Shapeshifter and the impact of representation in classification

Anonymous ACL submission

Abstract

This work explores the representation format of text during the classification process. We defined eight types of representations using graphs to study the impact of the representation in a model. We build the graphs based on the dependency tree and input them into the UGformer to classify the documents. As a result, we observed that the best result is always at least two percentage points higher than the worst result.

1 Introduction

Text classification is a task of the Natural Language Processing (NLP) field that may be applied to solve a wide variety of problems. Using text classification, practitioners can categorize sentiments about news, products, and services (Liu, 2020). They can also use it to identify hate of speech (Paz et al., 2020) and figurative language such as irony (Zhang et al., 2019) (Van Hee et al., 2018).

However, most of the works in the literature only consider the model used, not the representation of the information. There is little research that considers representing the text differently. That raises the question: *"What if I represent this data like that?"*. The papers concerning this task usually try to improve the results by creating new machine/deep learning models or parameter tuning existing models, but rarely trying to represent the text differently.

This paper follows this idea of *"What if?"* question and analyzes three distinct languages (English, German and Portuguese) into eight different representations. The comparisons made allow us to answer the research questions: **R1**: Can the shape we use to represent the text change the result of the same model? **R2**: If the representation format matters, how does it matter? **R3**: Is the result achieved for the English extensible for other languages?

This paper is structured to answer these research questions, as follows: Firstly, Section 2 analyzes the related work. Following, Section 3 describes the experiments and Section 4 presents the results. Finally, Section 5 concludes this paper and answer the research questions.

2 Related Work

One of the most basic ways to represent texts is using the Bag of Words (BoW) method. One of the first mentions of this technique in linguistics came from Harris (1954). The BoW consists of a list of words. This list contains how many of each word the document contains. Using this technique a matrix can be built where the rows represent the documents, and the columns represent the words. Each cell of this matrix is the counting of each word in each document. In this manner, the computer can represent all words of all documents. However, this method does not consider the relations between the words or the word's semantics or context.

Ma et al. (2018) developed a work where they use the BoW in the training of the model to enhance the training process to the translation process. According to Ma et al. (2018) it is possible to exist multiple correct translated sentences. They also state that usually, the correct translations have similar BoW. Therefore, in their method, the translation's target is not only the correct sentence but the BoW generated by the correct translation and its sentence. In another work, Fu et al. (2020) used a modified version of the BoW to generate paraphrases.

Another usual way to represent text is by using n-grams. A n-gram is similar to a sliding window of size n. In the sentence *"I parked my car in your garage"* the 2-gram (also known as bigram) representation is `I parked; [parked my] [my car]; [car in]; [in your]; [your garage]`. One of early usages of this technique was made by Broder et al. (1997). They used the n-gram to analyze the text from the web and search for plagiarism.

Gupta et al. (2019) developed a work that uses the n-grams. In their work, (Gupta et al., 2019) used the n-grams to enhance the creation of word vectors. Another work that uses the n-grams is the work by (Qi et al., 2020). Their work built a model that predicts the next n-gram of a sentence.

In 2013, Mikolov et al. (2013) created the word2vec. This technique uses machine learning to build a vector representation. This vector considers the syntactical and semantical information of the word. This idea spread and inspired the authors of GloVe (Pennington et al., 2014) and Fasttext (Joulin et al., 2016) to build similar techniques.

Rousseau and Vazirgiannis (2013) created another representation technique in 2013. Their technique represents text documents as graphs, the Graph of Words (GoW). In transforming a document into a directed graph, a node represents each unique word. Each arc represents a co-occurrence between the terms within a fixed-size sliding window. The scoring is giving by a function developed by Rousseau and Vazirgiannis (2013) called Term Weights - Inverse Document Frequency (TW-IDF) (Equation 1).

Consider the following notation: N is the size of collection; d is a document; t is a term that belongs to a document d; $df(t)$ is the document frequency of term t across the collection N; b is a parameter set experimentally (the default value is 0.20); $avgdl$ is the average document length; $tw(t, d)$ is the weight of the vertex associated with the term t.

$$\text{TW-IDF}(t, d) = \frac{tw(t, d)}{1 - b + b \times \frac{|d|}{avgdl}} \times \log \frac{N + 1}{df(t)} \quad (1)$$

After the creation of the GoW technique by (Rousseau and Vazirgiannis, 2013), other authors created variations in the graph creation Malliaros and Skianis (2015) and Yao et al. (2019).

