Impact of text preprocessing on effectiveness of document classification by Graph Neural Networks[*]

Volodymyr Matsko[1*,†], Solomiia Lyaskovska[2,†] and Oleksandr Starodub[3,†]

[1] *Lviv Polytechnic National University, S. Bandera, str. 12, Lviv, 79013, Ukraine*

[2] *Department of Systems of Artificial Intelligence, Lviv Polytechnic National University, S. Bandera, str. 12, Lviv, 79013, Ukraine*

[3] *Department of Management and International Business, Lviv Polytechnic National University, S. Bandera, str. 12, Lviv, 79013, Ukraine*

Abstract

Preprocessing of text data is a key step in document classification, especially when utilising graph neural networks. Existing research on the impact of preprocessing has focused on both traditional approaches, currently more popular convolutional neural networks, and more recent transformer-based models. To the best of our knowledge, the influence of preprocessing on graph neural networks remains insufficiently explored. This study examines the individual impact of 10 popular preprocessing methods on document classification using the TextGCN model. The results indicate that the selection of effective preprocessing methods can significantly enhance classification accuracy and drastically reduce the size of the graph. Moreover, the removal of rare tokens and punctuation yields the most substantial improvements in classification performance and graph size. In contrast, the processing of contractions has a negligible impact on accuracy, suggesting its limited relevance in this context. These results underscore the importance of strategic preprocessing choices in optimising the effectiveness of graph-based document classification models.

Keywords

Text preprocessing, document classification, Graph Neural Networks, TextGCN

1. Introduction

The classification of text documents remains one of the key tasks in natural language processing, with numerous applications in information systems, ranging from automatic sentiment analysis to the thematic classification of scientific publications and other documents. The development of graph neural networks has opened new prospects for modelling textual data, enabling the effective integration of both semantic and structural information. In particular, Graph Convolutional Networks GCNs demonstrate significant potential in classification tasks, as they are capable of accounting for the relationships between documents and individual text elements, which is difficult to achieve with traditional methods or convolutional neural networks [1]. Furthermore, the study by Bugueño and Melo shows that graph neural networks achieve performance comparable to transformer-based models when used for the classification of long texts, although they yield poorer results on short texts [2]. It is particularly noteworthy to mention the effectiveness of graph neural networks in semi-supervised learning, according to Kipf and Welling [3] and Yao et al. [1].

In the context of smart industries, these technologies are particularly relevant, as modern enterprises must rapidly process vast volumes of text documents. Automated classification of text documents not only allows for the systematisation of existing documents for quick access to critical information, but also enables the effective classification of incoming documents.

Preprocessing plays a crucial role in reducing the dimensionality of text representation, which, in turn, affects computational costs and model accuracy. Camacho-Collados et al. demonstrated not

[*] *SmartIndustry 2025: 2nd International Conference on Smart Automation & Robotics for Future Industry, April 03-05, 2025, Lviv, Ukraine*

[1*] Corresponding author.

[†] These authors contributed equally.

volodymyr.i.matsko@lpnu.ua (V. Matsko); solomiya.y.lyaskovska@lpnu.ua (S. Lyaskovska); oleksandr.v.starodub@lpnu.ua (O. Starodub);

0009-0008-2415-0739 (V. Matsko); 0000-0002-0822-0951 (S. Lyaskovska);

only the effectiveness of specific data preprocessing methods, but also the advantages of preprocessing the corpus from which word embeddings are generated for use in convolutional neural networks [4]. HaCohen-Kerner et al. [5] and Siino et al. [6] further validated the efficacy of various data preprocessing techniques for both classical classification methods and convolutional neural networks.

Despite the relatively extensive research on the application of graph neural networks for text document classification, the issue of data preprocessing remains, to the best of our knowledge, insufficiently examined. Moreover, since some graph convolutional networks demonstrate greater effectiveness when using one-hot vectors rather than word embeddings [1][7], this issue becomes even more pertinent as it may reduce the size of the graph.

In this paper, we assess the effectiveness of 10 preprocessing methods in the context of graph convolutional networks, analyse their impact on classification accuracy, and discuss the prospects for further research.

