*algorithms*

MDPI

*Article*

# On Enhancement of Text Classification and Analysis of Text Emotions Using Graph Machine Learning and Ensemble Learning Methods on Non-English Datasets

**Fatemeh Gholami [1], Zahed Rahmati [1], Alireza Mofidi [1,2] and Mostafa Abbaszadeh [1,*]**

[1] Department of Mathematics and Computer Science, Amirkabir University of Technology (Tehran Polytechnic), Tehran 15916-39675, Iran; fatemeh.gholmi@aut.ac.ir (F.G.); zrahmati@aut.ac.ir (Z.R.); mofidi@aut.ac.ir (A.M.)
[2] School of Mathematics, Institute for Research in Fundamental Sciences (IPM), Tehran 15916-39675, Iran
* Correspondence: m.abbaszadeh@aut.ac.ir

**Abstract:** In recent years, machine learning approaches, in particular graph learning methods, have achieved great results in the field of natural language processing, in particular text classification tasks. However, many of such models have shown limited generalization on datasets in different languages. In this research, we investigate and elaborate graph machine learning methods on non-English datasets (such as the Persian Digikala dataset), which consists of users' opinions for the task of text classification. More specifically, we investigate different combinations of (Pars) BERT with various graph neural network (GNN) architectures (such as GCN, GAT, and GIN) as well as use ensemble learning methods in order to tackle the text classification task on certain well-known non-English datasets. Our analysis and results demonstrate how applying GNN models helps in achieving good scores on the task of text classification by better capturing the topological information between textual data. Additionally, our experiments show how models employing language-specific pre-trained models (like ParsBERT, instead of BERT) capture better information about the data, resulting in better accuracies.

**Keywords:** non-English text classification; graph machine learning; ensemble learning method; (Pars) BERT



**Citation:** Gholami, F.; Rahmati, Z.; Mofidi, A.; Abbaszadeh, M. On Enhancement of Text Classification and Analysis of Text Emotions Using Graph Machine Learning and Ensemble Learning Methods on Non-English Datasets. *Algorithms* **2023**, *16*, 470. https://doi.org/10.3390/a16100470

Academic Editor: Boting Yang

Received: 3 August 2023
Revised: 26 September 2023
Accepted: 29 September 2023
Published: 4 October 2023



## 1. Introduction

In the last decade, there has been a tremendous growth in the number of digital documents and complex textual data. Text classification is a classically important task in many natural language processing (NLP) applications, such as sentiment analysis, topic labeling, and question answering. Sentiment analysis is particularly and significantly important in the realms of business and sales, as it enables organizations to gain valuable insights and make informed decisions. In the age of information explosion, processing and classifying large volumes of textual data manually is time-consuming and challenging. Moreover, the accuracy of manual text classification can be easily influenced by human factors such as fatigue and insufficient domain knowledge. It is desirable to automate classification methods using machine learning techniques, which are more reliable. Additionally, this can contribute to increase the efficiency of information retrieval and reduce the burden of information overload by effectively locating the required information [1]. See some of the works on text classification as well as some of its numerous real-world applications in [2–5].

On the other hand, graphs provide an important data representation that is used in a wide range of real-world problems. Effective graph analysis enables users to gain a deeper understanding of the underlying information in the data and can be applied in various useful applications such as node classification, community detection, link prediction, and more. Graph representation is an efficient and effective method for addressing graph

analysis tasks. It compresses graph data into a lower-dimensional space while attempting to preserve the structural information and characteristics of the graph to the maximum extent.

Graph neural networks (GNNs) represent today central notions and tools in a wide range of machine learning areas since they are able to employ the power of both graphs and neural networks in order to operate on data and perform machine learning tasks such as text classification. In this work, we generally aim to pursue some routes of investigation on text classification by GNNs and, in particular, to offer methods which combine ideas of ensemble learning, GNN structures, language models, and graph partitioning techniques for the task of text classification.

Now, we briefly detail our methods, results, proposed approaches, and experiments. In this paper, we will examine and conduct a careful investigation of various combinations of different language models as well as ensemble learning methods and several graph neural network structures (GNNs). In particular, we employ an ensemble method combination of graph neural networks (such as GCN, GAT, GIN) with pre-trained models (such as BERT [6] and ParsBERT [7]), and graph partitioning methods for text classification of non-English datasets, in particular datasets in Persian. The process involves first organizing the data in the form of a graph, followed by generating initial node features using multi-lingual BERT or ParsBERT. Then, we utilize graph neural network algorithms such as GCN to perform learning and predictions.

According to the obtained results, the performance of ParsBERT was better than that of BERT in both scenarios of balanced and unbalanced data, for example, in the case of a two-class dataset in balanced data, when using BERT we reached an accuracy of 81 percent and when using ParsBERT we reached an accuracy of 87 percent. Also, using either BERT or ParsBERT and in both balanced and unbalanced data scenarios, when using graph neural networks such as GCN, the performance of the model increases. For example, using BERT and a GCN, we reached a 91 percent accuracy score, which shows the importance of using graph neural networks. Also, by using the ensemble learning method, the accuracy of the model increased to 93 percent in the case of unbalanced data and a two-class dataset.

## 2. Related Works

This section briefly reviews some earlier works that took place in the area of text classification tasks in general as well as some performing text classification by graph neural networks and their various types. Generally speaking, text classification tasks can be categorized into three main groups: First, *traditional text classification* methods; second, *deep learning* methods; and third, *graph neural networks*. In the following paragraphs, we present a summarized review of each one.