Huang et al. (2020) developed a different way to build a graph from a text. In their work, Huang et al. (2020) create a directed graph in which each node is a vector that represents the word with dimension d. Each edge starts from a word in the text and stops in an adjacent word.

The classification uses a message passing mechanism in convolution. The message passing mechanism first collects information from adjacent nodes and updates its representations based on its original representations and collected information. The information is collected from the neighbors and their edges to create a new node representation.

This new representation is responsible for indicating how much of the original information the model must keep. Softmax (Bridle, 1990) and the ReLu (Nair and Hinton, 2010) functions can label the new representation of the nodes. Thus the text is classified. The model's objective is to minimize the cross-entropy loss function loss $= -g_i \log y_i$, where g_i is the ground truth, and y_i is the predicted label.

When compared to the TextGCN model in three datasets (R8, R52, and Ohsumed), the model proposed by Huang et al. (2020) consumes less memory and improves the accuracy.

The syntax trees have already been explored to represent texts. (Zhang et al., 2010) conducted one of the works in this regard. This paper will further explore this representation method.

3 Experiments

All experiments conducted intended to explore the impact of the text representation in the classification. Therefore, we did not apply any parameter tunning to the model as suggested by (Ehrenfried and Todt, 2021). Thus, the results presented by Section 4 could be better if we get the best representation model and apply parameter tunning to the deep learning model.

The experiments used the UGformer developed by Nguyen et al. (2019) in its first version as the Deep Learning model used to classify text. We made some minor changes to allow the use of custom node features. The code used to classify is text is available at `https://anonymous.4open.science/r/Graph-Transformer-C9FA/`. The UGformer, receives graphs as inputs, processes them using self-attention and feed-forward layers, and outputs a category for each graph. Subsection 3.1 provides more details.

As the UGformer receives graphs, we must represent text as graphs. Therefore, we developed software that transforms text into graphs. This software is available at `https://anonymous.4open.science/r/Text_Representation-2CB3/`. More details in this transformation is given in Subsection 3.2.

3.1 UGformer

The objective of UGformer is that given a set of M graphs $\{\mathcal{G}_m\}_{m=1}^{M}$ and their corresponding class labels $\{\mathrm{y}_m\}_{m=1}^{M}$, the UGformer is expected to learn a plausible embedding $\mathbf{o}_{\mathcal{G}_m}$ for each graph $\mathcal{G}_m$ to predict its label y_m.

Each graph $\mathcal{G}$ is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V}$ is a set of node, $\mathcal{E}$ is a set of edges, and $X \in R^{|\mathcal{V}|\times d}$ represents feature vectors of nodes. Theses graphs are fed into a transformer network to learn a node embedding $\mathbf{o}_{\mathrm{v}}$ for each node $\mathrm{v} \in \mathcal{V}$. Later, all node embeddings are summed to obtain the $\mathbf{o}_{\mathcal{G}}$, as showed by Equation 2.

$$\mathbf{o}_{\mathcal{G}} = \sum_{\mathrm{v}\in\mathcal{V}} \mathbf{o}_{\mathrm{v}} \tag{2}$$

To learn the node embeddings $\mathbf{o}_{\mathrm{v}}$, a set of neighbors $\mathcal{N}_{\mathrm{v}}$ of each node $\mathrm{v} \in \mathcal{V}$ is sampled and inputted with the origin node to the UGformer learning.

The nodes $\mathcal{N}_{\mathrm{v}} \cup \{\mathrm{v}\}$ are transformed into a sequence of feature vectors that are iteratively refined by the Transformer encoder followed by a feed-forward network and layer normalization.

Therefore in the k-th layer, a node $\mathrm{v} \in \mathcal{V}$, at the step t, the vector representation $\mathrm{h}_{t,\mathrm{u}}^{(k)}$ for all nodes $\mathrm{u} \in \mathcal{N}_{\mathrm{v}} \cup \{\mathrm{v}\}$ is defined as showed by Equation 3

$$\mathrm{h}_{t,\mathrm{u}}^{(k)} = \mathrm{LNORM}\left(\mathrm{x}_{t,\mathrm{u}}^{(k)} + \mathrm{TRANS}\left(\mathrm{x}_{t,\mathrm{u}}^{(k)}\right)\right) \tag{3}$$

Where $\mathrm{x}_{t,\mathrm{u}}^{(k)}$ is defined by Equation 4

$$\mathrm{x}_{t,\mathrm{u}}^{(k)} = \mathrm{LNORM}\left(\mathrm{h}_{t-1,\mathrm{u}}^{(k)} + \mathrm{ATT}\left(\mathrm{h}_{t-1,\mathrm{u}}^{(k)}\right)\right) \tag{4}$$

ATT(.) and TRANS(.) in Equations 3 and 4 denote a transformer network and a feed-forward network respectively.