2. Data, methods and models

2.1. Dataset

As our data corpus, we use the "Ohsumed"[1] dataset, which contains annotations of medical articles on cardiovascular diseases published in 1991. This dataset is widely used in studies on the effectiveness of document classification using graph neural networks [8]. We use a subset of articles that have only a single class; it comprises 7,400 documents that are unevenly distributed across 23 classes (diseases) and is further divided into 4,043 test documents and 3,357 training documents.

Following Yao et al. [1], we randomly selected 10% of the training set to create a validation set.

2.2. Preprocessing methods

During the analysis of studies dedicated to text classification using graph neural networks, we discovered that most employ the same preprocessing methods as those used in Yao et al. [1] and Kim [9]. Kim normalised the text to lowercase, split contractions, removed most special characters, and tokenised the remaining punctuation [9]. In addition to the procedures implemented by Kim [9], Yao et al. also removed stop words and words that occur fewer than five times in the corpus [1].

We have selected 10 commonly used preprocessing methods described in the literature, which can be divided into two categories: data standardisation and data cleaning.

The basic idea of data standardisation is simple – to ensure that tokens with the same meaning are represented identically in our vocabulary. For example, converting tokens to lowercase or normalising case involves transforming the entire text into lowercase. This process allows for the consolidation of tokens that differ only in their case (for instance, "Work" and "work" have the same meaning); however, in some cases, a loss of context may occur: the tokens "Apple" and "apple" convey different meanings, as the former denotes a company, while the latter refers to a fruit.

Stemming is the process of extracting the root of a word (the stem) by removing endings and suffixes. For example, the word "buying" is reduced to the common stem "buy". In contrast, lemmatisation is the process of obtaining the lemma (the base form) of a word taking into account its grammatical properties and context. The lemma is a normalised form that reflects the fundamental meaning of the word, and unlike stemming, lemmatisation considers the morphological rules of the language to accurately determine the base form. While these two methods, to the best of our knowledge, are not used in graph neural networks literature, they are quite popular in studies of preprocessing impact and have been employed in the research of HaCohen-Kerner et al. [5] and Siino et al. [6].

[1]http://disi.unitn.it/moschitti/corpora.htm

Tokenisation of punctuation involves separating punctuation characters from the text, as in natural language, text punctuation is typically attached directly to words. This can lead to tokens such as “end.” and “end” being perceived as different tokens; therefore, the text is split into separate tokens: “end” and “.”. Tokenisation of contractions pertains to the English language, where abbreviations exist, for example, “you’re”. Several approaches exist for tokenising such contractions: one approach involves splitting the contraction into two tokens (“you” and “re”), while another converts the contraction into its full form (“you” and “are”).

The main idea of data cleaning is also straightforward – to remove all data that is not relevant to the classification model. Stop word removal involves eliminating articles, conjunctions, pronouns, and other frequently used elements from the text that do not carry significant information for analysis. Additionally, numbers may be removed from the input data, as they are generally not relevant for text classification. Punctuation removal is another popular preprocessing method, since punctuation marks often do not provide the context necessary for many models. Furthermore, the removal of rare tokens is widely applied, particularly in graph neural networks, as reducing the vocabulary size contributes to a decrease in the graph size. For example, Yao et al. applied this method by removing all words that occurred fewer than five times in the texts [1], whereas Huang et al. created "public" edges for nodes that appeared fewer than twice [10].

2.3. Graph Convolutional Network

For our research, we replicated the graph, the TextGCN model, and its parameters as described in Yao et al. [1]. From the data corpus, after applying individual data preprocessing methods, we constructed a graph comprising both document nodes and token nodes. Document nodes are connected by edges to those token nodes that appear in the documents, with the weight of these edges computed using TF-IDF. Token nodes are interconnected when they co-occur in the data corpus, and the weight of the edges between these nodes is computed using PMI. An example of such a graph is presented in Figure 1.