First, text classification dates back to the early 1960s and by was conducted by domain experts. It required specialized knowledge about how to classify textual data into predefined categories. Machine learning approaches remained popular from the late 1980s until the early 1990s.The prevailing approach during this time for automatic text classification was knowledge engineering, which relied upon logical rules manually defined for text classification and eventually became the dominant approach. These methods primarily focused on the engineering of features and the algorithms used for classification. One commonly used technique for feature engineering was the bag-of-words [8] approach, wherein each word in the data was assigned a vector representation. These vectors served as inputs for machine learning algorithms such as logistic regression.

Deep learning approaches for text classification, as one of the essential areas of application of deep learning, has gained incredible popularity in various NLP problems. This framework mainly concerns applying recurrent neural networks (RNNs) [9–12], long short-term memory (LSTM) networks [13,14], and transformer architectures for the analysis and classification of texts. RNNs are capable of preserving internal states or memory to retain information from previous inputs. This feature allows them to understand and discover relationships between words in long sequences. Having said that, RNNs have certain weaknesses (such as weak long-term dependency). LSTM has been proposed as

a solution to address these weaknesses and is commonly used for sentiment analysis, language translation, and many other natural language processing tasks, as it is capable of processing sequential data of text and capturing long-term dependencies in it. Contrary to RNNs and LSTM, a newer and stronger architecture, namely transformers, utilizes an attention mechanism that allows for bidirectional and non-local relationships between data elements. Transformers give rise to a high capability to model long-range dependencies between words and sequence elements.

Graph neural networks for text classification is the third and most important approach that we survey in this introduction and will be our main concern in the rest of the paper. In fact, one major advancement in the theory of neural networks is utilizing neural networks that can capture graph-structured data. One of the most important graph neural networks used for text classification is the graph convolutional network (GCN) [15]. The GCN generalizes the concept of convolutional neural networks (CNNs) [16] to graphs; it performs local aggregation operations on the neighbors of each node in the graph. Another algorithm is GraphSAGE [17], which employs node sampling and aggregation to create representations. This can be highly beneficial in problems that involve large graph generation. The graph attention network (GAT) [18] algorithm indeed uses the attention mechanism, which means that, during the message-passing process, it learns how to apply different weights to the neighbors of each node, meaning it assigns a different importance to each neighbor. This helps the GAT to capture complex relationships and dependencies within the graph, leading to improved performance in various graph-based tasks.

The graph isomorphism network (GIN) [19] builds upon the Weisfeiler–Lehman (WL) graph isomorphism test, a widely used algorithm for distinguishing between graphs based upon their structural properties. The WL test iteratively aggregates and hashes node labels to decide whether graphs are isomorphic. By connecting GNNs to the WL test, the isomorphism network aims to differentiate between various graph structures. This theoretical framework provides insights into the limitations and strengths of GNN variants and offers a deeper understanding of their roles in graph representation learning.

Graph convolutional networks (GCNs) have been used for text classification in some papers, such as in the research conducted by Liang et al. [20], which is among first ones to discuss the use of GCNs for text classification. Yuxiao et al. [21], based on [20], introduced a slightly different graph construction and tested various text classification benchmarks. Yuxiao et al. conducted a comprehensive analysis of the role of node embeddings in a graph and the GCN learning techniques for text classification. The authors also introduced document–document edges in their graph construction, which had not been explored in previous studies. It is worth noting that two important techniques and notions, namely one-hot vectors and BERT-generated vectors, have been employed in [20,21] in a fundamental way. The initial input word/document node features are represented as either one-hot vectors or BERT embeddings. The well-known technology of BERT is a pre-trained model, meaning that it has been trained on a large dataset of texts, which allows the model to learn general language features and patterns before being used for specific tasks.

Yuxiao et al. [22] proposed a model called BertGCN which combines large-scale pre-training with transductive learning for text classification. The model utilizes a graph structure and BERT representations to represent documents as nodes in the graph. The experimental results in their paper show the impact of the BertGCN technique on various text classification datasets.

In this work, we generally aim to draw upon the above routes of investigation on text classification by GNNs. The distinction of our work from earlier ones, however, mainly comes from our particular ways of combining ideas of ensemble learning, different GNN structures, ParsBERT and BERT frameworks, as well as algorithms such as Cluster-GCN for graph partitioning for the task of text classification. Also, we try to enhance the text classification capability on non-English datasets by means of these ideas.

## 3. Methods

The aim of this section, which also presents our main contributions, is to investigate and demonstrate various ways of combining and applying four fundamental ideas and techniques from machine learning and graph theory in the context of the text classification problem (in particular, for non-English datasets) in order to enhance the prediction task. These fundamental ideas are as follows: 1– ensemble methods, 2– pre-trained language models, 3– GNN architectures, and 4– graph partitioning. We present our methods mostly in the languages of Persian, Arabic, and English, although, as we will see, our approaches and methods will have the capacity to work beyond these particular languages. We expose our method in the following subsections. One of our important goals was to propose some suitable combinations of methods for sentiment classification of user reviews in Persian (as well as a few other languages) data using graph neural networks. For that, first some graph structures were employed by transforming our textual dataset into a graph representation. Then, using graph neural networks techniques, prediction and classification tasks were performed. Our codes are available on 2 August 2023: https://github.com/AIuniversejourney/EnhanceTextClassification. In some parts of our codes, such as the ones we employ for the combination of ParsBERT and GNN, we used the codes of the papers [21,22] but made heavy modifications to different parts of them.