After T steps the output representation $\mathrm{h}_{T,\mathrm{v}}^{(k)}$ is fed to the next $(k+1)$-th layer. Therfore, multiple layers are stacked on top of each other to capture k-hops neighbors in the UGformer. At the end, all output representations of all K layers are concatenated to infer the node embedding $\mathbf{o}_{\mathrm{v}}$. The graph embedding $\mathbf{o}_{\mathcal{G}}$ is the sum of all nodes embeddings $\mathbf{o}_{\mathrm{v}}\ \forall\ \mathrm{v} \in \mathcal{V}$. The model parameters are learned by minimizing the cross-entropy loss function.

3.2 Text to Graph

As mentioned in the the beginning of this Section, we developed a software that transforms text into graphs.

This software requires that the user creates a sort of driver to transform the original text dataset into a set of files, where each file will be a graph. These files follow a standardized name pattern `<class>_<id>.txt` where `<class>` is a number that corresponds to the class of the document, and `<id>` is a number that can be anything; usually, we use the order of processing. All these files must be in a directory. Therefore, when executing the software we developed, the user must pass this directory as the input directory. There are several options in the software, but the most important in this manuscript are the `-l`, `-g` and `-t`.

The `-l` option allows the user to choose the language of the dataset. At the current step, the software accepts only Portuguese (pt), English (en), and German (de).

The `-g` option allows the user to choose the graph shape of the transformation. There are three options: `tree_only`, `tree_and_order` and `tree_and_order_multi_graph`. The `tree_only` option builds a dependency tree using the text provided. The `tree_and_order` builds a dependency tree and adds arcs of the order of reading. If an arc already exists because of the dependency tree, the arc for *order* will not be added. Finally, the `tree_and_order_multi_graph` option builds a dependency tree and adds arcs of the order of reading, but differently from the other option, it adds the arc even if already there is an arc concerning the same words.

The `-t` is the type of node tag. A node tag is represents the type of the node. Here there are six options: `none`, `dep`, `pos`, `dep-pos`, `pos-dep` and `sqrt_product`.

The `none` marks all nodes of the graph as the same type. The `dep` marks the nodes with a number that maps its dependency information. Although we know that the arc better represents the dependency information, the model chosen to conduct these experiments is not compatible with arcs with tags. The `pos` marks the nodes with a number that maps its part-of-speech information. The `dep-pos` composes the dependency number and the part-of-speech information by multiplying the dependency number by 100 and summing it with the part-of-speech number. The `pos-dep` is similar to the `dep-pos`, just inverting the order - Multiply the POS by 100 and sum it with de DEP. Finally, the `sqrt_product` marks the node with a custom function - It multiplies the DEP with the POS number, takes the square root of it. Because the result must be an integer, we apply the ceiling function to transform it into an integer.

The `dep-pos` and `pos-dep` arrived as a solution to integrate the information about dependency and part-of-speech in the node tag. The multiplication by 100 serves as a separator when summing both pieces of information. Otherwise it could happen that different `dep` and `pos` could sum to the same value. In this sense, we investigated what happens when different information type has the same label by using the `sqrt_product` node tag. Because only multiplying a number by another could lead to an unnecessary spread in the numbering, we used the square root.

As a result of the processing, it generates a file containing all documents as directed graphs that use the combination of parameters passed as described previously. This file containing all graphs uses the format specified by Zhang et al. (2018).

3.3 Experiments description

Since we described how UGformer works and how the options of our software allow the creation of graphs using text, we can describe the steps used to experiment with different representations.

We first built the three different graphs (Tree, Tree-Order, and Tree-Order-Multigraph) using the none option for the node tag. Then, we used the Tree-Order-Multigraph as the base for the other five types of node tag: DEP, POS, DEP-POS, POS-DEP SQRT_PRODUCT, which gave us a total of eight types of text representation for each dataset.

