Analysis of the impact of parameters in TextGCN

Henrique Varella Ehrenfried
hvehrenfried@inf.ufpr.br
Federal University of Paraná
Curitiba, Paraná, Brazil

Eduardo Todt
todt@inf.ufpr.br
Federal University of Paraná
Curitiba, Paraná, Brazil

ABSTRACT

Deep learning models uses many parameters to work properly. As they become more complex, the authors of these novel models cannot explore in their papers the variation of each parameter of their model. Therefore, this work describes an analysis of the impact of four different parameters (Early Stopping, Learning Rate, Dropout, and Hidden 1) in the TextGCN Model. This evaluation used four datasets considered in the original TextGCN publication, obtaining as a side-effect small improvements in the results of three of them. The most relevant conclusion is that these parameters influence the convergence and accuracy, although they individually do not constitute strong support when aiming to improve the model's results reported as the state-of-the-art.

KEYWORDS

neural networks, geometric deep learning, textgcn, parameter analysis

1 INTRODUCTION

The increase in the availability of the internet made possible the use of smart mobile devices. Therefore, people started to produce much data, as observed by Kemp [1]. Many of these data are in the text format, which may be difficult to classify. Text classification is an important task, so people can make filters to avoid offensive content. Those filters can also help authorities to identify cybercrimes and cyberbully.

As many papers present new models and methodologies to classify text, they do not show how each parameter affects their creation. Therefore, researchers willing to use these new models and methodologies must invest some time to discover all the potential of it.

In this paper, TextGCN, a model to classify texts, is explored. In doing so, the authors try to identify how four parameters, from a list of seven parameters, work. The parameters explored are Early stopping, Learning rate, Hidden 1 and Dropout. More details are presented in Section 4.

The goal of this paper is to guide future works that use TextGCN to achieve its best performance. As a result of this guide, researchers can see the limitations of the explored model.

The division of this work is the following: Section 2 introduces the related work. Section 3 presents a brief explanation of the GCN and the TextGCN models. Section 4 explains the experimentation done. Finally, Section 5 finishes this paper with the authors' final considerations.

2 RELATED WORK

Moraes and colleagues [2] compared many proposals made to classify text. Some of these proposals are the Support Vector Machines (SVM), Naive Bayes (NB) and Artificial Neural Networks (ANN). In their studies, Moraes and colleagues [2] identified that ANN performance is auspicious. However, there are other methods which are much better at training time. Their experiments used the bag-of-words (unigrams) approach.

Other initiatives on classifying text tried the Convolutional Neural Network (CNN) and a Long Short-Term Memory (LSTM) models to compute continuous representations of sentences. In Tang and colleagues' work [3], the input of a Gated Recurrent Neural Network (GRNN) is the output of these models (CNN and LSTM). A softmax classifier uses the output of the GRNN. The results achieved are promising since they are better than the other methods compared (the other methods are mainly variations of SVM).

Authors are exploring the perspective of word embedding. An example is a work from Yu and colleagues [4], which suggests that refining the word embedding can improve the results of the machine learning models that already exist (CNN, Bi-LSTM, and others).

Young and colleagues [5] made a comprehensive review of the natural language processing (NLP) field. In their work, they explain the essential concepts of the area. They also present the recent progress of the NLP area.

Most recent works started to use Geometric Deep Learning (GDL). One of these works was made by Zhang and colleagues [6], in which a Graph Convolutional Network (GCN) was used to classify sentiments using the syntactic tree of the sentences. Compared to the literature, their results are near the state-of-the-art, or they are state-of-the-art.

Tuning parameters, as this paper tries to do, is not new. Koutsoukas and colleagues[7] tested different methods with different parameters to classify different types of bioactivity. Their results suggest that a substantial effect on a Deep Neural Network is achieved by tuning parameters.

3 MODELS USED

To classify text using geometrical deep learning, the GCN model [8] and the TextGCN [9] model stand up. They define a convolutional model that receives as input a graph. The difference between them is that the TextGCN needs a specific type of graph, while the GCN receives any graph. It is important to note that the TextGCN uses the GCN model. Subsections 3.1 and 3.2 explain about these models.

3.1 Graph Convolution Network - GCN

Kipf and Welling are the authors of the GCN [8]. They defined a convolutional model that can process graphs. Basically, in the GCN, the model induces embedding vectors of nodes based on the neighbors of each node.