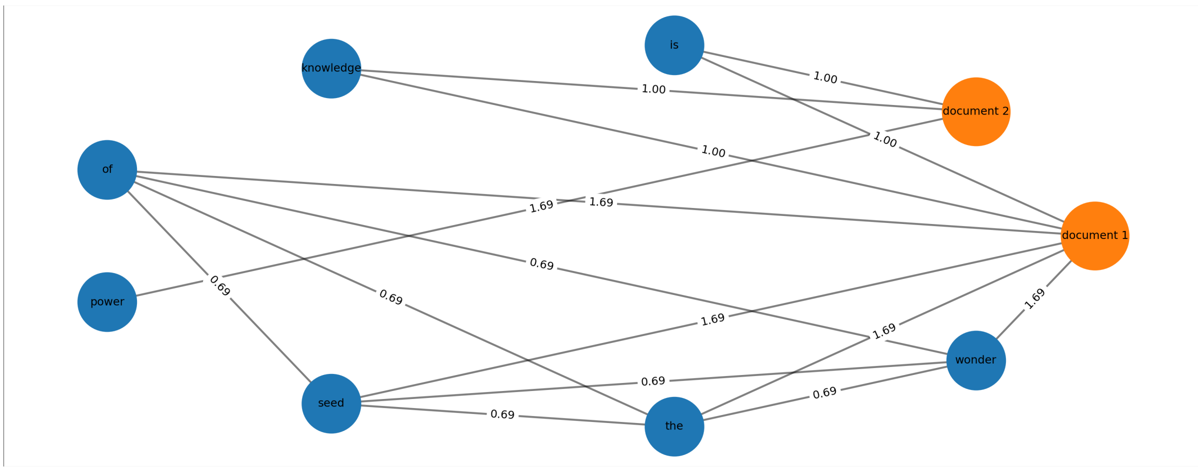

Figure 1: Simple graph with two documents

Each node in the graph is represented as a one-hot vector with a dimensionality equal to the number of nodes in the graph. This graph is processed by a simple two-layer graph convolutional network as described in [3]. The first convolutional layer has an input dimension size equal to the number of nodes in the graph and an output dimension size equal to the embedding size. The second convolutional layer receives the embeddings from the first layer and has an output dimension size equal to the number of classes in the corpus. The network is structured as follows:

$$Z = softmax(\hat{A}\, ReLU(\hat{A}\, X\, W_0)\, W_1) \tag{1}$$

where: W_0 and W_1 denote the weights of the first and second convolutional layers, Â represents the normalised symmetric adjacency matrix as described in Equation (2), X is the one-hot matrix, ReLU is the activation function, and softmax is the classifier as described in Equation (3).

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \quad (2)$$

where: Ã = A + I, where A denotes the adjacency matrix of a graph, I is the identity matrix (used to incorporate self-connections for nodes), and D̃ represents the diagonal node degree matrix, computed as $\tilde{D}_{ii}=\Sigma A(i,j)$.

$$softmax(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{K} \exp(x_i)} \quad (3)$$

where: K denotes the output dimension of the second convolutional layer.

3. Results

As a baseline for comparison, similar to HaCohen-Kerner et al. [5] and Siino et al. [6], we employed a simple tokenisation implementation that splits the text into tokens based on whitespace. We utilised two key metrics: classification accuracy and the number of tokens. The latter is significant because reducing the number of tokens results in a more compact graph, which positively impacts the model's training speed and reduces memory requirements. We ran our model 10 times for each preprocessing method. The results are shown in Table 1 where the "Accuracy" column presents the mean ± standard deviation.

Table 1.
Impact of text preprocessing methods on document classification

Preprocessing method	Number of tokens	Accuracy
None	91090	0.6292 ± 0.0109
Case normalisation	83749	0.6357 ± 0.0033
Stemming	74937	0.6351 ± 0.0073
Lemmatisation	88715	0.6304 ± 0.0076
Stop word removal	90965	0.6345 ± 0.0063
Remove digits	76768	0.6342 ± 0.0158
Remove punctuation	40682	0.6571 ± 0.0294
Tokenise punctuation	40703	0.6564 ± 0.0285
Split contractions	90876	0.6281 ± 0.0171
Expand contractions	91081	0.6285 ± 0.0064
Remove rare tokens	19649	0.6580 ± 0.0085

Splitting and expanding contractions resulted in a statistically insignificant decrease in classification accuracy according to the t-test ($p < 0.05$); it appears that this is due to the nature of the dataset containing scientific medical literature. The quantity and variation of contractions were insufficient to significantly influence the model's accuracy.

