Comparing Text Vectorization Methods for Sarcasm Detection on Textual Data

Gabriel da Silva Corvino Nogueira
Departamento de Ciência da Computação
Universidade de Brasília
Brasília, Brasil
180113330@aluno.unb.br

***Abstract*—This project aims to make comparisons between three different vectorization methods for textual data. The first method consists of the combination of two classic methods for document vectorization: term frequency–inverse document frequency (TF-IDF) and the use of lexicons. The other two methods involve representing text as a vector of word vectors, with the difference that the second will use locally trained word vectors, while the third will rely on the use of transfer learning. In this way, the efficiency of the methods will be compared through the performance of the classifiers trained with the vectorized data on the sarcasm detection task. For this reason, four classifiers will be used. Three of them are classic machine learning classifiers (Logistic Regression, Naive Bayes, and Support Vector Machine) and one of them is a deep learning based classifier (Convolutional Neural Network). By the end of this report, it is expected to demonstrate the strengths and weaknesses of each approach.**

***Index Terms*—deep learning, supervised learning, natural language processing, text classification,text representation, sentiment analysis**

I. Introduction

With the advent of the information age, the amount of data generated by human beings on networks has been growing exponentially. Sites like Facebook, YouTube, Twitter and Instagram receive thousands of new data from their users daily. In addition, much of this data is stored in textual format and may contain valuable information, such as an individual's opinion on a particular topic or their satisfaction with a certain product. Therefore, sentiment analysis in text has become a very useful feature to describe the opinion of a large group of people on a given subject and, consequently, attractive to large companies.

Text classification is an analytical process that takes any text document as input and assigns a label (or classification) from a predetermined set of class labels [1]. That said, the task of analyzing sentiments in text can be interpreted as a classification task in cases where the goal is to differentiate text polarity, such as in product reviews.

However, in order to be able to perform text classification through mathematical models, it is necessary that the document being classified can be represented in a numerical way as a set of values called features. Such representation can be obtained directly through information present directly in the text, such as word frequency and part-of-speach tagging; by subjective aspects, such as feelings linked to the words of a document; by metadata about the text, like location or number of likes of a tweet; or, in the case of deep leaning, it can be learned by the model in the training process.

This project aims to compare the performance of different numerical representations of texts in the process of detecting sarcasm in tweets. In particular, the study will compare three scenarios: when features are taken directly from text words; when the representation of texts is learned by training word vectors [2]; and when pre-trained word vectors are used. Each representation will be used to train and test classifiers such as Logistic Regression, Naive Bayes, Support Vector Machines and Artificial Neural Networks.

At the end of this analysis, it's expected to be possible to determine whereas the effort of finding features analytically guarantees better classification results than letting the features be learned by a model, being it locally or by transfer learning.

II. Related Work

In the context of textual data representation, members of the Federal University of Minas Gerais (UFMG) sought to explore a new set of features to improve the performance of models for fake news detection [3]. Hence, 141 textual features were evaluated and grouped in sets such as *Language Features* like bag-of-words approaches, n-grams, and POS tagging; *Lexical Features* like number os words, pronouns, verbs, etc; *Psycholinguistic Features* extracted from the LIWC [4] dictionary to capture additional signals of persuasive and biased language; *Semantic Features* that capture the semantic aspect of text by using machine learning models to quantify toxicity in text; and *Subjectivity* by computing the subjectivity and sentiment scores of the text using the TextBlob's API.

In addition to textual features, other features were build based on information about news' source, such as bias, credibility and trustworthiness, and domain location; as well as information about the publishing environment such as engagement and temporal patterns.

Then, several classic classifiers like k-nearest neighbors, Naive Bayes, Random Forest, SVM and Extreme Gradient Boosting (XGD) were used, culminating in the best performance of the Random Forest and XGD algorithms, both with an f1-score of 0.81.

Another work carried out by New York University [5] reported a series of experiments performed with convolutional

neural networks (CNN) trained on top of pre-trained word vectors for sentence classification. Furthermore, it demonstrated that a simple CNN with little hyperparameter tunning and static vector can achieve excellent results when compared with other classifiers.