To make the model work, it receives as input a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. It is also assumed that $|V| = n$.

In this graph G, all nodes have at least one edge. This edge is an edge to itself. Therefore $(v, v) \in E$ for any $v \in V$. Consider there is a matrix $X \in \mathbb{R}^{n \times m}$ as a matrix containing all n nodes with their features, where m is the dimension of the feature vectors and n the number of vertices.

Let A be an adjacency matrix of G and D as the degree matrix of G, where $D_{i,i} = \sum_j A_{ij}$. Notice that the diagonal elements of A are set 1 because of self-loops.

The first layer of this model can capture information about the immediate neighbors. If more layers are defined, the model can work with the information of farther nodes. There is an illustration of the processing of this model in a graph in Figure 1.

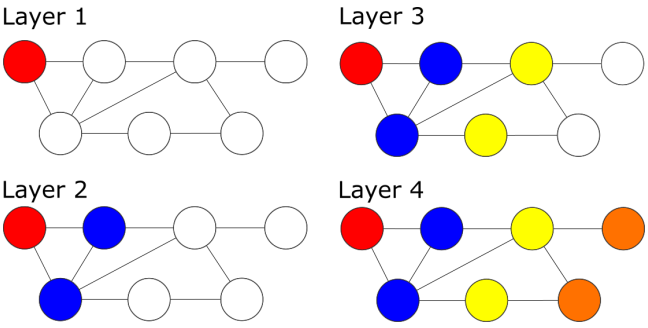

Figure 1: GCN Layers. From the node's perspective in the most left top corner (red also known as node 1), in the first layer, it only has information about itself. In the second layer, it retrieves information about the neighbors (blue nodes [nodes 2 and 5]), and so on.

To compute the k-dimensional node feature matrix $L^{(1)} \in \mathbb{R}^{n \times k}$ use Equation 1

$$L^{(1)} = \rho(\tilde{A}XW_0) \tag{1}$$

In Equation 1, the $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix, and $W_0 \in \mathbb{R}^{m \times k}$ is a weight matrix. ρ is the activation function, the default implementation of the GCN uses it as being a ReLU function $\rho(x) = max(0, x)$.

A generalized way to define the GCN for multiple layers is:

$$\begin{cases} L^{(0)} = X \\ L^{(j+1)} = \rho(\tilde{A}L^{(j)}W_j) \end{cases} \tag{2}$$

In Equation 2, j denotes the layer number.

3.2 Text Graph Convolution Network - TextGCN

Using GCN as a base, Yao and colleagues created the TextGCN [9]. This model uses the structure of the GCN explained previously. The difference between the GCN and the TextGCN is the input graph. Where the GCN accepts any graph, and the TextGCN uses as input a specific type of graph.

The TextGCN graph is composed of documents and unique words. Each document is a file that may contain many words. Therefore, the number of vertices of the graph $|V|$ is the number of unique words and documents. In order to create the edges, it is linked word to word and document to word using the following idea:

1) Add an edge between a word and a document if that word occurs in the document; 2) Add an edge between two words based on the co-occurrence of the words in the whole corpus; 3) To fulfill the GCN's constraints, add an edge between the node and itself (self-loop).

Each edge of this graph must have a weight. The weight is calculated depending on the type of nodes the edge links, as can be seen in Equation 3

$$A_{ij} = \begin{cases} \text{PMI}(i, j) & i, j \text{ are words} \\ \text{TF-IDF}_{ij} & i \text{ is a document and } j \text{ is a word} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The PMI, from Equation 3, can be calculated using Equations 4, 5, and 6. Notice that the $\#W(i)$ is the number of sliding windows in a corpus that contain the word i; $\#W(i, j)$ is the number of sliding windows that contain both the word i and j; and $\#W$ is the total number of sliding windows in the corpus.

$$\text{PMI}(i, j) = log\frac{p(i, j)}{p(i)p(j)} \tag{4}$$

$$p(i, j) = \frac{\#W(i, j)}{\#W} \tag{5}$$

$$p(i) = \frac{\#W(i)}{\#W} \tag{6}$$

It is possible to calculate the TF-IDF using Equation 7

$$\text{TF-IDF}_{ij} = tf_{j,i} \times log\left(\frac{N}{df_j}\right) \tag{7}$$

In Equation 7, the $tf_{j,i}$ is the frequency of the word j in document i. df_j is the number of documents that contain the word j. Finally, N is the number of documents. Furthermore, Figure 2 shows an example of the TextGCN input graph.