Case normalisation, lemmatisation, and removal of digits demonstrated statistically insignificant improvements in classification accuracy, whereas all other data preprocessing methods showed statistically significant enhancements in classification accuracy.

The removal of rare tokens yielded the highest accuracy, significantly increasing model classification accuracy while substantially reducing the vocabulary size. Methods that handle punctuation also demonstrated good accuracy. Specifically, punctuation removal considerably reduced the number of tokens in the vocabulary and significantly improved classification accuracy. In contrast, tokenising punctuation proved less effective, as the GCN uses information from neighbouring nodes in the graph, and it appears that the presence of punctuation-related nodes hinders with effective classification. It is important to note that many standard word tokenisation implementations in libraries automatically perform punctuation tokenisation.

Stemming proved to be a more effective preprocessing method than lemmatisation. This indicates the effectiveness of aggressive consolidation of semantically similar words, which reduces noise and facilitates a more compact representation of the data.

Stop word removal had a negligible impact on the vocabulary size, as the number of these words is quite low; however, it resulted in a significant improvement in classification accuracy.

Removing numbers from the documents led to a significant reduction in vocabulary size and a statistically insignificant improvement in accuracy. In our opinion, various approaches to this method warrant further investigation, since it is possible that replacing numerical tokens with a tag (#NUMBER#) or more detailed tags may also lead to improvement in model accuracy.

4. Conclusions

This study has demonstrated the effectiveness of many popular text processing methods. These methods can not only increase the accuracy of document classification by graph convolutional networks but also reduce the size of the graph. The best results were achieved by removing rare tokens and punctuation, while contraction processing methods showed a statistically insignificant decrease in model accuracy. Overall, the selection of effective data preprocessing methods can significantly improve the final accuracy of the model.

Future research will focus on investigating the impact of other text processing methods (such as noise replacement, abbreviation expansion, error correction, etc.), the effect of combining text preprocessing techniques on various graph neural networks, the use of preprocessed embeddings, and the effect of preprocessing across different datasets.

Declaration on Generative AI

During the preparation of this work, the authors utilised ChatGPT and LanguageTool to identify and rectify grammatical, typographical, and spelling errors. Following the use of these tools, the authors conducted a thorough review and made necessary revisions, and accept full responsibility for the final content of this publication.

References

[1] Yao, Liang & Mao, Chengsheng & Luo, Yuan, Graph Convolutional Networks for Text Classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 2019, 7370-7377. 10.1609/aaai.v33i01.33017370

[2] Bugueño, Margarita & de Melo, Gerard. Connecting the Dots: What Graph-Based Text Representations Work Best for Text Classification using Graph Neural Networks?, in:

Findings of the Association for Computational Linguistics: EMNLP, Association for Computational Linguistics, Singapore, 2023, pp 8943–8960

[3] Kipf, T. N, and Welling, M., Semi-supervised classification with graph convolutional networks, in: ICLR, 2017

[4] Jose Camacho-Collados and Mohammad Taher Pilehvar, On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis, in: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, Association for Computational Linguistics, Brussels, Belgium, 2018, pp 40–46

[5] HaCohen-Kerner Y, Miller D, Yigal Y., The influence of preprocessing on text classification using a bag-of-words representation, PLOS ONE 15(5): https://doi.org/10.1371/journal.pone.0232525

[6] Marco Siino, Ilenia Tinnirello, Marco La Cascia, Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers, volume 121 of Information Systems, 2024, ISSN 0306-4379, https://doi.org/10.1016/j.is.2023.102342

[7] Han, S.C., Yuan, Z., Wang, K., Long, S., & Poon, J., Understanding Graph Convolutional Networks for Text Classification, 2022 ArXiv, abs/2203.16060.

[8] Wang, Kunze & Ding, Yihao & Han, Soyeon, Graph neural networks for text classification: a survey, volume 57 of Artificial Intelligence, 10.1007/s10462-024-10808-0.

[9] Yoon Kim, Convolutional Neural Networks for Sentence Classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp 1746–1751

[10] Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang, Text Level Graph Neural Network for Text Classification, 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), Association for Computational Linguistics, Hong Kong, China, 2019, pp 3444–3450