III. Proposed Method

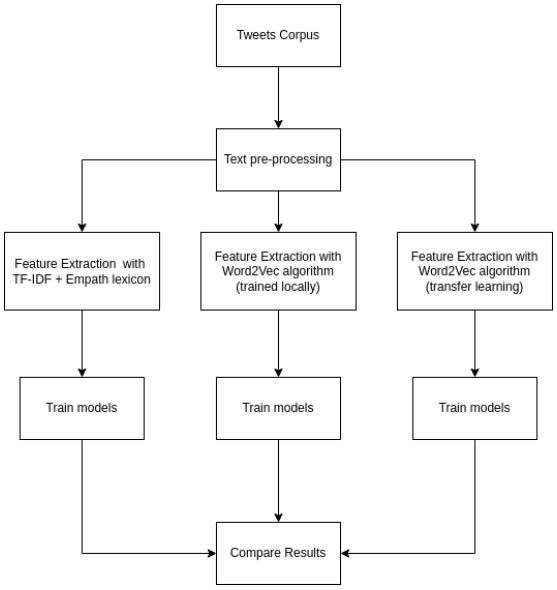

Fig. 1. Flowchart describing the proposed method.

Figure 1 presents the flowchart that illustrates all the steps of the proposed method. Through it, it is possible to perceive that three approaches will be carried out for the numericalization of texts. The first approach involves mixing two traditional methods, TF-IDF and lexicons. On the other hand, the other two approaches involve the use of the Word2Vec algorithm, with the difference that in the second approach, the algorithm will be trained with the corpus used in the experiments and in the third, transfer learning will be used.

Thus, based on each of the strategies, it is planned to train the following models:

- Logistic Regression
- Naive Bayes
- Support Vector Machine
- CNN-static [5]

Finally, the predictions of the trained models will be compared in order to conclude on the effectiveness of the combinations between numericalization methods and models.

A. The data

So that the techniques presented above could be compared, they were applied in the sarcasm detection task. Thus, the **Sarcastic Tweets** [6] corpus was used. Such dataset consists of 104,940 sarcastic and 104,937 non-sarcastic tweets, however, for reasons of processing capacity, it was decided to use only 25% of the dataset, respecting the original data distribution.

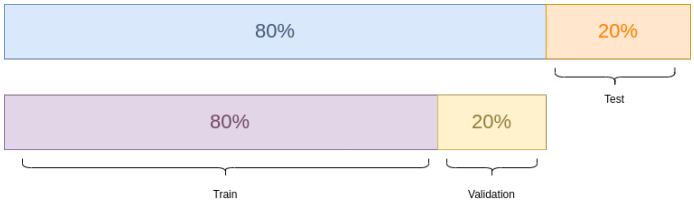

Fig. 2. Train, test and validation division adopted in the experiments.

Therefore, the dataset was divided into training, validation and test sets by adopting the division shown in Figure 2, also preserving data distribution.

B. Text pre-processing

Since working with tweets implies a considerable amount of textual noise, a textual pre-processing phase was put into practice. First, regular expressions were used in order to remove elements such as usernames, web links and retweet flags. Hence, all tweets were tokenized, decapitalized and all stop words were removed. In the end, the tokens obtained were subjected to the stemming process through the Porter Stemming Aglorithm [7].

Nevertheless, it is important to note that, in case of the data presented to the CNN-static model, the process of removing stop words and stemming was not applied, so that the model could take benefit of the word vector representation in the classification context.

C. Feature extraction with TF-IDF and lexicons

The first method used to extract features from tweets was the combination of two methods widely used in classic natural language processing, the term frequency-inverse document frequency (TF-IDF) and lexicons.

TF-IDF is famous method for text vectorization, it represents a text document as a vector in the $\mathbb{R}^n$ space, where n represents the vocabulary size. The difference between this technique and the traditional bag-of-words approach is that the TF-IDF measure evaluates how relevant a word is to a document in a collection of documents.

Lexicon is the component of a NLP system that contains information (semantic, grammatical) about individual words or strings [8]. A lexicon will turn a text document into a vector in the $\mathbb{R}^c$ space, where c is the number of categories supported by the lexicon. Hence, each feature of this vector will represent the count of words that belong to a predefined category.