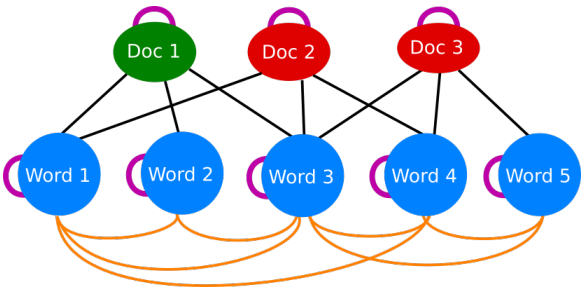

Figure 2: TextGCN example Graph. Notice the omission of the weights that aims to improve readability. Doc 1 (green) is a positive document node, Doc 2 and Doc 3(red) are negative document nodes. Word {1,2,3,4,5} (blue) nodes are word nodes

.

Use the created graph in the GCN model. After it passes the last GCN layer, the algorithm must run a *softmax* classifier: $Z = softmax(\text{last GCN Layer})$. If the last GCN layer is the second layer, then $Z = softmax(\tilde{A}\,\text{ReLU}(\tilde{A}XW_0)W_1)$.

The *softmax* classifier can be defined as $softmax(x_i) = \frac{1}{\mathcal{Z}} \exp(x_i)$ with $\mathcal{Z} = \sum_i \exp(x_i)$. The definition of the loss function is the cross-entropy error over all labeled documents:

$$\mathcal{L} = - \sum_{d \in \mathcal{Y}_D} \sum_{f=1}^{F} Y_{df} \ln Z_{df} \quad (8)$$

In Equation 8, the $\mathcal{Y}_D$ is the set of documents indices that have labels. F is the dimension of the output features, which is equal to the number of classes. Y is the label indicator matrix. The gradient descent can train the weight parameters.

4 EXPERIMENT

This paper investigates how the parameters: Early Stopping, Learning Rate, Dropout, and Hidden 1 work in the TextGCN model.

4.1 Experiment preparation

MR, Ohsumed, R8, and R52 are the datasets used to experiment with each parameter's impact. All evaluation happened with the code developed by Yao and colleagues [9], which is accessible in GitHub[1]. This work adopts the same methodology used by Yao and colleagues [9]. All datasets collected are from the work of Yao and colleagues [9]. Thus the results can be compared.

They [9] describe the dataset as follows.

The MR dataset is a movie review dataset for binary sentiment classification, in which each review only contains one sentence [10][2]. The corpus has 5,331 positive and 5,331 negative reviews. We used the training/test split in [11][3].

The Ohsumed corpus[4] is from the MEDLINE database, a bibliographic database of crucial medical literature maintained by the National Library of Medicine. In this work, we used the 13,929 unique cardiovascular diseases abstracts in the first 20,000 abstracts of the year 1991. Each document in the set has one or more associated categories from the 23 disease categories. As we focus on single-label text classification, the documents belonging to multiple categories are excluded so that 7,400 documents belonging to only one category remain. 3,357 documents are in the training set, and 4,043 documents are in the test set.

R52 and R8 [5] (all-terms version) are two subsets of the Reuters 21578 dataset. R8 has eight categories split into 5,485 training and 2,189 test documents. At the same time, R52 has 52 categories and split into 6,532 training and 2,568 test documents. Details about the descriptive statistics of the dataset are in Table 1.

4.1.1 Parameter explanation. The "Learning Rate" parameter controls how much to change the model according to the estimated error. If the value of the learning rate is small, it will take longer to train a model. If it is large, the result calculated as the optimal could be the non-optimal result.

[1]Original code available at https://github.com/yao8839836/text_gcn - Accessed on October, 27th 2020
[2]http://www.cs.cornell.edu/people/pabo/movie-review-data/ - Accessed on October, 27th 2020
[3]https://github.com/mnqu/PTE/tree/master/data/mr - Accessed on October, 27th 2020
[4]http://disi.unitn.it/moschitti/corpora.htm - Accessed on October, 27th 2020
[5]https://www.cs.umb.edu/~smimarog/textmining/datasets/ - Accessed on October, 27th 2020

Table 1: Summary statistics of datasets.

