6603	0.52	0.526	0.8280	0.8349
NB	0.5594	0.5594	0.6208	0.6208	0.6621	0.6621
RSS	0.5962	0.6030	0.6874	0.6930	0.7132	0.7156
Polarity Dataset						
	Bag of Words		TF-IDF		Word Embedding	
Classifier	Average	Best	Average	Best	Average	Best
KNN	0.5220	0.5220	0.5149	0.515	0.566	0.566
SVM	0.5039	0.504	**0.5199**	0.52	**0.73**	0.73
MLP	**0.53**	0.536	0.5186	0.526	0.675	0.691
NB	0.5179	0.518	0.519	0.519	0.596	0.596
RSS	0.5109	0.536	0.5119	0.5294	0.6091	0.636
Movie Reviews Dataset						
	Bag of Words		TF-IDF		Word Embedding	
Classifier	Average	Best	Average	Best	Average	Best
KNN	0.5019	0.5019	0.5016	0.5016	0.5306	0.5306
SVM	0.5078	0.5078	0.5081	0.5081	**0.6322**	0.6322
MLP	0.5097	0.5112	0.5091	0.5118	0.5817	0.5908
NB	0.5002	0.5002	0.5005	0.5005	0.5644	0.5644
RSS	**0.5109**	0.5135	**0.5124**	0.5149	0.5473	0.5562

In Table III, similar to Table II, the bold values are the best average F1 score for each feature engineering method. In Polarity and mainly in Movie Reviews dataset the values of the F1 score were much lower than accuracy values.

In some cases, the classifier with the best accuracy, for a feature engineering method, does not present the best F1 score. This effect can be observed, for example, in Movie Reviews dataset when bag of words is used, where the best accuracy found was using random forest classifier, but the best F1 score was using MLP.

TABLE III
F1 Score

IMDB Dataset						
	Bag of Words		TF-IDF		Word Embedding	
Classifier	Average	Best	Average	Best	Average	Best
KNN	0.5747	0.5747	0.6041	0.6041	0.7037	0.7037
SVM	**0.6602**	0.6602	**0.7415**	0.7415	**0.8329**	0.8329
MLP	0.6577	0.6603	0.5185	0.5258	0.8274	0.8348
NB	0.5027	0.5027	0.5761	0.5761	0.6601	0.6601
RSS	0.5927	0.5996	0.6874	0.6921	0.7115	0.7138
Polarity Dataset						
	Bag of Words		TF-IDF		Word Embedding	
Classifier	Average	Best	Average	Best	Average	Best
KNN	0.5219	0.5219	0.5065	0.5065	0.5658	0.5658
SVM	0.5004	0.5004	**0.5199**	0.5199	**0.7294**	0.7294
MLP	**0.5294**	0.5349	0.5185	0.5258	0.6717	0.6904
NB	0.4549	0.4549	0.4652	0.4652	0.5940	0.5940
RSS	0.5069	0.5340	0.5077	0.5294	0.6066	0.6326
Movie Reviews Dataset						
	Bag of Words		TF-IDF		Word Embedding	
Classifier	Average	Best	Average	Best	Average	Best
KNN	0.3381	0.3381	0.3380	0.3380	0.5303	0.5303
SVM	0.3711	0.3711	0.3713	0.3713	**0.6318**	0.6318
MLP	**0.4270**	0.4496	0.4105	0.4537	0.5735	0.5905
NB	0.3349	0.3349	0.3345	0.3345	0.5619	0.5619
RSS	0.3981	0.4535	**0.4152**	0.4561	0.5444	0.5534

From Tables II and III, we can see that Movie Reviews dataset is the most difficult of all tested datasets, probably because it contains multiple languages inside it. And we can see that SVM is the best classifier for almost all feature engineering methods, which is aligned with the literature statement, that SVM is good in text classification. Furthermore, bag of words is not a good feature engineering technique, only term frequency does not seems to be enough to represent sentence meaning.

VI. Conclusion

In this work we compared different methods of feature engineering and different classifiers applied to the text classification problem. Using two metrics and three datasets we could evaluate which are the best feature engineering method and the classifiers who works better for these kind of features.

References

[1] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," in *Mining text data*. Springer, 2012, pp. 163–222.

[2] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[3] V. Korde and C. N. Mahender, "Text classification and classifiers: A survey," *International Journal of Artificial Intelligence & Applications*, vol. 3, no. 2, p. 85, 2012.

[4] M. Ikonomakis, S. Kotsiantis, and V. Tampakas, "Text classification using machine learning techniques," *WSEAS transactions on computers*, vol. 4, no. 8, pp. 966–974, 2005.

[5] K. Soumya George and S. Joseph, "Text classification by augmenting bag of words (bow) representation with co-occurrence feature," *IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p-ISSN: 2278-8727Volume*, vol. 16, pp. 34–38, 2014.

[6] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, 2003, pp. 133–142.
[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.
[8] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," *arXiv preprint arXiv:1309.4168*, 2013.
[9] C. Donghui and L. Zhijing, "A new text categorization method based on hmm and svm," in *2010 2nd International Conference on Computer Engineering and Technology*, 2010.
[10] S.-B. Kim, H.-C. Rim, D. Yook, and H.-S. Lim, "Effective methods for improving naive bayes text classifiers," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2002, pp. 414–423.
[11] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015
[12] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2004, p. 271.
[13] ——, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2005, pp. 115–124.
[14] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1165–1174.
[15] U. Orhan, M. Hekim, and M. Ozer, "Eeg signals classification using the k-means clustering and a multilayer perceptron neural network model," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13 475–13 481, 2011.