The UGformer used the same parameters for all datasets in all configurations. The parameters used are in Table 1

The UGformer uses the StratifiedKFold technique from the sklearn libray[1], using the number of splits as 10. Therefore, 90% of the dataset is used to train the model and 10% to test the model.

[1]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html - Last access December 2nd, 2021

Parameter	Value
Batch Size	4
Dropout	0.5
Fold IDX	1
Feed Forward Hidden Size	1024
Learning Rate	0.0001
Number of Epochs	30
Number of Hidden Layers	2
Number of Neighbors	4
Number of Timesteps	1
Sampled Number	512

Table 1: Parameters of UGformer used in all experiments.

3.3.1 Datasets used

This work used four datasets - MR, Ohsumed, B2W-Rating-Balanced, and 10kGNAD. The MR and Ohsumed datasets were extracted from the TextGCN project (Yao et al., 2019). While the B2W-Rating-Balanced[2] and 10kGNAD were extracted from GitHub[3]

The MR dataset is a movie review dataset for binary sentiment classification, in which each review only contains one sentence (Pang and Lee, 2005)[4]. The corpus has 5,331 positive and 5,331 negative reviews.

The Ohsumed corpus[5] 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 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.

The B2W-Rating-Balanced (Ehrenfried and Todt, 2022) consists of 41,945 reviews divided into five classes equally. These classes are the review score given by the review's author. This score is an integer number between 1 and 5, where one is the lowest and five is the highest. Portuguese is the

[2]https://github.com/HenriqueVarellaEhrenfried/B2W-Datasets

[3]https://tblock.github.io/10kGNAD/

[4]http://www.cs.cornell.edu/people/pabo/movie-review-data/ - Accessed on October, 27th 2020

[5]http://disi.unitn.it/moschitti/corpora.htm - Accessed on October, 27th 2020

Dataset	Metric	Value
MR	Language	English
	Docs	10,662
	Classes	2
	Mean tokens	21.04 +/- 9.42
	Max tokens	60
	Min tokens	1
B2W	Language	Portuguese
	Docs	41,945
	Classes	5
	Mean tokens	26.36 +/- 25.77
	Max tokens	796
	Min tokens	1
10kGNAD	Language	German
	Docs	10.273
	Classes	9
	Mean tokens	345.85 +/- 257.61
	Max tokens	4804
	Min tokens	3
Ohsumed	Language	English
	Docs	7,400
	Classes	23
	Mean tokens	193.74 +/- 76.36
	Max tokens	615
	Min tokens	30

Table 2: Datasets used in the experiments. In the mean row, the standard deviation is the number after the +/- sign.

language of all reviews of this dataset.

The 10kGNAD consists of 10273 news articles from an Austrian online newspaper categorized into nine topics (Web, Panorama, International, Wirtschaft, Sport, Inland, Etat, Wissenschaft, Kultur). It is important to note that the 10kGNAD is an unbalanced dataset. The goal of this dataset is to categorize each document into one topic. All articles are in German. Table 2 presents a summary of the content of each dataset.

Table 2 shows information about the language of the dataset, the total number of documents, the number of classes, the average number of tokens and its standard deviation (denote by the +/- sign), the maximum and the minimum number of tokens in a document.

3.3.2 Metrics

We use the model's accuracy to evaluate the model's performance with the representation method used. In this regard, we capture the best result and its epoch and the last result in the 30^{th} epoch. We also calculated the mean and standard deviation between epochs starting from the second epoch because the first epoch is usually used to learn the first parameters of the model. Thus the results of the first epoch are usually the lowest.

4 Results

This Section presents the results achieved with the eight different representations. Table 3 presents the metrics collect during the experiments. While Figure 1 shows the results across epochs.

Table 3 Contains 5 metrics: *Mean*, *Std Dev*, *High Acc*, *High Acc epoch*, and *Last Acc*. The *Mean* and *Std Dev* use as data the per epoch results starting from the second epoch to calculate the average and standard deviation as Section 3.3.2 explains. The highest accuracy achieved is represented by the metric *High Acc*. The *High Acc epoch* shows which epoch happend the best accuracy. Finally, the *Last Acc* metric represents the last accuracy achieved (in epoch 30). Following we will discuss the results achieved by dataset.