	MR	Ohsumed	R8	R52
# Docs	10,662	7,400	7,674	9,100
# Training	7,108	3,357	5,485	6,532
# Test	3,554	4,043	2,189	2,568
# Words	18,764	14,157	7,688	8,892
# Nodes	29,426	21,557	15,362	17,992
# Classes	2	23	8	52
Average Length	20.39	135.82	65.72	69.82

"Epochs" can be seen as iterations where the dataset passes through the neural network. Each epoch corresponds to the dataset been passed forward and backward to the neural network completely.

"Hidden 1" is the parameter to set the number of neurons in the layer N. In this case, the number of neurons in layer 1. This layer is "hidden" because it is not the input layer nor the output layer.

"Dropout" is a technique to regularize the network. It reduces the model's overfitting by "turning off" some neurons randomly.

"Weight Decay" is another technique to reduce overfitting. It consists of a number that penalizes the complexity of the model in training.

"Early Stopping" is a technique to avoid the model to be trained more than needed. When training a model more than needed, the loss function increases and makes it harder to find the optimal solution.

The default configuration is similar to the configuration defined by Yao and colleagues [9], which is in Table 2. The difference is that the number of epochs is 10000 instead of 200.

Table 2: The default configuration of the GCN defined by the authors of TextGCN [9]

Parameter	Description	Default Value
Model	Model String	gcn
Learning Rate	Initial learning rate	0.02
Epochs	Number of epochs to train	200
Hidden 1	Number of units in hidden layer 1	200
Dropout	Dropout rate (1 - keep probability)	0.5
Weight Decay	Weight for L2 loss on embedding matrix	0
Early Stopping	Tolerance for early stopping (# of epochs)	10

4.1.2 Experiment methodology. This paper studied the parameters: Early Stopping, Learning Rate, Dropout, and Hidden 1. Therefore, the other parameters are their default values (GCN for the Model and 0 for the Weight Decay). The maximum number of epochs is 10000 instead of the original 200 proposed by [9]. Notice that it can run fewer epochs if the early stop trigger. Therefore there is no guarantee that the 10000 will run. Table 3 shows the parameters used in each experiment run. If another parameter displayed in Table

2 is not in Table 3, all experiments will use the default value. The only exception is the number of epochs that is 10000 instead of 200.

The experiment consists of a variation of a parameter to evaluate its impact on the results. Experiment 0 is the default configuration proposed by Yao and colleagues [9]. Consequently, it is the baseline.

Each experiment was executed ten times. There was no change in the loss function in these Experiments neither in the number of hidden layers, which was the same used and described originally by Yao and colleagues [9].

The experiment consisted of 19 configurations. All data were collected and analyzed using a set of scripts made available at a GitHub repository. In this repository, there are the scripts and the results achieved.[6].

Table 3: Experiment configurations used. Each line is a setup of a experiment that was executed 10 times for each dataset.

Name	Learning Rate	Hidden 1	Dropout	Early Stopping	Evaluate
ES10	0.02	200	0.5	10	Early Stopping
ES100	0.02	200	0.5	100	
ES1000	0.02	200	0.5	1000	
LR01	0.1	200	0.5	10	Learning Rate
LR001	0.01	200	0.5	10	
LR0001	0.001	200	0.5	10	
LR00001	0.0001	200	0.5	10	
DO01	0.02	200	0.1	10	Dropout
DO02	0.02	200	0.2	10	
DO03	0.02	200	0.3	10	
DO04	0.02	200	0.4	10	
DO05	0.02	200	0.5	10	
DO06	0.02	200	0.6	10	
DO07	0.02	200	0.7	10	
DO08	0.02	200	0.8	10	
DO09	0.02	200	0.9	10	
HD50	0.02	50	0.5	10	Hidden 1
HD350	0.02	350	0.5	10	
HD500	0.02	500	0.5	10	

4.2 Experiment results

After executing all experiments, a script available in the GitHub repository[6] processed them. A T-test for means of two independent samples from descriptive statistics[7] is the statistical test used to compare if the means were equal (Null hypothesis) or not (Alternative hypothesis). This script compares the results of each experiment and the results from the paper [9]. In the paper, due to the page limit, just the best result of each dataset is discussed.

4.2.1 Reproducing the original experiment. Initially, a comparison between the results from the paper [9] and ES10 must exist to evaluate if the results from [9] are reproducible since they use the same parameters.

Table 4 compares the original experimental results made by [9] and the experiments run by the authors. It is possible to notice that the T-test, at the alpha value of 0.05, fails to reject the null hypothesis that the means are equal in three datasets (Ohsumed, R8, and R52).