### *3.1. Preprocessing*

The preprocessing stage for our textual dataset before graph construction involves several steps. It is also worth mentioning that, for non-English datasets, the pre-processing stage may be different from some technical aspects relevant to that particular language. We explain our approach here, although similar processes have been followed in earlier works such as in [21]. First, stop words are removed. These are common words that carry little to no semantic importance and their removal can enhance the performance and efficiency of natural language processing (NLP) models. Next, punctuation marks such as colons, semicolons, quotation marks, parentheses, and others are eliminated. This simplifies the text and facilitates processing by NLP models. Stemming is applied to transform words into their base or root form, ensuring standardization of the textual data. Finally, tokenization is performed, which involves dividing the text into smaller units known as tokens, which is usually achieved by splitting it into words. These pre-processing steps prepare our text for subsequent graph construction tasks. As we will see, these tokens as well as the documents themselves, will be the nodes of the graph.

### *3.2. Graph Construction*

There exist different methods for graph construction in the context of text classification. An important, known method is to construct a graph for a corpus based on word co-occurrence and document–word relations. This graph is a heterogeneous graph that includes word nodes and document nodes, allowing the modeling of global word co-occurrence and the adaptation of graph convolution techniques [20]. After constructing the graph, the next step is to create initial feature vectors for each node in the graph based upon pre-trained models, namely BERT and ParsBERT.

It is worth noting that all the unique words obtained from the pre-processing stage, along with all the documents, altogether form the set of nodes of our graph. Edges are weighted and defined to belong to one of three groups. One group consists of those edges between document–word pairs based upon word occurrence in documents. The second group consists of those edges between word–word pairs based upon word co-occurrence in the entire corpus. The third group consists of those edges between document–document pairs based upon word co-occurrence in the two documents. As shown in Equation (1), the weight $A_{i,j}$ of the edges $(i, j)$ is defined using the *term frequency-inverse document frequency (TF-IDF) [8] measure* (for document–word edges) and the *point-wise mutual information (PMI) measure* (for word–word edges). The PMI measure captures the

semantic correlation between words in the corpus. Moreover, *Jaccard similarity* is used to define the edge weight by calculating the similarity between two documents [21].

$$A_{i,j} = \begin{cases} PMI_{i,j} & i \text{ and } j \text{ are both words,} \\ TF - IDF_{i,j} & i \text{ is a doc and } j \text{ is a word,} \\ Jacard_{i,j} & i \text{ and } j \text{ are both documents,} \\ 1 & i = j, \end{cases} \tag{1}$$

It is worth noting that these measurements (TF-IDF and PMI) are normalized in the sense that we actually consider and use the normalized symmetric adjacency matrix instead of the raw matrix $A$. This method has been used in other earlier works and has also been experimentally shown to be useful.

Each document is fed to the pre-trained model multi-lingual BERT or the ParsBERT model, resulting in a numerical vector representation. For each word, a min pooling operation is applied over the BERT or ParsBERT representations of the documents containing the word. More precisely, in the same manner as that of reference [21], given a word, the min pooling over the BERT representations of all documents containing that word gives rise to the representation of that word. Now, the feature vector associated to every node is defined.

We now talk address the pre-trained models BERT and ParsBERT. These models have gained widespread popularity in text processing and natural language processing (NLP). They provide high-quality embeddings that serve as features for downstream tasks. They eliminate the need for manual feature engineering. The BERT vector, also known as BERT embedding, is in fact a high-dimensional vector that represents a text. This embedding is created by encoding words in a sentence using a pre-trained BERT model trained on a large dataset of textual data. The encoding process generates a fixed-length vector for each word in the sentence, which can then serve as input for other natural language processing models. The ParsBERT model, similarly, was pre-trained from diverse sources of Persian textual data.

It is worth noting that, beside the BERT frameworks, there have been many other popular frameworks for finding representations of texts such as "word mover's embedding", "Word2vec", etc. BERT is a language model widely used for a variety of natural language processing tasks, in particular tasks that require an understanding of the context, while some other frameworks, such as the mentioned word mover's embedding, focuses on word alignment and is useful for tasks involving semantic similarity or semantic distance. As previously mentioned, in this work we use BERT and ParsBERT frameworks to obtain initial representations of documents/words; then, as we will explain below, we use GNN structures to turn these representations into an even richer one by capturing the underlying graph structure associated with the data.

Once the text graph is constructed, it is fed to a two-layer GCN. Each GCN layer performs message passing among the nodes based upon their neighborhood relationships; this allows for the integration of information from larger neighborhoods. We review the message passing step more in detail. As previously mentioned, the initial representation vectors made by BERT and ParsBERT are given as the initial features of the nodes to the input of the graph convolutional neural network (GCN). The information of these nodes is conveyed by the process of messages passing through graph neural networks in such a way that each node in the graph computes a message for each of its neighbors. Messages are in fact a function of nodes, neighbors, and the edges between them. Messages are sent and each node aggregates the messages it receives using a function such as sum or average. After receiving messages, each node updates its attributes as a function of its current attributes and the aggregated messages. The basic GNN message passing formula is defined as follows:

$$h_u^{(k)} = \sigma(W_{self}^{(k)} h_u^{(k-1)} + W_{neigh}^{(k)} \sum_{v \epsilon N(u)} h_v^{(k-1)} + b^{(k)}) \tag{2}$$

where $W_{self}^{(k)}, W_{neigh}^{(k)} \epsilon R^{d^{(k)} * d^{(k-1)}}$ are trainable parameter matrices and $\sigma$ denotes an element-wise non-linearity (e.g., ReLU). The bias term b $\epsilon$ $R^{d^{(k)}}$ is also the bias term. The final feature vectors obtained for each text data node are considered as the output of the GCN, which are passed through a SoftMax classifier for final prediction. By jointly training the BERT and GCN modules, we leverage the advantages of both pre-trained models and graph models.