The MR dataset is the dataset with the second-lowest *Std Dev* metric but the second-worst in its delta (Best result - Worst result). We observed that the MR accuracy could vary 3.09 percentage points in its best accuracy, only changing the representation method. The MR seems to benefit from having a node tag since the DEP POS experiment achieved the best accuracy in both the highest accuracy and the last accuracy.

The B2W-Rating dataset is the dataset with the lowest *Std Dev* and the lowest delta in the *Std Dev* metric. However, this resulted in a minor difference between the best accuracy and the worst - Only 1.02 percentage points. Like the MR dataset, the DEP POS experiment obtained the best accuracies. Another similar fact between these two datasets is the results of the DEP and POS experiments. Both obtained better results in de the DEP experiment than the POS experiment.

The 10kGNAD dataset is the dataset with the highest *Std Dev* and *Std Dev* delta. Also, it is the dataset with the highest difference between the highest best accuracy and the lowest best accuracy. The difference is 4.77 percentage points of difference. Given that data, we suspect that the German language is more sensitive to text representation than Portuguese (B2W-Rating) and English (MR and Ohsumed). Unlike the other two datasets

Dataset	Metric	Tree	Order	Multigraph	DEP	POS	DEP POS	POS DEP	SQRT PROD
MR (English)	Mean	78.28	77.06	77.11	77.94	77.76	78.94	77.48	**79.04**
	Std Dev	2.80	**1.56**	4.10	2.69	2.70	3.63	2.88	1.91
	High Acc	81.36	79.01	81.91	81.26	80.60	**82.10**	80.51	80.97
	High Acc epoch	20	11	25	23	10	14	17	19
	Last Acc	79.76	75.73	79.94	80.04	80.00	**80.60**	78.91	79.57
B2W (Portuguese)	Mean	54.01	53.37	52.87	54.11	53.46	**54.13**	53.21	53.03
	Std Dev	**1.03**	2.46	1.98	1.97	1.65	1.83	1.96	2.10
	High Acc	55.14	55.45	55.30	55.83	55.47	**56.16**	55.21	55.49
	High Acc epoch	10	17	17	24	24	27	18	20
	Last Acc	54.71	54.68	54.33	54.66	54.45	**55.64**	54.42	54.14
10kGNAD (German)	Mean	74.74	76.37	**76.37**	74.61	74.93	75.59	75.16	73.78
	Std Dev	**4.79**	6.38	7.40	7.03	6.15	5.03	5.42	5.49
	High Acc	81.23	83.27	**84.44**	81.81	81.61	82.49	81.23	79.67
	High Acc epoch	23	29	26	30	25	25	20	27
	Last Acc	76.65	**83.17**	80.54	81.81	80.25	80.06	80.93	77.92
Ohsumed (English)	Mean	60.55	61.73	59.48	**62.23**	61.26	61.40	61.40	61.35
	Std Dev	6.62	6.04	6.11	**4.99**	5.60	5.03	5.78	5.50
	High Acc	68.38	**70.81**	68.38	68.92	69.19	68.51	69.19	69.86
	High Acc epoch	28	26	28	15	28	28	30	29
	Last Acc	68.11	66.62	67.70	64.19	66.62	65.27	**69.19**	65.00

Table 3: Results of All experiments. The best result is written in bold and green, while the worst result is underlined and written in red. Tree is the experiment where only the dependency tree is used. Order is the dependency tree and the order of reading. Multigraph is the dependency tree and the reading order, but more than one arc can exist between two nodes. DEP is the Multigraph configuration plus the information of dependency as node tag. POS is the Multigraph configuration plus the information of part-of-speech as node tag. DEP POS is the Multigraph configuration plus the information of dependency and part-of-speech as node tag. POS DEP is the Multigraph configuration plus the part-of-speech information and dependency as node tag. SQRT PROD is the Multigraph configuration plus the square root of the product of the index of the dependency times the index of the part-of-speech as node tag.

already discussed, the 10kGNAD did not benefit from node tag information. The best result was achieved in the Multigraph experiment, and the best last accuracy result was in the Order experiment.

The Ohsumed dataset obtained the second-highest *Std Dev*, but it did not vary much. Its delta is only 1.63. As expected from the previous experiments, the lowest the variation in the *Std Dev*, the lower the variation in the highest accuracy. The difference between the best highest accuracy and the worst highest accuracy is 2.43 percentage points, differently from the MR dataset, the other English dataset, the Ohsumed dataset its best result in the Order experiment.