[6] https://github.com/HenriqueVarellaEhrenfried/AnalysisResultsTextGCN - All results and scripts to analyze them - Accessed October 30th 2020

[7] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind_from_stats.html#scipy.stats.ttest_ind_from_stats - Documentation of the T-Test used - Accessed October 27th 2020.

Table 4: Results achieved vs results from the original paper ([9]). Exp stands for Experiment. Orig stands for Original, which means the data from the paper [9]. The number after the "±" sign is the standard deviation. The P-Value column refer to the T-test for means of two independent samples from descriptive statistics. To know the parameters used in this experiment, please refer to the Table 3.

Dataset	MR	Ohsumed	R8	R52
Exp name	ES10	ES10	ES10	ES10
Exp Mean	0.7593 ± 0.0008	0.6869 ± 0.0033	0.9716 ± 0.0015	0.9366 ± 0.0019
Orig Mean	0.7674 ± 0.0020	0.6836 ± 0.0056	0.9707 ± 0.0010	0.9356 ± 0.0018
P-Value	5.546869e-08	0.1204304282	0.1147659345	0.2280961891
Confidence Level	99.9999%	87.9570%	88.5234%	77.1904%
Higher Mean	Original	Experiment	Experiment	Experiment
Diff (Exp-Orig)	-0.0081	0.0033	0.0009	0.0010

This result is the expected behavior since the same settings applied in both experiments (original and this paper's experiment) are the same. Therefore, this paper's authors expected the same results since an assumption is that the experiments conducted by [9] are reproducible.

The unexpected is the result of the MR Dataset. It is inferior to the original result, and the null hypothesis of the T-test can be rejected with a confidence level of 99.9999%. Therefore, we were unable to reproduce the experiment with the MR dataset. Even though, we proceeded as planned and executed all experiments with the MR dataset.

4.2.2 All results. After investigating the reproducibility of the results, the other experiments could go on. The best results were compared to the original paper result. In this comparison, the results achieved were statistically tested if they are the same or not, as described previously. The results of the statistical tests are in Table 5.

All the results are in Table 6, which spotlights the best results of each dataset. Additionally, the chart in Figure 3 shows the results of each experiment graphically.

Table 5: Best results. Exp stands for Experiment. Orig stands for Original, which means the data from the paper [9]. The number after the "±" sign is the standard deviation. The P-Value column refers to the T-test for means of two independent samples from descriptive statistics. To know the parameters used in this experiment, please refer to the Table 3.

Dataset	MR	Ohsumed	R8	R52
Exp Name	DO08	DO01	LR001	ES1000
Exp Mean	0.7612 ± 0.0009	0.6886 ± 0.0018	0.9736 ± 0.0007	0.9383 ± 0.0013
Orig Mean	0.7674 ± 0.0020	0.6836 ± 0.0056	0.9707 ± 0.0010	0.9356 ± 0.0018
P-value	8.299081e-07	2.131075e-02	6.360535e-07	1.153807e-03
Confidence Level	99.9999%	97.8689%	99.9999%	99.8846%
Higher Mean	Original	Experiment	Experiment	Experiment
Diff (Exp-Orig)	-0.006173	0.004998	0.002940	0.002678

It is possible to notice that each dataset's results tend to be numerically similar, with some exceptions, as shown by the chart in Figure 3. An observation of this chart reveals that the increase in the early stop makes the average decrease considerably (A delta of 0.0483)

Table 6: All results of the experiments. The best results are highlighted in dark blue. The number before the sign ± is mean, and after it is the standard deviation. Each experiment runs ten before its descriptive statistics were measured.