We now address the ideas behind GCNs and how they help us in our text classification. GCNs are designed to efficiently capture semantic relationships and rich dependencies among the nodes of a graph, enabling a better understanding and representation of text content. Text classification often requires considering textual information adjacent to words or phrases for accurate predictions. GCNs can collect information from neighboring nodes in the graph, efficiently gathering and disseminating contextual information. This capability allows GCNs to take advantage of the local context of each node and make informed decisions regarding text classification problems. On the other hand, one of the challenges of text classification is addressing inputs of different lengths, such as sentences and phrases with varying numbers of words. GCNs can naturally handle variable-length inputs using the graph structure. By leveraging the graph structure, GCNs can depict relationships and dependencies between words or sentences, which provides a more robust and flexible approach for text classification. Additionally, GCNs excel in modeling global dependencies in a graph. In text classification, global dependencies refer to dependencies that encompass the entire dataset and the text. By propagating information throughout the graph, GCNs can capture these global dependencies, enabling a comprehensive understanding of textual data and their classification. According to our obtained results, the performance of most of our combinations of models improved, both for balanced data and unbalanced data, when using graph neural networks, and in particular GCNs. This improvement can be attributed to the advantages of using the graph neural networks mentioned above, as they enhance the quality of the representations and the robustness of text classification models.

### *3.3. Graph Partitioning*

We now address another ingredient of our combination of techniques, which posses both conceptual as well as technical significance in our results. A fundamental challenge in graph neural networks is the need for a large space to store the graph and the representation vectors created for each node. To address this issue, one of our contributions is to apply Cluster-GCN algorithm [23], which partitions the graph into smaller clusters, as will be explained below. The Cluster-GCN algorithm utilizes a graph-clustering structure to address the challenge posed by large-scale graph neural networks. In order to overcome the need for extensive memory and storage for the graph and its node representation vectors, the algorithm divides the graph into smaller subsets by using graph-clustering algorithms like METIS [24]. METIS aims to partition the graph into subgraphs of approximately equal size while minimizing the edge connections between them. The process involves graph coarsening, wherein vertices in the original graph are merged to create a smaller yet representative graph for efficient partitioning. After generating the initial subgraphs using the graph partitioning algorithm, the algorithm refines the partitioning in a recursive manner by the application of an uncoarsening algorithm. This recursive process propagates partitioning information from smaller to larger levels while maintaining the balance and size of the subgraphs. By dividing the graph into smaller clusters, the model's performance improves in terms of computational space and time. The decision to employ graph-clustering methods is driven by the aim to create partitions wherein connections within each group are strong, capturing the clustering and community structure of the graph effectively. This approach is particularly beneficial in node embeddings, as nodes and their neighbors usually belong to the same cluster; it enables efficient batch processing.

### *3.4. Ensemble Learning*

One of the other ingredients of our combination of techniques is using ideas from the theory of ensemble learning. Bagging and stacking are important techniques in neural

networks. As another contribution in this paper, techniques of ensemble learning types are employed. Indeed, our ensemble learning method uses different combinations of GNNs and (Pars) BERT to obtain fine results in language-specific text classification tasks. Next, we will explain this in more detail.

After the preprocessing stage in the dataset, we construct the graph in the way that was described above. Then, using the Cluster-GCN algorithm, we divide the input graph into four disjoint-induced subgraphs. Each of these subgraphs is then fed to a separate graph neural network. According to our experiences, graph convolutional networks (GCNs) have shown better performance in text classification processing in many of our attempts. Therefore, we highlight the use of GCNs in our combination. In addition to GCNs, we also utilize a graph isomorphism networks (GINs) framework as well as graph attention networks (GATs) as two other parts of the ensemble learning in our combination. It is worth noting that the GIN part in our combination here intends to capture the global structure of the graph. An overview of the algorithm used is shown in Figure 1. Once training was conducted over separated individual models, after which we obtained four different trained GNN models. In the test stage, test samples pass through all these models and each one creates its own classification output vector. Then, the results of all of these models are combined by taking the average of those output vectors. Via this process, the ensemble method can help to enhance the prediction accuracy by amalgamating the strengths of multiple models.

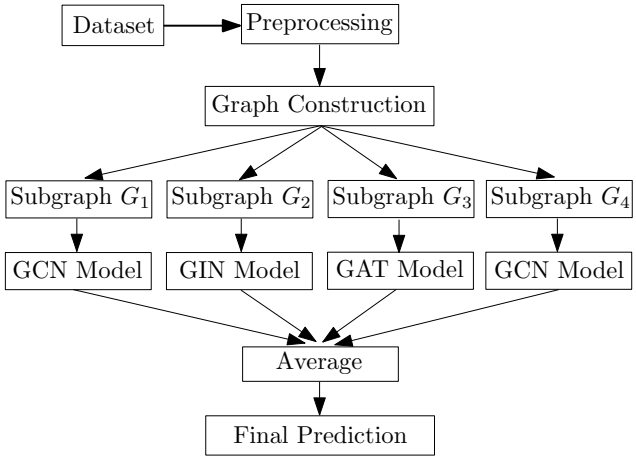


**Figure 1.** Using group algorithms in graph neural networks.

As another part of our combination, we employ frameworks of ParsBERT (instead of BERT) when addressing Persian or Arabic datasets in our method to obtain the initial representations of the words and documents as nodes of our graph, as mentioned earlier. BERT is a known language model which associates a vector to a text. ParsBERT is a specified version of BERT which is fine-tuned for text classification in the Persian language. Both BERT and ParsBERT create the initial feature vector of the nodes in the learning tasks. We employed both in our different experiments but with an emphasis on ParsBERT, and later we will provide a report of the results as well as an explanation of the idea behind emphasizing the use of ParsBERT. After obtaining an initial representation from ParsBERT, the different mentioned GNN methods started to operate on them and, eventually, an averaging of the results produced our final classification results.