Given the last result, we suspect that the best representation depends not only on the language but also on the size of the text. The 10kGNAD and the Ohsumed contain more text than the MR and the B2W per document, favoring the DEP POS representation. When we increased the quantity of text, the DEP POS experiment became just another average experiment. We need to investigate if the same behavior happens in Portuguese or German. Another fact is that the 10kGNAD and Ohsumed are the datasets with the most quantity of classes, 9 and 23, respectively. Perhaps because of the higher number of classes, more layers must be used to

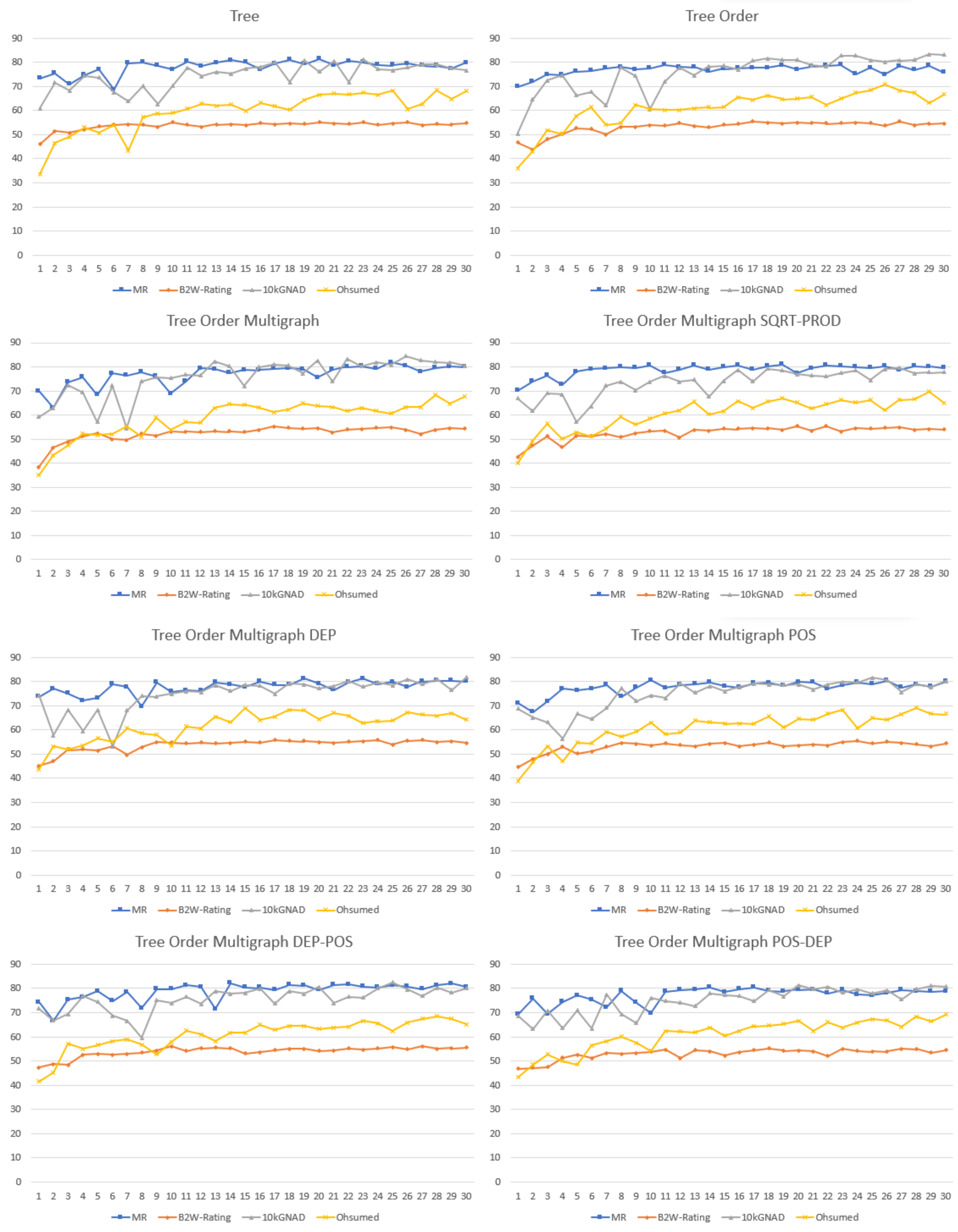

Figure 1: Results per epoch in each type of representation. The blue line with a square represents the MR dataset; The orange line with a diamond represents the B2W-Rating dataset; The gray line with a triangle represents the 10kGNAD dataset; The yellow line with a cross represents the Ohsumed dataset. Observe that the variation usually occurs in the first ten epochs. After these epochs, the line tends to stabilize.

classify the text better.