Name	MR		Ohsumed		R8		R52	
	Accuracy	Epochs	Accuracy	Epochs	Accuracy	Epochs	Accuracy	Epochs
ES10	0.7593 ± 0.0008	18.0 ± 0.0000	0.6869 ± 0.0033	78.4 ± 1.2000	0.9716 ± 0.0015	92.6 ± 4.3174	0.9366 ± 0.0019	131.9 ± 10.1139
ES100	0.7235 ± 0.0011	102.0 ± 0.0000	0.6862 ± 0.0022	126.6 ± 0.6633	0.9694 ± 0.0010	144.7 ± 2.6476	0.9373 ± 0.0013	204.2 ± 6.1935
ES1000	0.7110 ± 0.0011	1002.0 ± 0.0000	0.6713 ± 0.0014	1002.0 ± 0.0000	0.9662 ± 0.0011	1002.0 ± 0.0000	**0.9383 ± 0.0013**	1012.4 ± 2.0100
LR01	0.7523 ± 0.0028	12.0 ± 0.0000	0.6782 ± 0.0045	28.4 ± 0.6633	0.9650 ± 0.0016	32.3 ± 2.0025	0.9294 ± 0.0032	49.3 ± 2.1932
LR001	0.7592 ± 0.0011	29.1 ± 0.3000	0.6864 ± 0.0032	127.4 ± 2.7276	**0.9736 ± 0.0007**	145.5 ± 7.5000	0.9343 ± 0.0022	215.9 ± 15.5271
LR0001	0.7602 ± 0.0010	191.3 ± 0.4583	0.6766 ± 0.0017	1014.4 ± 25.4016	0.9712 ± 0.0007	1198.2 ± 36.7500	0.9234 ± 0.0039	1721.0 ± 132.7275
LR00001	0.7606 ± 0.0008	1803.2 ± 4.8332	0.6710 ± 0.0025	9268.1 ± 234.2428	0.9709 ± 0.0005	9006.5 ± 185.9275	0.8935 ± 0.0015	10000.0 ± 0.0000
DO01	0.7565 ± 0.0012	17.9 ± 0.3000	**0.6886 ± 0.0018**	73.9 ± 0.9434	0.9725 ± 0.0010	87.3 ± 2.9000	0.9380 ± 0.0009	130.4 ± 5.5172
DO02	0.7573 ± 0.0007	18.0 ± 0.0000	0.6879 ± 0.0019	74.5 ± 1.2845	0.9721 ± 0.0008	88.9 ± 2.3854	0.9373 ± 0.0016	131.5 ± 8.2614
DO03	0.7581 ± 0.0007	18.0 ± 0.0000	0.6884 ± 0.0015	76.1 ± 0.9434	0.9718 ± 0.0011	90.2 ± 3.0919	0.9368 ± 0.0012	131.1 ± 6.8037
DO04	0.7586 ± 0.0009	18.0 ± 0.0000	0.6874 ± 0.0012	76.8 ± 1.4697	0.9725 ± 0.0010	90.1 ± 4.3232	0.9371 ± 0.0018	133.9 ± 6.3000
DO05	0.7590 ± 0.0009	18.1 ± 0.3000	0.6866 ± 0.0016	78.2 ± 1.3266	0.9727 ± 0.0014	91.6 ± 3.4409	0.9372 ± 0.0019	132.6 ± 10.9654
DO06	0.7595 ± 0.0006	18.9 ± 0.3000	0.6853 ± 0.0019	79.8 ± 1.2490	0.9730 ± 0.0009	88.7 ± 4.1243	0.9342 ± 0.0019	131.4 ± 9.6768
DO07	0.7603 ± 0.0015	19.0 ± 0.0000	0.6862 ± 0.0028	82.8 ± 2.0881	0.9723 ± 0.0007	97.7 ± 4.2673	0.9340 ± 0.0033	135.0 ± 11.0454
DO08	**0.7612 ± 0.0009**	20.0 ± 0.0000	0.6841 ± 0.0031	87.2 ± 1.1662	0.9725 ± 0.0013	95.3 ± 5.5507	0.9327 ± 0.0041	140.5 ± 11.8849
DO09	0.7612 ± 0.0016	21.5 ± 0.5000	0.6820 ± 0.0033	98.7 ± 3.5791	0.9719 ± 0.0019	102.7 ± 5.3113	0.9260 ± 0.0033	147.0 ± 14.1492
HD50	0.7593 ± 0.0012	27.9 ± 0.3000	0.6787 ± 0.0019	131.3 ± 6.2618	0.9714 ± 0.0019	149.5 ± 12.9402	0.9237 ± 0.0061	228.2 ± 27.0695
HD350	0.7588 ± 0.0011	16.0 ± 0.0000	0.6875 ± 0.0021	65.2 ± 1.3266	0.9714 ± 0.0010	76.0 ± 2.0494	0.9375 ± 0.0020	110.3 ± 4.0509
HD500	0.7589 ± 0.0011	14.0 ± 0.0000	0.6880 ± 0.0018	57.0 ± 0.7746	0.9703 ± 0.0007	68.9 ± 2.6627	0.9374 ± 0.0011	102.6 ± 6.5452