Through this classification process, we made several observations. First, we observed how applying GNN models achieves better scores on the task of text classification by better capturing the topological information between users and their opinions. Second, we observed that the performance of ParsBERT had been better than that of BERT in both balanced and imbalanced data scenarios. This can be attributed to the nature of our data being in the Persian language (or even similar non-English languages such as Arabic). Since ParsBERT was trained on a large collection of Persian texts, it allows ParsBERT to

learn better textual representation vectors for Persian text compared to the original BERT. This results in significantly higher encoding power in the beginning of the process of generating representations, which eventually leads to better ultimate representation after passing through the several layers and steps of operations of the GNN. Eventually, this results in a better final performance. This observation suggests and emphasizes using the language-specific pre-trained language model (like ParsBERT, instead of BERT) for obtaining better initial and final representations in the context of non-English language classification problems.

In the next section, we will detail different experiments we conducted based on the various ways of combining the four main techniques mentioned above, namely ensemble methods, pre-trained language models, GNN architectures, and graph partitioning. For instance, in one experiment, we tested a model that combined the ensemble technique and three different architectures of GCN, GIN, and GAT, as well as BERT. In another experiment, we omitted the ensemble technique and solely used a combination of GCN and ParsBERT. We will examine several of such combinations and compare the results. This gives compelling insights into how a combination of different techniques impacts the final classification results.

## 4. Datasets

**Digikala:** (https://github.com/AIuniversejourney/EnhanceTextClassification/blob/main/DATASET/digikala_data.csv accessed on 2 August 2023). This dataset contains user comments in Persian on various products from a store website called Digikala, which is among the largest online shops in the Persian language. The website offers a wide range (hundreds of thousands) of products, including electronic products, books, clothes, and more. Users have the opportunity to express their opinions about the products they have purchased. The dataset consists of 100,000 rows and 12 different columns, containing various forms of information such as user opinions, product advantages and disadvantages, number of likes and dislikes, product IDs, and more. The characteristics of this dataset are presented in Table 1. Each row in this dataset is related to a user's opinion about a product. Here, we removed the comments with missing labels, and in the end, about 63,000 comments remained with their labels.

**Table 1.** Digikala dataset.

| Digikala | Recommended | No-Idea | Not-Recommended | Total |
|---|---|---|---|---|
| Label | 1 | 0 | −1 | - |
| Number of Data | 36,960 | 10,528 | 16,098 | **63,586** |

**Arabic-twitter-corpus-AJGT:** (https://github.com/komari6/Arabic-twitter-corpus-AJGT 2 August 2023). This dataset is an Arabic Jordanian General Tweets (AJGT) dataset, which contains 1800 tweets in Arabic, which are classified into positive and negative categories. The dataset is balanced; 900 tweets are classified in the positive class and 900 tweets are classified in the negative class.

**DeepSentiPers:** (https://github.com/JoyeBright/DeepSentiPers 2 August 2023). This is a Persian, balanced dataset that includes users' opinions on digital products, classified into five categories. The details of this dataset are presented in Table 2.

**Table 2.** DeepSentiPers dataset.

| DeepSentiPers | Delighted | Happy | Natural | Angry | Furious | Total |
|---|---|---|---|---|---|---|
| Label | −2 | −1 | 0 | 1 | 2 | - |
| Number of Data | 1342 | 2184 | 3152 | 697 | 40 | **7415** |

**MR:** (http://www.cs.cornell.edu/people/pabo/movie-review-data/ 2 August 2023). This dataset is used for binary sentiment analysis, which includes user comments in English about different movies. These comments are classified into two categories: positive and negative; 5331 data are found in the positive class and 5331 data are found in the negative class.

## 5. Experimental Results

We conducted a series of diverse experiments that explored the combinations of four primary techniques: ensemble methods, pre-trained language models, graph neural network architectures, and graph partitioning. Our objective was to investigate how these techniques can be synergistically combined to enhance the performance of classification tasks. One set of experiments involved creating ensemble models, wherein we integrated three distinct graph neural network architectures (GCN, GIN, and GAT) along with the utilization of BERT as a pre-trained language model. Through these ensemble models, we aimed to showcase the potential improvements achieved by combining these complementary techniques. In addition to the ensemble approach, we undertook another experiment wherein we focused on a specific combination, specifically using GCN and ParsBERT together without the ensemble technique. This allowed us to analyze the performance of this simplified pairing and assess its impact on the classification results. Throughout our investigation, we explored several such combinations, thoroughly evaluating their respective outcomes. Ultimately, by comparing the results of these different techniques and combinations, we gained valuable insights into how the blending of various approaches influences the final classification results. These findings provide us with a deeper understanding of how to leverage the strengths of each technique effectively, paving the way for more informed decision-making in real-world applications.