Accordingly, the final representation of a tweet was made by concatenating its representations generated by the aforementioned methods, which resulted in vector in $\mathbb{R}^{n+c}$ space.

D. Feature extraction with the Word2Vec algorithm

Just like texts, words can also be represented by means of a vector (word vector). The Word2Vec Algorithm is a set of techniques that can be used for learning high quality word vectors from huge data sets whit billions of words and with milions of words in the vocabulary [2].

Therefore, a document such as a tweet can be represented as a vector of word vectors generated by Word2Vec. Such representation is addressed in the last two methods shown in Figure 1, with the contrast that, in the first case, the word vectors will be obtained by training the Word2Vec model from the Sarcastic Tweets corpus. Whereas in the second case, pre-trained vectors, trained on the Google News corpus, will be used.

E. Model training

In order to compare the vectorization methods, it was proposed that three classical classifiers and a convolutional neural network were used for the sarcasm detection task.

As the task addressed is a binary classification problem, the Logistic Regression, Naive Bayes and Support Vector Machine (SVM) classifiers were chosen. Since such models deal with input vectors that have numerical values as features, it is possible to provide them with the vectorized data obtained through the process described in Section III-C.

Conversely, for these models to be able to receive the vectorized tweets though the strategy presented in Section III-D, it is necessary that the vectors of word vectors that represent the documents are flattened so that the document in represented by a single vector composed only of real values. Consequently, the resulting vectors end up having a large dimensionality value. Therefore, in order not to exceed storage and processing limits, it is necessary to reduce the dimensionality of such vector representation. For this purpose, the Principal Component Analysis algorithm (PCA) can be used.

In addition to the classic models, the CNN-static convolutional neural network suggested in the "Convolutional neural networks for sentence classification"article [5] was implemented to detect sarcasm in tweets. Figure 3 presents the proposed network architecture. Through it, it is possible to perceive that a document is treated as a matrix composed of word vectors. Since, the convolution operation is made in a window of h words (filter width). Such operation can be done with multiple values of h through parallel convolutional layers. Right after the convolution phase, a max-pooling layer is applied and finally the resulting values are passed to a fully connected layer with dropout and softmax output.

For this reason, this model can be easily applied to vectorized tweets using the methods presented in Section III-D. However, the model becomes unfeasible to be applied to vectorized data using the method presented in section III-C, since the vectors generated by the document are very sparse and represent the words in a numerical and not vectorial way, which would generate a great loss of information if a reshaping operation was done.

F. Comparing model results

After applying the vectorization techniques presented and training the models, the effectiveness of each technique can be measured by the models performance. Therefore, precision, recall and F1 score metrics must be taken into account. In addition, the confusion matrix for each of the trained classifiers must be presented.

IV. Experimental Results

TABLE I
Dataset Split

	Sarcastic Tweets	**Non-sarcastic Tweets**	**Total**
Train Set	16,729	16,851	33,580
Validation Set	4,243	4,152	8,395
Test Set	5,263	5,231	10,494
Total	26,235	26,234	52,469

First, the data were divided into training, validation and test sets as shown in Table I. Then, after carrying out the text pre-processing stage, the vectorization method described in Section III-C was applied to the documents. To this end, the Empath [9] lexicon, a free alternative to LIWC, was used. Empath assigns 194 class by default to a document, so for each text present in the dataset, a vector $v_1 \in \mathbb{R}^{194}$ was generated. Afterward, the vectorization process through TF-IDF was applied to the dataset, ignoring tokens with frequency lower than 2, which resulted in vectors $v_2 \in \mathbb{R}^{11959}$. Finally, vectors of type v_1 were concatenated with vectors of type $v_2 \in 2$ generating the final representation of the documents through vectors $v_3 \in \mathbb{R}^{12153}$

Subsequently, the models were trained with the vectorized data, using the randomized search process to optimize hyper-parameters through the validation set. Figure 4 presents the confusion matrices obtained by the classifiers in the test set.