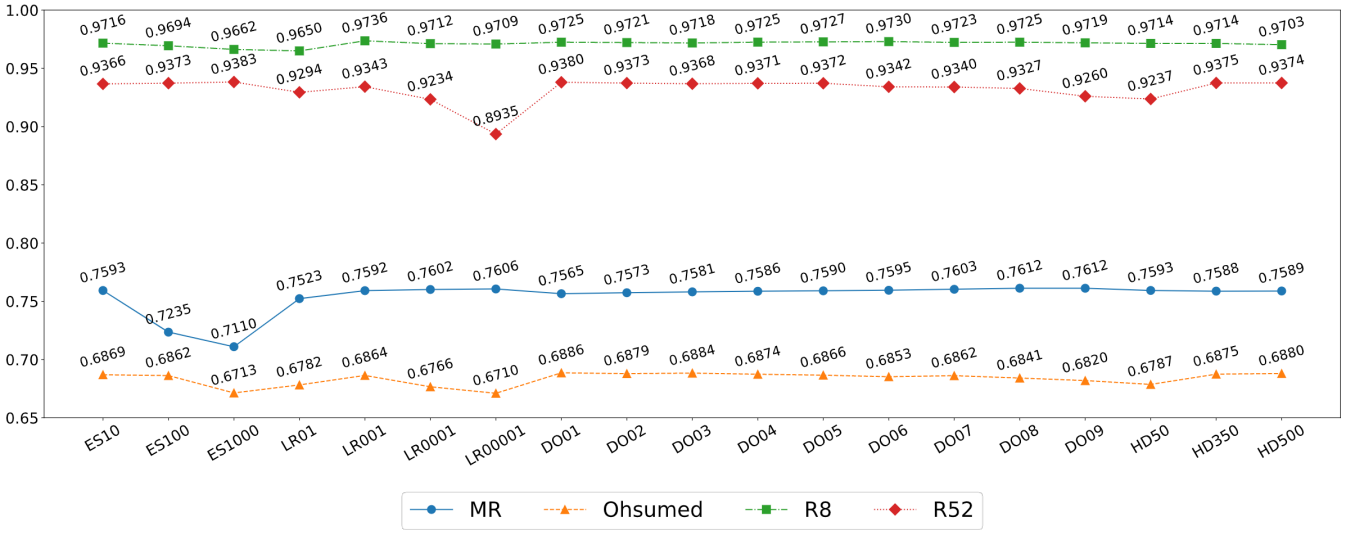

Figure 3: TextGCN average accuracy results for each experiment and dataset. Blue line (solid line with dot markers) shows the results of the MR Dataset, Yellow line (dashed line with triangle-shaped marker) shows the results of the Ohsumed dataset. Green line (dashdot line with square-shaped marker) shows the results of R8. Red line (dotted line with diamond-shaped marker) shows the results of R52. Notice that with some exceptions, all results of a dataset, numerically, tends to be similar.

on the MR Dataset. However, the other parameters do not seem to affect the results as much.

On the Ohsumed dataset, experiments that changed the early stop, and the learning rate presented different results. The other experiments achieved similar results. The exception is the Hidden 1, in which the higher the number of nodes, the better.

On the R8 dataset, the early stop parameter seems to affect the results of the model slightly. All the other experiments obtained similar results. Even the best result (when the Learning Rate was 0.01) has small improvements over the other experiments.

On the R52 dataset, different settings resulted in different results. This dataset seems to be very sensitive to the Learning Rate parameter, achieving a delta of 0.0408. It also showed to be slightly sensitive to other parameters: Dropout achieved a delta of 0.012, and Hidden 1 delta is 0.0115.

The level of confidence showed in Table 5 allows the authors to say that if a user makes some parameter tuning of the TextGCN, better results are possible. Some suggestions to start tuning the parameters of the TextGCN model for each dataset, based on the data retrieved by the experiments conducted are in Table 7.

Notice that the described parameters in Table 7 are guesses oriented by the results of the experiments showed. Therefore they may not improve the results of further experiments.