### *5.1. Experiment 1: Applying ParsBERT, BERT, and GCN on Our Dataset*

Under two scenarios, we examined the performance of our proposed models. The first scenario involved a two-class dataset with two labels: "recommended" and "not-recommended". The second scenario dealt with a three-class dataset which had three labels: "recommended", "not-recommended", and "no-idea". For both scenarios, we evaluated the model's performance and accuracy in two settings: one with unbalanced data and the other with balanced data. The results of these experiments are presented in Table 3. For every case in which the dasaset was imbalanced, we also calculated $F_1$, precision, and recall along with the accuracy. In the three-class dataset, 51,961 data points were allocated for training, while 5774 data points were designated for testing. The number of extracted word was 44,758, so the number of nodes was 102,493. For the two-class dataset, 43,149 data points were assigned to training, while 4795 data points were reserved for testing. The number of extracted words was 40,304, so the number of nodes was 88,248.

As is made evident from the results in Table 3, the performance of ParsBERT was better than BERT in both balanced and imbalanced data scenarios. The key insight here is that this enhancement can be attributed to the nature of our data being in the Persian language. Since ParsBERT was trained on a large collection of Persian texts, this allowed for ParsBERT to learn better textual representation vectors for Persian text compared to BERT, resulting in a significantly higher accuracy. In both cases of using BERT and ParsBERT, and in both balanced and imbalanced data scenarios, when employing graph neural networks such as GCNs, the model's performance increases; this is due to the advantage of utilizing graph neural networks.

Regarding the statistics of the above models, we repeated the above experiments several times, each time with a different model weights initialization. As mentioned in Table 3, for example, the mean of the results (accuracies) of the strongest model in the above table, namely ParsBERT+GCN, is 91.1 (on 2Class) with a standard deviation of less than 0.06. For the BERT+GCN model (on 2Class), the standard deviation is just marginally higher (0.08) and the mean is also slightly (around 0.5%) lower than the ParsBERT+GCN

model, which can again be due to the fact that the representations of ParsBERT in the Persian language are marginally better than BERT's.

**Table 3.** Accuracy (%) of the proposed model on the Digikala dataset (mean ± standard deviation for the ParsBERT+GCN and BERT+GCN experiments).

| Model | Digikala | | | |
|---|---|---|---|---|
| | 2Class | | 3Class | |
| | Balance | Imbalance | Balance | Imbalance |
| ParsBERT | 68 | Accuracy = 87<br>$F_1$ = 72<br>Precision = 69<br>Recall = 80 | 57 | Accuracy = 62<br>$F_1$ = 55<br>Precision = 52<br>Recall = 54 |
| ParsBERT+GCN | 70 | Accuracy = 91.1 ± 0.06<br>$F_1$ = 74<br>Precision = 71<br>Recall = 80 | 58 | Accuracy = 63.9 ± 0.09<br>$F_1$ = 55<br>Precision = 52<br>Recall = 59 |
| BERT | 57 | Accuracy = 81<br>$F_1$ = 68<br>Precision = 66<br>Recall = 74 | 54 | Accuracy = 56<br>$F_1$ = 54<br>Precision = 51<br>Recall = 55 |
| BERT+GCN | 57 | Accuracy = 90.6 ± 0.08<br>$F_1$ = 71<br>Precision = 69<br>Recall = 80 | 55 | Accuracy = 57.2 ± 0.15<br>$F_1$ = 53<br>Precision = 51<br>Recall = 55 |

### *5.2. Experiment 2: Ensemble Learning via Combining Different GNN Structures*

As seen in Table 4, in all cases of using ensemble learning techniques, the accuracy and performance of the model increases. According to Tables 3 and 4, we highlight the following: In the Digikala-2Class dataset, we reached a 91% accuracy using Pars(Bert)+GCN in imbalanced data by using the ensemble learning technique, and this accuracy increased to 93% thanks to ensemble learning. In the Digikala-3Class dataset, we reached about 64% of accuracy using ParsBert+GCN and the ensemble learning technique, and this accuracy reached about 68% thanks to ensemble learning.

**Table 4.** Accuracy (%) of ensemble learning techniques (mean ± standard deviation for the last row).

| Model (Graph) | Digikala | | | |
|---|---|---|---|---|
| | 2Class | | 3Class | |
| | Balance | Imbalance | Balance | Imbalance |
| GCN (G1) | 66 | 89 | 54 | 64 |
| GAT (G2) | 64 | 81 | 48 | 59 |
| GIN (G3) | 63 | 78 | 51 | 55 |
| GCN (G4) | 64 | 86 | 53 | 67 |
| Ensemble Learning | 69 | Accuracy = 93.2 ± 0.02<br>$F_1$ = 77<br>Precision = 78<br>Recall = 79 | 58 | Accuracy = 68.4 ± 0.09<br>$F_1$ = 58<br>Precision = 52<br>Recall = 60 |

It is worth noting that, in the three-class dataset, 50,331 data points were allocated for training, while 12,583 data points were designated for testing. The number of extracted words was 7448, so the number of nodes was 70,362. For the two-class dataset,

42,446 data points were assigned to training, while 10,612 data points were reserved for testing. The number of extracted words was 4129, so the number of nodes was 57,187.

It is worth recalling that, in this experiment, the size of the training sets associated to the rows $G_1, \ldots, G_4$ in the table are smaller than the size of the training set of the ParsBERT+GCN experiment in Table 3, since in those cases training was conducted on the mentioned subgraphs.