Furthermore, the vectorization method described in section III-D was applied by employing word vectors trained in the Sarcastic Tweets corpus. Thus, for each word, a vector of 100 dimensions was generated. Therefore, so that the matrices representing the tweets could have the same size, a padding process was applied, limiting the data provided to the classical models to 100 words and the CNN data to 344 words, totally reasonable numbers, since a tweet can have a maximum of 280 characters.

Even so, the matrices provided to the classical classifiers undergo the process of flattening, which ends up generating vectors of large dimensions. Therefore, a dimensionality reduction was made through the PCA algorithm, resulting in vectors with 1000 numerical components and a sum of variance ratios of 0.986, which implies that much of the data continued to be represented by this new vector space.

Next, the classical classifiers were trained, also using the validation set to optimize hyperparameters via randomized search. However, this process proved to be much more costly in terms of time, specifically with regard to the SVM model, which limited the procedure.

Regarding the CNN model, it is important to emphasize that the hyperparameters used in the model originally proposed were adopted. Thus, the network is formed by 3 parallel convolutional layers, with filter width of sizes 3, 4, 5 respectively and a dropout layer with a coefficient of 0.5. In addition, adam optimizer was used with a learning rate of 10^{-3}. Finally, the model was trained for 10 epochs, using the early stopping criterion with the patience coefficient equal to 2.

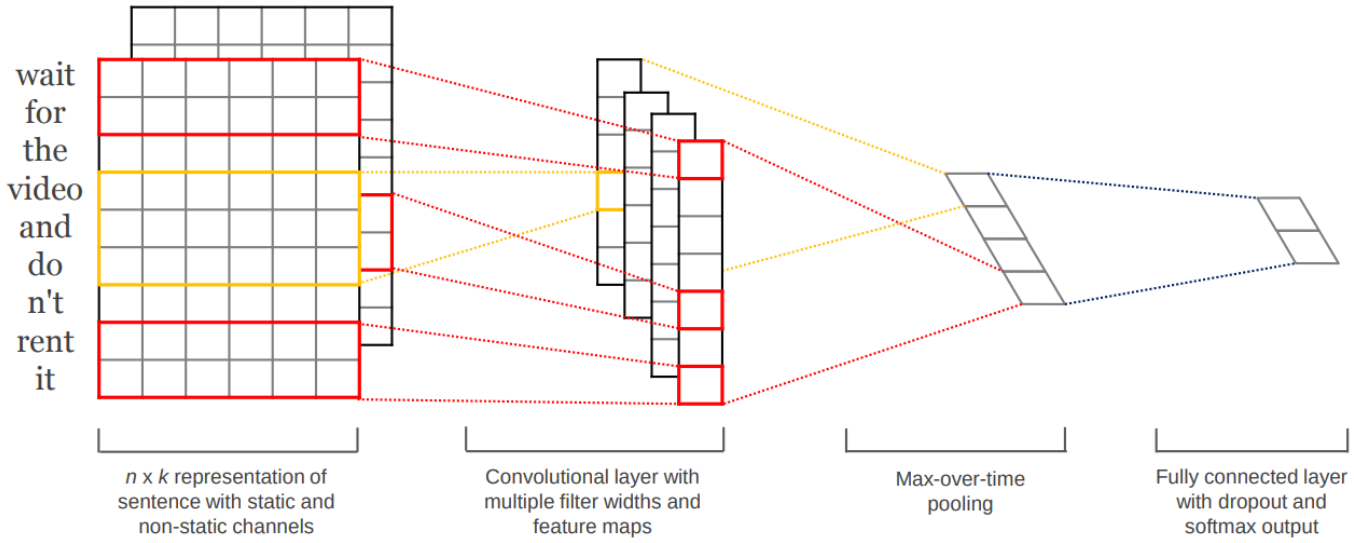

Fig. 3. Model architecture with two channels for an example sentence (image taken from the "Convolutional neural networks for sentence classification" article).