Table 7: Suggestions of parameters to start tuning the model for each dataset

Dataset	Early Stop	Learning Rate	Dropout	Hidden 1
MR	ES $\leq$ 10	LR $\leq$ 0.00001	0.8 $\leq$ DO $\leq$ 0.9	HD $\geq$ 500
Ohsumed	ES $\leq$ 10	LR $\approx$ 0.01	DO = 0.9	HD $\geq$ 500
R8	ES $\leq$ 10	LR $\approx$ 0.01	DO = 0.6	50 $\leq$ HD $\leq$ 350
R52	ES = 100	LR $\approx$ 0.01	DO = 0.1	HD $\approx$ 350

5 CONCLUSIONS

This work's objective is to present the impact of each studied parameter in the TextGCN model. In the process, a methodology was adopted to compare them. The authors expect that newer methods and models present this kind of data in the future. So the use of newly developed techniques could be higher since the optimization process to a specific use would be easier.

The experiments were unable to provide support to a global optimal configuration of the TextGCN that could be used in all datasets, with the same success rate. Some datasets can benefit from lower Dropout, some from higher Dropout, and some do not change with Dropout change. Thus, it is essential always to analyze the impact of each parameter in the target dataset. This analysis can help the user of a model improve their model's performance with an oriented guess, guiding them to the best parameters.

Another observation made during the experiments is that it does not matter the parameters used in each dataset (in the context of the experiments executed)—the impact of each parameter alone tends to be very small. We made the same statistical test between the results achieved, but the numerical difference is not significant to justify the comparison. All these tests are available in the GitHub repository. However, there are scenarios that the parameters can present negative impacts. Examples of such scenarios are when Early Stop is 1000 and when the Learning Rate is set to 0.0001.

The achieved consistency in the results could be because of the shape of the modelled graph. We speculate that the type of documents is which leads to these results. R8 and R52 are text documents that contain fewer different words, while the Ohsumed and the MR are datasets that contain more variability in the text. Observe that even though the R52 has more documents than Ohsumed, R52 has fewer words, as Table 1 shows. Once there is less variability in the data, graphs diameters tend to be smaller because each node interacts with less different nodes. This event allows the accessibility of the entire datasets with fewer GCN layers while training. As the experiments used only 2 GCN layers, as Yao and colleagues did, graphs with big diameter may not achieve the best performance. This topic needs further investigation in future work.

During this work, the variability in the settings leading to better results drew the attention. TextGCN's authors [9] could have better results if the authors experimented with different sets of parameters, although this improvement is not expressive. The evidence is that in just analyzing how each parameter affected each dataset, it was possible to improve three datasets' results. Therefore, future work can investigate how accurate the suggestions made in Table 7 were and how good the TextGCN is in other domains. In further investigation, aiming to enhance the TextGCN results, a relevant issue highlighted by this work is the potential of the relationship between the graph and the model used to classify it.

ACKNOWLEDGMENTS

This research was sponsored by FNDE/MEC and UFPR/C3SL. We would like to thank Nvidia, that supported this research with GPU donation.

REFERENCES

[1] Simon Kemp. Digital 2020: 3.8 billion people use social media, Jan 2020. URL https://wearesocial.com/uk/blog/2020/01/digital-2020-3-8-billion-people-use-social-media. Last access in October 27th, 2020.

[2] Rodrigo Moraes, João Francisco Valiati, and Wilson P. Gavião Neto. Document-level sentiment classification: An empirical comparison between svm and ann. *Expert Systems with Applications*, 40(2):621 – 633, 2013. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2012.07.059. URL http://www.sciencedirect.com/science/article/pii/S0957417412009153.

[3] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1167. URL https://www.aclweb.org/anthology/D15-1167.

[4] Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xuejie Zhang. Refining word embeddings for sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 534–539, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1056. URL https://www.aclweb.org/anthology/D17-1056.

[5] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing [Review Article], aug 2018. ISSN 15566048.

[6] Chen Zhang, Qiuchi Li, and Dawei Song. Aspect-based sentiment classification with aspect-specific graph convolutional networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4568–4578, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1464. URL https://www.aclweb.org/anthology/D19-1464.

[7] Alexios Koutsoukas, Keith J Monaghan, Xiaoli Li, and Jun Huan. Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. *Journal of cheminformatics*, 9(1):42, 2017.

[8] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

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

[10] Bo Pang and Lillian Lee. 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*, ACL '05, page 115–124, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219855. URL https://doi.org/10.3115/1219840.1219855.

[11] Jian Tang, Meng Qu, and Qiaozhu 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*, pages 1165–1174. ACM, 2015.