We also observed that the dataset could take longer or shorter to reach a local maximum depending on the chosen representation, as Figure 1 shows.

5 Conclusions

This work presented eight different manners to represent text and tested them with three different languages - English, Portuguese and German. During this work, we developed three different graphs, and one of them contains six variations concerning the tag of each node. The experiments conducted gave us the answer to the research questions: Research question **R1** asked about if the shape of representation can influence the result of the model. As discussed in Section 4, the model is not the only variable in the classification process. The representation shape can influence it.

Research question **R2** asked about how the representation format influences the classification process. As presented by Table 3, different representation formats obtain different performances depending on the size of the dataset and its language. However, the DEP POS variant tends to perform better for small texts (i.e., Something similar in size as a Twitter post).

Research question **R3** asked if the results achieved for the English language could be extended to other languages. As we tested German and Portuguese, we could verify that, similarly to English, changing the representation format can change the classification process. However, we observed that the best representation format could change depending not only on the language but also on the text size. As the experiments concerning MR and Ohsumed showed. Nevertheless, more studies must be conducted to understand better the behavior of different documents with different sizes with types of representations.

Future work will work on other representations methods and investigate the impact of different size documents in the classification process.

Acknowledgements

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

References

John Bridle. 1990. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. 1997. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8-13):1157–1166.

Henrique Varella Ehrenfried and Eduardo Todt. 2021. Analysis of the impact of parameters in TextGCN. In *Anais Computer on the Beach 2021*, pages 014–019.

Henrique Varella Ehrenfried and Eduardo Todt. 2022. Should i buy or should i pass: E-commerce datasets in portuguese. In *PROPOR 2022*.

Yao Fu, Yansong Feng, and John P Cunningham. 2020. Paraphrase generation with latent bag of words. In *Proceedings of the 2019 Advances in Neural Information Processing Systems (NeurIPS 2019)*.

Prakhar Gupta, Matteo Pagliardini, and Martin Jaggi. 2019. Better word embeddings by disentangling contextual n-gram information. In *NAACL-HLT (1)*, pages 933–939. Association for Computational Linguistics.

Zellig S. Harris. 1954. Distributional structure. *iWORD/i*, 10(2-3):146–162.

Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2020. Text level graph neural network for text classification. Technical report.

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

Bing Liu. 2020. *Sentiment Analysis Mining Opinions, Sentiments, and Emotions*. Cambridge University Press.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. In *ACL 2018*.

Fragkiskos D. Malliaros and Konstantinos Skianis. 2015. Graph-based term weighting for text categorization. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015*, pages 1473–1479. Association for Computing Machinery, Inc.

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

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA. Omnipress.

Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. 2019. Universal graph transformer self-attention networks. *arXiv preprint arXiv:1909.11855*.

Bo Pang and Lillian Lee. 2005. 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. Association for Computational Linguistics.

María Antonia Paz, Julio Montero-Díaz, and Alicia Moreno-Delgado. 2020. Hate speech: A systematized review. *SAGE Open*, 10(4):215824402097302.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2401–2410.

François Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and tw-idf: New approach to ad hoc ir. In *Proceedings of the 22nd ACM International Conference on Information; Knowledge Management*, CIKM '13, page 59–68, New York, NY, USA. Association for Computing Machinery.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. SemEval-2018 task 3: Irony detection in English tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50, New Orleans, Louisiana. Association for Computational Linguistics.

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. AAAI Press.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *AAAI*.

Shiwei Zhang, Xiuzhen Zhang, Jeffrey Chan, and Paolo Rosso. 2019. Irony detection via sentiment-based transfer learning. *Information Processing & Management*, 56(5):1633–1644.

Wei Zhang, Peifeng Li, and Qiaoming Zhu. 2010. Sentiment classification based on syntax tree pruning and tree kernel. In *2010 Seventh Web Information Systems and Applications Conference*. IEEE.