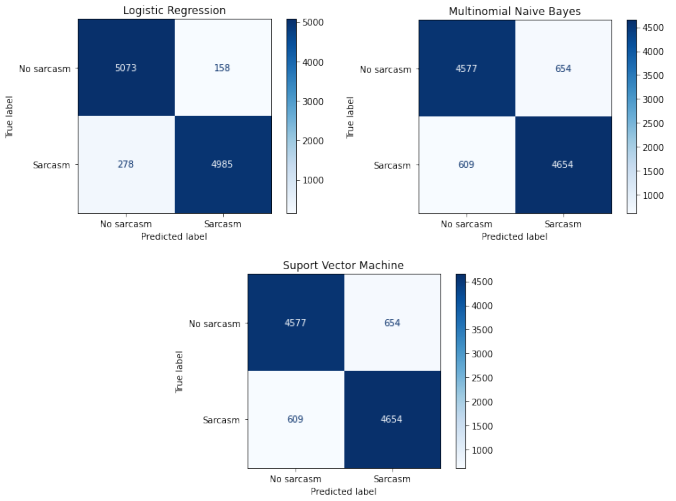

Fig. 4. Confusion matrix of classifiers that use the text vectorization method described in Section III-C.

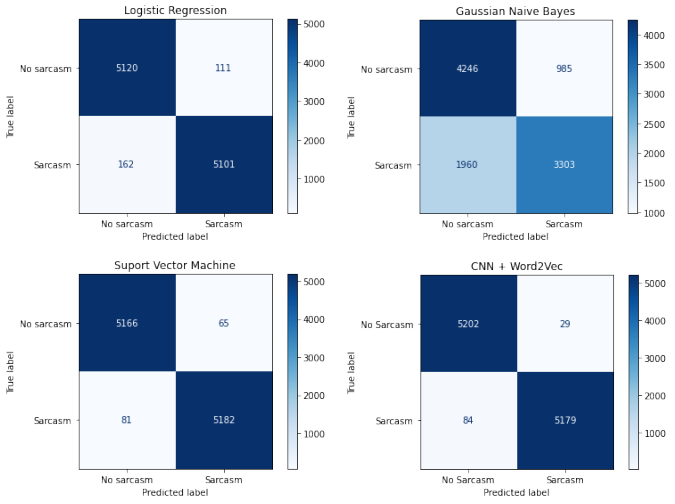

Fig. 5. Confusion matrix of classifiers that use the text vectorization method described in Section III-D without pre-trained vectors.

Since, Figure 5 presents the confusion matrices of the classifiers trained with data that comes from the vectorization process based on word vectors trained with tweets present in the corpus.

Last of all, the text vectorization process via Word2Vec was repeated, however using pre-trained word vectors, trained with the Google News corpus. Therefore, each word vector has 300 dimensions. Due to memory limits, it was necessary to restrict the classic classifiers input size to the maximum of 35 word vectors. Thus, dimensionality reduction was applied, resulting in vectors with 2000 numerical components and a sum of variance ratios of 0.918. In addition, an attempt was made to optimize hyperparameters for non-deep learning classifiers. However, the process was demanding in computational resources, which made its application unfeasible for the SVM classifier.

Hence, Figure 5 presents the confusion matrices of the classifiers trained with data that comes from the vectorization process based on word vectors trained with Google News corpus.

Ultimately, the Table II, presents a summary of the results obtained. Therefore, it is possible to see that the models based on convolutional neural networks achieved the best results, together with the SVM classifier trained with data obtained through word vectors without transfer learning. However, the first, third and fourth models in the table require a considerably

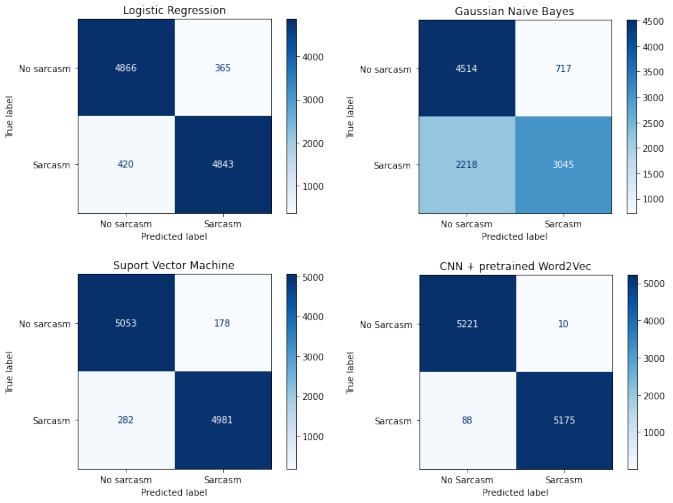

Fig. 6. Confusion matrix of classifiers that use the text vectorization method described in Section III-D pre-trained vectors from the Google News corpus.

lower computational cost in the training process and presented competitive results when compared with the best ones.