We now briefly address the statistics of our above-mentioned ensemble model. In order to obtain a better understanding of the model, we repeated the above experiments and ran our model several times, each time with a different model weights initialization, and obtained the accuracy and other parameters of every repeated experiment to see how robust the results were. As mentioned in Table 4, the mean of the results (accuracies) is 93.2 with a standard deviation of less than 0.02 on the 2Class dataset. It is worth to mention that, as is known in the area of ensemble learning, the ensemble learning methods and models possess the advantage of tending to have high robustness.

Ensemble learning can significantly improve the prediction accuracy compared to using a single model. This technique, by combining diverse models, effectively captures different aspects of the data and aims to reduce model bias and errors. Additionally, since this technique involves multiple models trained on different subsets of the data and utilizes different algorithms, it is less affected by outliers or noise in the data, enabling the creation of more accurate and robust predictions. It is important to note that ensemble learning also has potential drawbacks, such as increased computational complexity, longer training time, and the need for more resources. Moreover, the effectiveness of the employed methods depends on the diversity and quality of each model. However, the advantages of ensemble learning make it a powerful technique for improving prediction accuracy and generalization in various machine learning scenarios.

Comparing the results of the ensemble method proposed here from Table 4 with the results of Table 3 (last two rows) shows that this method gives better results than applying only BERT or BERT+GCN, which was also considered in [21,22] (in the context of English language). It is worth mentioning that other frameworks such as dependency grammar-based rules, LSTM, and CNN have been considered for sentiment analysis on Persian datasets in papers such as [25,26]. However, to the best of our knowledge, before the present paper there was no other work on the sentiment analysis of Persian texts using a combination of ensemble learning methods with GNN structures.

### *5.3. Brief Report of Some Additional Experiments*

Here, we proceed by doing conducting additional experiments on a number of additional datasets in the same line as before to better understand the difference between accuracy in English and non-English datasets when BERT+GCN is used. On the Arabic-twitter-corpus-AJGT dataset, we experimented with BERT and GCN. We believe that taking some further steps in the pre-processing stage can lead to a better accuracy in this dataset. However, this is out of the scope of the present paper and we leave further investigation on it to future work. We report that, by using BERT and GCN, we reached a 98% accuracy in train data and a 83% accuracy in test data.

Similarly, the number of data instances in the MR dataset (an English dataset) is much smaller than that in the Digikala dataset (in Persian), yet significantly better results were achieved compared to the Digikala dataset. In fact, BERT+GCN has poor accuracy when operating on Digikala-2Class compared to the MR dataset. Despite having more data instances, the accuracy obtained on the Digikala dataset was lower than the accuracy achieved on the MR dataset (it is worth mentioning here that, in a limited experiment on the DeepSentiPers dataset using ParsBERT, the result improved compared to when using BERT). These phenomena of differences in the results on English and Persian datasets can be attributed to the strength of BERT in English language data that we used to obtain the features of the nodes in the graph. As mentioned, BERT models have been trained on larger datasets, including the entire English Wikipedia and a vast collection of English books,

whereas BERT models for Persian, e.g., ParsBERT, may have been trained on smaller-scale datasets. This extensive pre-training allows English BERT to capture a wide range of language patterns and semantic relationships, making it a powerful model for English text processing. On the other hand, tokenization, the process of dividing text into smaller units such as words, differs between English and Persian due to the linguistic variations. In English, words are usually separated by spaces, which makes tokenization relatively straightforward. In Persian, words are connected without clear spacing, making the tokenization process more challenging. The summary of the experiments on AJGT and MR is reported below.

As was mentioned before, the MR and AJGT datasets are both balanced and the number of positive and negative comments in them is equal. In the above experiments, we allocated 85% of the data points for training and the rest for testing. Similar to our previous experiments in the previous sections, we repeated the above experiments several times, each time with a different model weights initialization. As mentioned in Table 5, the means of the results (accuracies) were 86.3 and 82.8 on the MR dataset and the AJGT dataset, respectively. Moreover, the standard deviations of the results were less than 0.1 and 0.5 on the MR dataset and the AJGT dataset, respectively. We recall that both the MR and AJGT datasets are smaller than the Digikala dataset.

**Table 5.** Summary of experiments on AJGT and MR.

| | **Accuracy on Test Data** | **Accuracy on Train Data** |
|---|---|---|
| AJGT | $82.8 \pm 0.5$ | 98 |
| MR | $86.3 \pm 0.1$ | 97 |

## 6. Conclusions and Further Works

In this paper, we investigated various ways of combining and applying some fundamental ideas and techniques from machine learning and graph theory, namely ensemble methods, pre-trained language models, GNN architectures, and graph partitioning, in the context of the text classification problem (in particular for non-English datasets) in order to tackle the prediction task and enhance the results. We tested our ideas on concrete problems such as the sentiment classification of user reviews in Persian (as well as a few other languages).

As future work, one can elaborate ideas of combinations of methods and consider a wider range of techniques of combination. Moreover, as another direction, we can enrich our ensemble techniques by incorporating other sophisticated GNN architectures. As another direction, one can utilize a broader range of pre-trained language models, as in this study, our focus was primarily focused on BERT and ParsBERT. The are numerous other favorable candidates that can be taken into account.

We now address some limitations of our work and some suggestions regarding them. One of the limitations of the method of ensemble learning is the relatively large amount of computational power required for training datasets. Moreover, many major machine learning methods such as transductive methods, besides their many advantages, have the common characteristic that they cannot be easily adapted to new out-of-sample testing data. However, there are ideas concerning this fundamental limitation of such methods (see for example [27]). An idea for a further work here could be considering the integration of such ideas and the methods we used in this paper in order to make models more easily adaptable once new nodes are added to the constructed graph structures.

**Author Contributions:** Methodology, F.G., Z.R., A.M. and M.A.; Software, F.G., Z.R. and A.M.; Validation, Z.R. and A.M.; Investigation, M.A.; Writing—original draft, F.G., Z.R. and A.M.; Writing—review & editing, Z.R., A.M. and M.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** Our codes are available at the following link: https://github.com/AIuniversejourney/EnhanceTextClassification accessed on 2 August 2023.

**Acknowledgments:** The author Alireza Mofidi is indebted to Institute for Research in Fundamental Sciences, IPM, for support. His research in this paper was in part supported by a grant from IPM (No.1400030117).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Li, Q.; Peng, H.; Li, J.; Xia, C.; Yang, R.; Sun, L.; Yu, P.S.; He, L. A Survey on Text Classification: From Traditional to Deep Learning. *ACM Trans. Intell. Syst. Technol.* **2022**, *13*, 1–41. [CrossRef]
2. Zhang, L.; Wang, S.; Liu, B. Deep learning for sentiment analysis: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Dis.* **2018**, *8*, e1253. [CrossRef]
3. Aggarwal, C.C.; Zhai, C. A survey of text classification algorithms. In *Mining Text Data*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 163–222.
4. Zeng, Z.; Deng, Y.; Li, X.; Naumann, T.; Luo, Y. Natural language processing for ehr-based computational phenotyping. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2018**, *16*, 139–153. [CrossRef] [PubMed]
5. Dai, Y.; Liu, J.; Ren, X.; Xu, Z. Adversarial training based multi-source unsupervised domain adaptation for sentiment analysis. In Proceedings of the AAAI Conference on Artificial Intelligence 2020, New York, NY, USA, 7–12 February 2020; pp. 7618–7625.
6. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the NAACL-HLT, Minneapolis, MN, USA, 3–5 June 2019; pp. 4171–4186.
7. Farahani, M.; Gharachorloo, M.; Farahani, M.; Manthouri, M. Parsbert: Transformer-based model for persian language understanding. *Neural Process. Lett.* **2021**, *53*, 3831–3847. [CrossRef]
8. Salton, G.; Buckley, C. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.* **1988**, *24*, 513-523. [CrossRef]
9. Medsker, L.R.; Jain, L.C. *Recurrent Neural Networks: Design and Applications*; CRC Press: Boca Raton, FL, USA, 1999.
10. Liu, P.; Qiu, X.; Huang, X. Recurrent neural network for text classification with multi-task learning. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI), New York, NY, USA, 9–15 July 2016; AAAI Press: Washington, DO, USA, 2016; pp. 2873–2879.
11. Luo, Y. Recurrent neural networks for classifying relations in clinical notes. *J. Biomed. Inform.* **2017**, *72*, 85–95. [CrossRef]
12. Lai, S.; Xu, L.; Liu, K.; Zhao, J. Recurrent convolutional neural networks for text classification. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
13. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
14. Tai, K.S.; Socher, R.; Manning, C.D. Improved semantic representations from tree-structured long short-term memory networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Beijing, China, 26–31 July 2015.
15. Zhang, S.; Tong, H.; Xu, J.; Maciejewski, R. Graph convolutional networks: A comprehensive review. *Comput. Soc. Netw.* **2019**, *6*, 11. [CrossRef]
16. Kim, Y. Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014.
17. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1–11.
18. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph Attention Networks. *Stat* **2018**, *1050*, 4.
19. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How Powerful are Graph Neural Networks? In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
20. Yao, L.; Mao, C.; Luo, Y. Graph convolutional networks for text classification. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27–28 January 2019; Volume 33, pp. 7370–7377.
21. Han, S.C.; Yuan, Z.; Wang, K.; Long, S.; Poon, J. Understanding Graph Convolutional Networks for Text Classification. *arXiv* **2022**, arXiv:2203.16060.
22. Lin, Y.; Meng, Y.; Sun, X.; Han, Q.; Kuang, K.; Li, J.; Wu, F. BertGCN: Transductive Text Classification by Combining GNN and BERT. In Proceedings of the Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, Online, 1–6 August 2021; pp. 1456–1462.
23. Chiang, W.-L.; Liu, X.; Si, S.; Li, Y.; Bengio, S.; Hsieh, C. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 257–266.
24. Karypis, G.; Kumar, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.* **1998**, *20*, 359–392. [CrossRef]
25. Dashtipour, K.; Gogateb, M.; Lia, J.; Jiangc, F.; Kongc, B.; Hussain, A. A Hybrid Persian Sentiment Analysis Framework: Integrating Dependency Grammar Based Rules and Deep Neural Networks. *Neurocomputing* **2020**, *380*, 1–10. [CrossRef]

26. Ghasemi, R., Ashrafi A., S.A., Momtazi, S. Deep Persian sentiment analysis: Cross-lingual training for low-resource languages. *J. Inf. Sci.* **2022**, *48*, 449–462. [CrossRef]
27. Dai, Y.; Shou, L.; Gong, M.; Xia, X.; Kang, Z.; Xu, Z.; Jiang, D. Graph fusion network for text classification. *Knowl.-Based Syst.* **2022**, *236*, 107659. [CrossRef]