TABLE II
CLASSIFICATION RESULTS

Models	Metrics (macro average)			
	Precision	*Recall*	*F1 score*	*Accuracy*
LR + TF-IDF + Lexicon	0.96	0.96	0.96	0.96
NB + TF-IDF + Lexicon	0.88	0.88	0.88	0.88
SVM + TF-IDF + Lexicon	0.96	0.96	0.96	0.96
LR + Word2Vec	0.97	0.97	0.97	0.97
NB + Word2Vec	0.72	0.73	0.72	0.72
SVM + Word2Vec	0.99	0.99	0.99	0.99
CNN + Word2Vec	0.99	0.99	0.99	0.99
LR + Pre-trained Word2Vec	0.93	0.93	0.93	0.93
NB + Pre-trained Word2Vec	0.72	0.74	0.71	0.72
SVM + Pre-trained Word2Vec	0.96	0.96	0.96	0.96
CNN + Pre-trained Word2Vec	0.99	0.99	0.99	0.99

V. CONCLUSION

The method proposed by this work aims to compare different techniques for text vectorization, as well as their influence in the classification process. Thus, it became evident that models based on Deep Learning are capable of offering excellent results. However, its training process requires much more processing power than other models. On the other hand, as highlighted in the previous section, the models that used TF-IDF and the Empath lexicon were much more efficient than other models in the training process, although their score in the evaluation metrics was relatively lower. In addition, it was possible to notice that models that used word vectors trained in the Sarcastic Tweets corpus presented better results than models that used transfer learning. This indicates that training these vectors in a specific vocabulary is able to express their aspects more accurately. Finally, it is noticeable that the Naive Bayes classier had a decrease in its performance when using word vectors to represent text.

As a result, it is possible to conclude that the use of Word Vectors to represent text documents provides a good performance for models based on neural networks, however, in most cases, it does not have a great advantage in being combined with classic machine learning models, as it results in a small variation of the results at the cost of a very high processing time.

Eventually, due to the limitations for hyperparameter optimization in carrying out this work, it is interesting that these experiments are performed again with the appropriate hardware, so that more solid conclusions can be drawn. Furthermore, it is interesting to develop a way to use the convolutional neural network proposed with vectors generated by the method presented in section III-C.

REFERENCES

[1] "Chapter 17 - summary," in *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications*, G. Miner, D. Delen, J. Elder, A. Fast, T. Hill, and R. A. Nisbet, Eds. Boston: Academic Press, 2012, pp. 1007–1016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123869791000451

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[3] J. C. S. Reis, A. Correia, F. Murai, A. Veloso, F. Benevenuto, and E. Cambria, "Supervised learning for fake news detection," *IEEE intelligent systems*, vol. 34, no. 2, pp. 76–81, 2019.

[4] J. Pennebaker, M. Francis, and R. Booth, "Linguistic inquiry and word count (liwc)," 01 1999.

[5] Y. Kim, "Convolutional neural networks for sentence classification," 2014.

[6] W. Webb, "Sarcastic tweets." [Online]. Available: https://data.world/williamwebb35/sarcastictweets

[7] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[8] L. Guthrie, J. Pustejovsky, Y. Wilks, and B. M. Slator, "The role of lexicons in natural language processing," *Commun. ACM*, vol. 39, no. 1, p. 63–72, jan 1996. [Online]. Available: https://doi.org/10.1145/234173.234204

[9] E. Fast, B. Chen, and M. S. Bernstein, "Empath," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, may 2016. [Online]. Available: