IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 March 2023, accepted 8 April 2023, date of publication 24 April 2023, date of current version 28 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3269574

RESEARCH ARTICLE

Enhancing the Performance of WSD Task Using Regularized GNNs With Semantic Diffusion

BİLGE ŞİPAL SERT[1], EREN ELMA[2], AND AYŞE BERNA ALTINEL[3]
[1]Afiniti AI R&D, 34752 İstanbul, Turkey
[2]Department of Computer Engineering, Faculty of Engineering, Istanbul University—Cerrahpaşa, 34320 İstanbul, Turkey
[3]Department of Computer Engineering, Faculty of Technology, Marmara University, 34722 İstanbul, Turkey

Corresponding author: Ayşe Berna Altınel (berna.altinel@marmara.edu.tr)

This work was supported in part by the Scientific and Technological Research Council of Turkey (TÜBİTAK) with grant number 120E187.

ABSTRACT The ambiguity of a polysemous word, regardless of its language, is referred to as the ability to have more than one meaning. Many techniques from Word Sense Disambiguation (WSD) have been used to clarify the ambiguity. With these techniques, the correct sense of a word according to its particular context can be determined. In English, there exist many ambiguous words. For example, the word *cell* may have different meanings based on its context. For instance, it may mean *the basic structural unit of all organisms* in *biology domains*, or it may mean *a device that delivers an electric current as the result of a chemical reaction* in *textual materials about technology or electricity*, or it completely means another different thing: *a room where a prisoner is kept* in *textual materials about prison*. In this study, to enhance the performance of the WSD task, we develop a methodology with Graph Convolutional Neural Networks (GCN), including normalization, thresholding, and regularization modules. We attempt to improve the traditional Text GCN algorithm with a semantic diffusion process, which increases the classification performance of the WSD task. As far as we know, there is no work for such a comprehensive classifier for the WSD task in English. To show the effect of the suggested model, we performed experiments on the SensEval dataset, which is a very popular dataset and benchmark in the WSD domain. The experiment results show that the regularization effect and diffusion process in GCN and Text GCN architectures are powerful strategies for the WSD task.

INDEX TERMS Diffusion algorithm, graph convolutional neural networks, graph neural networks, graph regularization, word sense disambiguation.

I. INTRODUCTION

In natural languages, some words often can have different meanings (senses) depending on the context of their usages in various sentences. These types of words are called polysemous and exist in many languages. For instance, according to WordNet,[1] there are 40 different senses of the English word head [1]. Some senses of the word head are as follows: 1) the part of the body above the neck where the eyes, nose, mouth, ears, and brain are, 2) a person or animal when considered as a unit, 3) a measure of length or height equal to the size of a head, 4) the mind and mental abilities, 5) someone in charge of or leading an organization, group, 6) the top part or beginning of something. Polysemous words lead to uncertainty in the context of their occurrence due to having two or more meanings depending on the circumstances. Consequently, disambiguation is a challenging problem that must be handled meticulously in many tasks related to natural languages.

In Natural Language Processing (NLP), understanding the correct sense of a word is necessary for many processes. For example, Part-of-Speech (PoS) tagging is a process of NLP. In PoS tagging, each word of the sentence is labeled with its appropriate part of speech, such as a verb, noun, or adverb. To properly perform PoS tagging, the exact sense of the word must be known. In PoS tagging, the ambiguity problem should be resolved to at least an acceptable degree to achieve high performance. Another example is text translation, which

The associate editor coordinating the review of this manuscript and approving it for publication was Giacomo Fiumara.

[1]https://wordnet.princeton.edu/

is the process of converting the language of the text to another language without changing the content and meaning. To properly convert a language to another, ambiguity must be handled carefully since many polysemous words exist in many languages. Also, in other NLP tasks such as text summarization, information retrieval, topic classification, and sentiment polarity detection, ambiguity still can be a problem due to the characteristics of natural languages; thus, Word Sense Disambiguation (WSD) holds a vital role in NLP from past to present [2], [3], [4].

There are two types of ambiguity; these are lexical and syntactic ambiguity [2]. Lexical ambiguity has two or more meanings within a single word, and the syntactic meaning has two or more meanings within a sentence or sequence of words. Practically in NLP, syntactic ambiguity is harder to detect compared to lexical ambiguity.

Various research has been done to solve the problem of detecting ambiguity. These approaches can be categorized roughly as dictionary-based methods [5], [6], [7], [8], [9] and machine learning methods [10], [11], [12], [13], [14], [15], [16]. Dictionary-based methods habit to using an external resource or corpus (i.e., WordNet[1], BalkaNet [17], BabelNet [18], IMS [19], Wikipedia, Twitter, etc.) to get the capability of different semantic meanings of words. In WSD, supervised machine learning methods and deep learning algorithms are usually preferred over dictionary-based methods due to having better performance.

In the last years, graph-based neural networks and diffusion-based models have been highly published [20], [21], specifically in image processing [21], [22], [23]. Besides these achievements, recent NLP studies also aim to utilize graph-based neural network algorithms such as Graph Convolutional Networks (GCN) and Graph Attention Networks (GAT) [14], [15], [24] to accomplish WSD tasks as well. However, compared with the usage of these algorithms and their variants in image processing, these efforts are in the beginning phase for WSD.

In this study, we want to further these attempts by merging the semantic diffusion kernel approach [25], [26], [27], [28] with graph-based neural network algorithms to develop a methodology for enhancing the performance of the WSD task. The success of previous NLP studies on the diffusion process as a semantic kernel indicates this process is superior in distinguishing the correct meaning compared to other kernel-based methods [26], [27], [29]. The success of the diffusion-based image-processing graph algorithms and semantic diffusion kernels inspires our methodology. We aim to achieve a similar success of diffusion-based graph algorithms for finding the correct sense of English polysemous words in textual data. The main challenge is to create a graph-based algorithm that utilizes a diffusion process on textual data to accomplish the WSD task.

We propose two different and novel strategies to use the diffusion process with graph algorithms GCN and GAT. Our first strategy is inspired by the Text GCN algorithm [24]. We replace the Pointwise Mutual Information (PMI) based matrix with a semantic diffusion matrix by aiming to capture the semantic similarities within words better than PMI.

The second strategy is very similar to [20] but differs in adjacency matrix creation for the textual data. Our strategies include several test schemes with normalization and graph regularization to observe the direct effect of these techniques. We conduct many experiments on the benchmark dataset: Interest, Line, Hard, and Serve from SensEval [30]. According to the experimental results, our technique is superior in the sense of accuracy metrics. Moreover, we achieve this superiority with a smaller training matrix.

Our contributions can be summarized as follows:

- The primary purpose of this study is to show the effectiveness of a methodology including semantic diffusion algorithms and Graph Neural Networks (GNN) with specialized regularization thresholding (sparsification [20]) and normalization techniques. Based on our literature search, there needs to be more effort for such a comprehensive model construction for the WSD task in English.
- In the Text GCN algorithm, a large and heterogeneous text graph is used [24]. We improve the traditional Text GCN algorithm with a semantic diffusion process which contributes to the literature due to its unique property of being the first of its kind.
- We aim to replace the convolution process on words of the Text GCN algorithm and mimic the process by the diffusion process. In the simplest sense, the semantic diffusion process does a weight propagation on the graph of the words and creates a similarity score between each word. One of the convolution layers of Text GCN precisely focuses on doing this process as well, and this observation is the starting point of this study.
- We propose a novel way of utilizing textual data in diffusion-based graph algorithms [20]. We created a semantic diffusion matrix and obtained an adjacency matrix within documents. We applied the graph convolution to this adjacency matrix called Semantic Diffusion GCN.

According to the experimental results, our technique is superior in several metrics. Moreover, we achieve this superiority by Semantic Diffusion GCN with a smaller training matrix than the Text GCN algorithm.

We organize the remainder of the paper as follows; In Section II, the background and we present selected related work from past to present. In Section III, we explain the suggested methodology. In Section IV, we present experimental results and corresponding discussions. Lastly, in Section V, we explain the conclusion and the future directions of this study.

II. BACKGROUND AND RELATED WORK

A. WORD SENSE DISAMBIGUATION

WSD is a classification task that tries to identify the most appropriate sense of a polysemous word between various possible senses specified with a lexicon depending on the

context. Two types of WSD exist. The first is the Lexical WSD. In Lexical WSD, there are a couple of pre-selected sorts of target words. Each word has multiple senses. Supervised machine learning algorithms are usually used for finding the correct sense of every word. Second type of WSD problem is called the All-words task. The critical distinction between the Lexical and the All-words task is, in the All-words task, the disambiguation of whole sorts of words is intended (i.e., adverb, noun, verb, adjective) throughout the whole corpus. In general, wide-coverage systems are required for this task. This work is dedicated to finding the correct sense of the words given.

B. RELATED WORK

In this section, different studies from the literature from past to present are summarized below, which focus on the WSD problem that mainly uses the datasets under the Senseval and SemEval tasks, which are benchmarks for this domain.

In the study by Yang et al. [31], a WSD model is proposed based on a sequence topic model that uses sense dependency. In the proposed model, to determine the true sense of a word, unlike the usual WSD approaches that only use the information of context, this proposed model includes the global sense distribution in addition to contextual sense dependency. As an approach to the proposed model, the correct meaning of the targeted word depends upon the sense of the word which comes before that word. This dependence is presented using the Markov chain assumption. According to the proposed approach, the methodology can be rule-based and unsupervised. As the dataset, the model mainly uses the ''Senseval'' dataset, which is highly used for WSD models. Specifically, five different datasets are used. These are, SemEval-2007, SemEval-2013, Senseval-2, Senseval-3, and SemEval-2015. Experiments were done to observe the performance of the model. According to this evaluation, when all datasets are included, the model achieves an overall 66.9% of the F1 score. This evaluation shows that the proposed model's performance is low compared to some WSD models, but it achieves the performance of other state-of-the-art knowledge-based approaches.

In the study by Sousa et al. [32], an evaluation study for WSD is made based on semi-supervised approaches with word embeddings. For this purpose, multiple datasets were used. These datasets are, Senseval-2, Senseval-3, and Semeval-2007. The primary purpose of this study is to adapt semi-supervised algorithms for WSD using word embeddings such as Word2Vec, FastText, and BERT models. An evaluation of four different graph-based semi-supervised WSD models is conducted. In the first step of the methodology, preprocessing steps are applied to the datasets. In the preprocessing steps, the datasets are tokenized, lemmatized, removed from the stop words, and PoS-tagged. Embeddings Word2Vec, FastText, and BERT were used with PoS tags of words to represent WSD. The graphs are constructed, and different distance measure algorithms were used to choose the best one to represent the word sense clusters. For evaluating the performances of the built models, the F1 scores are calculated for each model. According to the results, the BERT model achieved the best performance compared to other models, with an 87.4% F1 score.

In the study by Rahman and Borah [33], an unsupervised method is proposed for WSD. The proposed WSD technique is implemented based on an unsupervised methodology. Multiple datasets were used that were created explicitly for the WSD problem. These datasets are, respectively, Senseval-2, Senseval-3, SemEval-07, SemEval-13, and SemEval-15. For the methodology, first, preprocessing steps are applied to all datasets. In the preprocessing steps, stop words are removed from the datasets, and stemming is applied to find the root form of words. Then PoS tagging is applied to find the tag of each word in the datasets, and lastly, named entity tagging is applied to distinguish different names, locations, organizations, etc. The collocation feature is obtained for the purpose of finding the correct meaning of the word. Collocation refers to the words or phrases used with other words or phrases [33]. With the collocation, it is possible to find the information on which words occur near the other words. The collocation score is found for deriving the collocation feature. The formula for finding the collocation score is detailed and explained in the study [33]. For each sense of a word, the collocation score is calculated. And according to this calculation, the correct sense of the word is determined. The proposed model is evaluated for all the datasets. According to the results, the proposed system achieved the highest score, which is 77.8%.

In the study by Kohli [34], the WSD problem is tried to be solved by proposing a method that uses transfer learning and augmentation. For datasets the proposed model uses various datasets. For training, the system mainly uses SemCor 3.0. SemCor 3.0 is a specially annotated WSD corpus of over 226k word sense tags. For training purposes, SemEval-2007, SemEval-2013, SemEval-2015, Senseval-2, and Senseval-3 corpuses are used. Data augmentation is performed on the datasets used for training purposes. For the base of the model, Multi-Task Deep Neural Networks (MT-DNN) architecture is used. The model consists of multiple shared layers and task-specific layers. After building the model, it is trained and tested with the specific datasets. Then the prepared model is evaluated. According to this evaluation, the model achieved a score of 79.0%.

In the study by Duarte et al. [35], deep analysis of semi-supervised learning and neural networks WSD are represented. As datasets, Senseval-2, Senseval-3, Semeval-2007, and SemCor corpora are used. In the experimental setup, in the first step, the datasets are preprocessed by applying, respectively, tokenization, lemmatization, stop word removal, and PoS-Tagging. Then the word embeddings from Word2Vec, FastText, GloVe, Bert, and Electra are extracted to represent the WSD. After the embedding extractions and combined with the PoS tags of words, for semi-supervised learning algorithms, Label Propagation (LP), Local and Global Consistency (LGC), Gaussian Random Fields (GRF),

OMNI-Prop (OMNI) algorithms are trained and used with different distance measures such as cosine, Euclidean, Manhattan, Chebyshev, etc. These models are run to build graphs with different parameters. After all these steps, to evaluate the prepared algorithms, the F1 scores of each algorithm are calculated on different datasets. According to the results, LGC algorithms with Electra achieved the best performance with an 88% F1 score.

In the study by Zhang et al. [14], a WSD model using GCN is proposed to enhance the disambiguation accuracy. In the proposed model, PoS, semantic category, words were extracted from contexts of the ambiguous word and set as discriminative features. Sentence and discriminative features are set as nodes of the proposed WSD graph. For embeddings of nodes and edge weights, Doc2Vec, Word2Vec, PMI, and Term Frequency-Inverse Document Frequency (TF-IDF) were used. For experiments, training, and test corpus of SemEval-2007: Task #5 was used. Experiments were separated into four groups. In two groups, a comparison was made between GCN, CNN, LSTM, CNN+LSTM, and CNN+BiLSTM models on constructed WSD graphs. In other groups, experiments were done with GCN to analyze the effect of the number of layers and window size on average accuracy separately. According to the results, it showed that the performance of GCN is better on WSD than other models, and better results could be obtained with less number of layers.

In a study by Kwon et al. [9], two novel methods were suggested. An encoding method for representing word vectors using Lexical Knowledge Bases (LKBs) is suggested as the first method. Secondly, a method that uses word similarity to specify the context of the ambiguous word was suggested. For realizing the second method, specific thresholds were used to decide whether a word was sufficiently relevant to an ambiguous word. The thinking behind this process was that some words were more important than others for describing context. For experiments, SemEval-02, SemEval-03, SemEval-07, SemEval-13, SemEval-15 datasets were used. Also, WordNet[1] and BabelNet were used as LKBs. According to the results, it showed that their model was more powerful than the compared knowledge-based WSD systems.

In the study by Le et al. [15], two mechanisms were proposed to enhance the performance of deep learning models for metaphor detection. For the first mechanism, a dependency parse tree was used with GCN. Using dependency parse trees relied on the idea that some words in sentences are more important than others for metaphor detection. As the second mechanism, a multi-task learning framework composed of four modules was proposed to take advantage of the similarity between WSD and metaphor detection tasks on modeling and semantic level. Experiments were done with VUA, MOH-X, and TroFi datasets. According to the results, state-of-the-art performance was achieved with their proposed model for metaphor detection.

In the study by Zhao et al. [16], the ES-GCN algorithm was proposed for semi-supervised node classification. In the ES-GCN framework, the entropy aggregation (EA) layer used for improving the reasoning ability of GCN was placed. This step was the novel part compared to previous studies. It is mentioned that multiple EA layers could be stacked after GCN layers in the study. Another proposed method was to expand supervised information. According to the study, supervised information had a significant effect on the performance of GCN models. For this reason, and feature aggregation (FA) checking part that was used for generating pseudo-labels had higher quality was placed in the ES-GCN. For experiments, Cora, CiteSeer, PubMed, Chameleon, Squirrel, and Acotr datasets were used. According to experiment results, ES-GCN gave better results than baseline algorithms generally (especially experiments on the small ratios of labeled nodes).

In the study by Calvo et al. [36], a method was presented to disambiguate the words that were unseen during training. For experiments with four proposed models, Senseval 2 and Senseval 3 English Lexical Sample were used. Proposed models had the advantage of disambiguation from the point of not being mandatory to a fixed set of words. Senseval 3 ELS was used for comparison with the state-of-the-art mentioned in the study. According to experiment results, the performance of models was close to the state-of-the-art that is mentioned in the study but could not outperform.

In the study Butnaru by et al. [37], a bio-inspired unsupervised, knowledge-based three-phased global WSD algorithm (named ShotgunWSD version 2.0) was proposed. ShotgunWSD had a different computational manner for describing the relatedness of two senses of words. For experiments, SemEval 2007, SemEval 2013, SemEval 2015, Senseval-2, Senseval-3, and unified datasets were used. In experiments, ShotgunWSD 2.0 was compared to previous versions of ShotgunWSD and other knowledge-based or unsupervised approaches. According to experiment results, ShotgunWSD outperformed many unsupervised, knowledge-based methods and the previous version of ShotgunWSD. Also, the baseline method mentioned in that paper was outperformed on two datasets.

In the study by Corrêa Jr. et al. [38], it was pointed out that improvements in the literature on WSD play a significant role in systems such as search engines and machine translators. For the study, context words and words wanted to disambiguate were used as elements of a bipartite network. The inductive Model Based on Bipartite Heterogeneous Network (IMBHN) algorithm was used with bipartite networks for WSD. In the IMBHN algorithm, relevances of topical/local features (topical and local features mentioned in the study) for related ambiguous words were used for inferring senses. For the experiments, Senseval-3 English Lexical Sample Task, SemEval-2007 Task 17 English Lexical Sample, and sub-datasets (i.e., Interest, Line, Hard, and Serve) of SensEval [30] were used. According to experiment results, IMBHN outperformed the other models on sub-datasets Interest, Line, and Serve. Also, IMBHN outperformed the baseline model with significant differences in F-scores for Senseval-3 and

SemEval-2007. In general, it was observed that their method also had good performance with using smaller amounts of data on training.

In the study by Silva et al. [39], it was pointed out that WSD had a crucial role in machine translation, and high-level learning was important for WSD besides low-level order learning. Conventional approaches for supervised classification methods to disambiguate the word sense considers only physical or topological features. Conversely, the hybrid approach composed of low-level and high-level order learning was employed. According to experiment results, it was showed that high-level terms play a significant role in enhancing the performance of classification in real-world and artificial networks.

III. APPROACH

This section will describe the methodology and present the necessary notations used throughout the article.

A. SEMANTIC DIFFUSION

We assume that the corpus in use has a size of m documents and t terms. In our setting, we denote an undirected graph with $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. We let m be the number of nodes and $A \in \text{Mat}_m(\mathbb{R})$ to be the adjacency matrix of size m. In this study, we use Bag of Words (BoW) to compute the representation matrix $X \in \text{Mat}_{m,t}(\mathbb{R})$ which is called the document-by-term matrix representation of textual data to be classified. Our straightforward approach to creating the adjacency matrix for the graph $\mathcal{G}$ for textual data is as in (1). We call this matrix a BoW-based adjacency matrix.

$$A = XX^T \in \text{Mat}_m(\mathbb{R}) \tag{1}$$

In the next section, we use the semantic diffusion matrix to create an adjacency matrix. We compare the effects of the straightforward approach, and the diffusion approach gives the added value of the latter.

The diffusion process is a method that has roots from the heat equation, and it can be described as the discretization of the Gaussian kernel [25], [40]. The semantic diffusion matrix is denoted as S and calculated as in (2).

$$S = \sum \frac{\lambda^i}{i!} G^i \tag{2}$$

In (2), G is calculated as in (3).

$$G = X^T X \in \text{Mat}_t(\mathbb{R}) \tag{3}$$

In the eyes of the semantic studies on graphs, the diffusion process models the semantic similarities between terms in the corpus as explained in [26] and [27].

Algorithm 1 Semantic Diffusion Matrix

Require: Matrix X, Parameter λ, Taylor Step T
Ensure: Semantic Diffusion Kernel
Initialization:
$G = X^T X$
$S : 2I_t \in \text{Mat}_t(\mathbb{R})$
1: **for** $i = 1$ to $T - 1$ **do**
2: $\quad S := S + \frac{\lambda^i}{i!} G^i$
3: **end for**
4: **return** $0.5\ S$

B. TEXT GCN WITH DIFFUSION

In the Text GCN algorithm, in [24] Yao et al. use a large and heterogeneous text graph. The adjacency matrix of this graph contains term (word) nodes and document nodes to model term co-occurrence within the whole corpus. In this algorithm, two convolution layers are used. One layer captures the term level convolution, and the other layer document level convolution [24].

Let us describe the construction of the adjacency matrix in Text GCN in detail and how we tweak it. In Text GCN, PMI is computed between words to grasp inter-word relations. Hence in the graph, an edge exists between each term with a weight value coming from the PMI computation defined in (4).

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

Between a document and all terms used in this document, connections are built by using TF-IDF values. A critical remark is that separate documents have no connection, as seen in (5).

$$A_{ij} = \begin{cases} \text{PMI}_{ij}, & i,j \text{ are terms, PMI}_{ij} > 0 \\ \text{TF-IDF}_{ij}, & i \text{ is document, } j \text{ is term} \\ 1, & i = j \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

The adjacency matrix is a matrix $A \in \text{Mat}_{m+t,m+t}(\mathbb{R})$, which can be visualized as in (6).

$$A = \begin{pmatrix} I_m & \text{TF-IDF} \\ \text{TF-IDF}^T & \text{PMI} \end{pmatrix} \tag{6}$$

In the matrix (6), $I_m \in \text{Mat}_m(\mathbb{R})$ denotes the identity matrix of size m, TF-IDF $\in \text{Mat}_{m,t}(\mathbb{R})$ denotes the term frequency-inverse document frequency matrix that gives the relations between terms (words) and documents of size $m \times t$ and PMI $\in \text{Mat}_t(\mathbb{R})$ denotes the matrix that gives the relations between terms that is of size t. The Text GCN algorithm relies on inter-term and term-document relations to find the document connections. There are some drawbacks of this algorithm as the corpus grows. One is that the PMI matrix in the adjacency matrix given in (6) results in a larger term-by-term matrix. This process results in hardware limitations during computations, but more importantly, it adds noise to the system. Some attempts are made to avoid this issue, for example, limiting the vocabulary used in TF-IDF computations. This strategy resembles the thresholding technique Section III-D that we apply for GCN and GAT. In Text GCN computations, we did not apply such methods.

In this WSD study, we replace the word-by-word PMI-based matrix with a semantic diffusion matrix S. The S matrix already carries semantic similarities within the terms. Our experiments (see section IV) show that it describes inter-word semantic relation better than PMI, specifically for WSD tasks. In essence, diffusion is a learning process, namely a model. We transfer what the semantic diffusion process learned to the GCN model. To use the semantic diffusion process, we replace the PMI matrix with the S matrix. Hence, we rewrite the adjacency matrix in (5) as in (7).

$$A = \begin{pmatrix} I_m & \text{TF-IDF} \\ \text{TF-IDF}^T & S \end{pmatrix} \tag{7}$$

Semantic Diffusion matrix $S \in \text{Mat}_t(\mathbb{R})$ will give a dense sub-graph structure. This structure contains several semantic links between the words. There are two normalization steps that we interchangeably apply to the semantic diffusion matrix. The first normalization is called semantic normalization. This process aims to keep the semantic similarity of different words [0, 1) where the score of the same word is 1. We compute the semantic normalization by (8).

$$\frac{S_{ij}}{\sqrt{|\ S_{ii}\ ||\ S_{jj}\ |}} \tag{8}$$

Rapid changes between close points cause the model to fail to fulfill the cluster assumption. A method called graph normalization is a way to solve this issue. In the following sections, we will call this process regularization to avoid confusion with semantic normalization. We note that for the Text GCN algorithm, we do not regularize the whole adjacency graph but only the semantic diffusion part of it. We will denote the regularized matrix as in (9).

$$D^{-1/2} S D^{-1/2} \tag{9}$$

D is the degree matrix of S can be computed as in (10).

$$D_{ii} = \sum_i S_{ij} \tag{10}$$

After the graph is built, we feed this structure to two layered GCN as described in [41] and [24].

C. GCN WITH DIFFUSION

Text GCN algorithm utilizes a large matrix that we replace with a more compact version that encapsulates the semantic relations. We aim to eliminate the convolution process on words of the Text GCN algorithm and mimic the process by the diffusion process and carry this information to a more compact adjacency matrix. The semantic diffusion process propagates weight on the graph of the words and creates a similarity score between each word. One of the convolution layers of Text GCN precisely focuses on doing this process. Since diffusion processes are similar to convolution on words, we can reduce this adjacency matrix in (5) and (6) to the matrix of size m. To empirically show that this idea has its merits, we introduce a method to utilize the semantic diffusion matrix in the GAT and GCN settings.

Let us first review the Graph convolution setting described in [41]. Consider a multi-layer GCN with the following layer-wise propagation rule as in (11).

$$H^{(l+1)} = \sigma(D^{-\frac{1}{2}} A D^{-\frac{1}{2}} H^{(l)} W^{(l)}) \tag{11}$$

Here, A is the adjacency matrix of the undirected graph with self-loops. The matrix D is the degree matrix of A. The weight matrix W is the matrix to be learned, and the sigma function σ denotes an activation function.

In [20], they also create a diffusion matrix based on the heat kernel. They create a transition matrix, mainly a graph-regularized diffusion matrix as defined in (12).

$$T_{sym} = D^{-1/2} S D^{-1/2} \tag{12}$$

The symmetric transition matrix T_{sym} will not serve our needs to get inter-word and document similarities simultaneously. There are other ways to create this matrix in different circumstances [42]. Our approach is more to describe the semantic interrelations between documents through the diffusion process. Namely, we want to use a matrix that gives the inter-document similarity from the diffusion process. To accomplish this, we need to define the node features of our graph as Feature Matrix by (13). For GCN and GAT models, we normalize the S matrix given in (13) as in (8).

$$\text{Feature matrix} = F = XS \tag{13}$$

We construct the adjacency matrix in the semi-supervised setting as in (14).

$$A = XS(XS)^T = XSS^T X^T \tag{14}$$

We experimented with all possible regularization scenarios. The semantic matrix S is semantically normalized as in (8). The adjacency matrix A can be regularized as in (15).

$$D^{-1/2} A D^{-1/2} \tag{15}$$

D is the degree matrix of A can be computed as in (16).

$$D_{ii} = \sum_i A_{ij} \tag{16}$$

D. THRESHOLDING

The diffusion process creates similarity links between each term in the corpus. This results in a dense matrix S. To get a document-by-document relationship we apply (14) to S, which is eventually converted to a complete graph structure. During our experiments, we observed that the learning process is hindered drastically with a complete graph containing such weak links. Removing some links from the graph restored the learning process. We conclude that such links create noise in the system. This observation is based on carefully tracking the loss functions after each epoch. We cut some edges with weights smaller than the given threshold value, which solves this issue. We took this threshold value as another hyperparameter and did the threshold search on training and validation data. Test data is never introduced in the threshold search to avoid any leakage.

In literature, we also come across such techniques to clean noise [20], [43] on the diffusion matrices. Our observation implies that learning on graphs happens on sparse matrices better than dense ones. A more theoretical approach to noise-cleaning on graphs in the case of diffusion application can be found in [43].

In the study [14], the effect of each GCN layer in the Text GCN setting is explained very clearly. The first GCN layer deals with the local word (two right and two left words) connection, and the second layer of GCN deals with global word connections. The study clearly stated that the global connection between words has a better effect on the performance of the Text GCN. Similarly, in our study, we observed that more than one GCN layer would give the best performance. Moreover, our diffusion GCN framework becomes more robust with global word-by-word relations obtained from the diffusion process.

On the other hand, the thresholding on similarity scores from the diffusion process heuristically defines sense-based locality on the words. A study that emphasizes the power of local word connections in a graph setting is [38]. In our opinion, the locality definition in that study can be reformulated by thresholding on similarity measures. This can extend the graph structure in our study to bipartite graphs settings. Adapting the GCN structure to a bipartite graph is also possible via Graph Convolutional Collaborative Filtering (GCCF) structure [44], which can be a good future work direction.

E. TRAINING

We apply semantic normalization to S by (8) and, depending on the experiment graph regularization to adjacency matrix A in (15) by using (16). Then, we get the best threshold value and eliminate noise from the adjacency matrix by applying the thresholding process.

For Text GCN model training, the adjacency matrix is as in (6) or (7). We train Text GCN and GCN as explained in [41]. As pointed out in [41], the repeated application of the convolution creates exploding or vanishing gradients. A small trick of adding self-loops solves this problem. The following operations can define forward propagation in GCN. During convolution, graph regularization is applied to the adjacency matrix with the self-loops.

$$f(X, A) = \text{softmax}(A\text{ReLU}(AXW^{(0)})W^{(1)}) \tag{17}$$

Note that in (17), for Text GCN, X is set to be the identity matrix of size A [24]. For GCN/GAT, it is the feature matrix. The softmax function is defined as in (18).

$$\text{softmax}(\boldsymbol{x_i}) = \frac{\exp(\boldsymbol{x_i})}{\Sigma_i \exp(\boldsymbol{x_i})} \tag{18}$$

But in our GCN/GAT experiments, we observed better results without the softmax layer. Hence we skipped this step. The loss function in (19) is the same as defined in [41].

$$\mathcal{L} = -\sum_{d}^{} \sum_{f}^{F} Y_{df} \ln Z_{df} \tag{19}$$

In (19), the index d denotes the indices of labeled documents and F is the dimension of output features, Z is the output of (17), and Y is the labeled indicator matrix as described in [24].

We followed the same structure described in the GCN part for the GAT layer but added an attention layer on top of our GNN architecture. The GAT structure is used as described in [46].

IV. EXPERIMENTS AND RESULTS

A. DATASETS

Experiments were done on four different sub-datasets (i.e., Interest, Line, Hard, and Serve) of the SensEval [30] dataset widely used for WSD. Information about datasets is shown in Table 1. For all experiments, we split datasets into training, validation, and test sets as stratified. Label ratios mentioned in all result tables (Table 2, Table 3, Table 7, Table 6, Table 8) were ratios of labeled nodes to all nodes in the dataset. The validation set ratio was set as %10 of unlabeled nodes. In addition, because of imbalanced datasets, some labels are too few for splitting stratified. When a kind of label was not found in the training set, we randomly selected one sample of this kind of label from the test set and transferred it to the training set. In final, every kind of label was found in the training set. Distributions of senses in each dataset can be found in Fig. 1.

TABLE 1. Graph structure for each dataset.

Dataset	Number of Documents	Number of Words	Number of Classes
Interest	2,367	6,371	6
Serve	4,378	16,667	4
Line	4,146	16,073	6
Hard	4,333	11,589	3

B. EXPERIMENTS ON TEXT GCN WITH DIFFUSION

In experiments on Text GCN with diffusion, we set the window size as 10 for method Text GCN, and we set the $\lambda = 0.02$, Taylor Step as 4 for experiments of all diffusion methods. Using the Adam optimizer, we trained models with two convolutional layers (330 neurons). The Single random seed was used. The random seed was 42.

The abbreviations that we use in Table 2 and Table 3 are as follows. T-DIFF stands for the model that we get by replacing PMI with Semantic Diffusion Matrix, as in Algorithm 1. The model T-DIFF REG is created by replacing PMI with Semantic Diffusion Matrix that is regularized by (9). T-DIFF NORM is created by replacing PMI with Semantic Diffusion Matrix that is normalized by (8). T-DIFF NORM REG is obtained by replacing PMI with Semantic Diffusion Matrix that is normalized and regularized by (8) and (9), respectively.

C. RESULT ANALYSIS FOR EXPERIMENTS ON TEXT GCN WITH DIFFUSION

In this sub-section, we analyze the experiment results in Table 2 and Table 3.

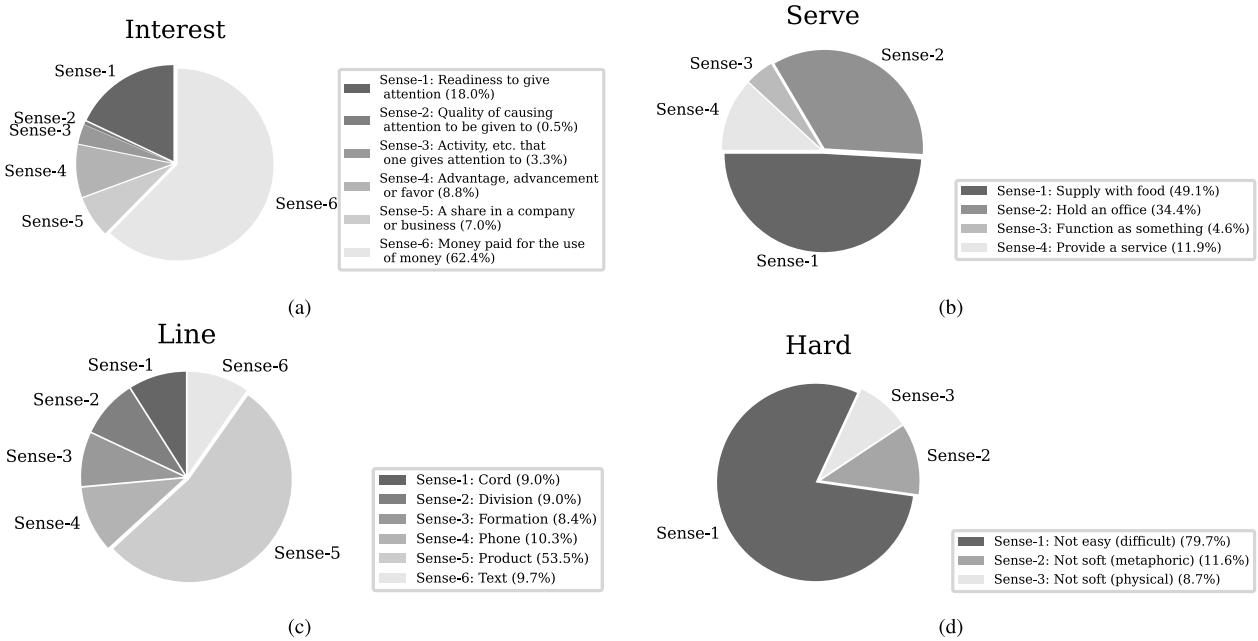

FIGURE 1. Sense distribution and explanation of each sense for each dataset.

For the Interest dataset, in general, T-DIFF REG and/or T-DIFF NORM REG perform better between roughly 3% - 15% than traditional Text GCN in accuracy and/or F1 scores.

TABLE 2. Accuracy scores of exchanging PMI versus versions of diffusion algorithm on Text GCN (%).

Dataset	Label Ratio	Text GCN	T-DIFF	T-DIFF NORM	T-DIFF REG	T-DIFF NORM REG
Interest	80	77.46	70.66	74.88	**81.92**	80.75
	70	77.00	70.58	74.18	79.49	**80.59**
	60	75.12	69.84	74.65	75.94	**76.88**
	50	75.77	69.67	73.33	77.65	**79.34**
	5	66.75	61.22	**68.97**	68.13	68.53
	3	64.60	60.24	**67.07**	66.20	66.54
	2	63.73	59.08	64.54	65.12	**65.45**
Serve	80	79.06	71.70	74.75	**82.23**	81.60
	70	81.56	71.07	74.20	82.32	**82.83**
	60	79.76	70.18	72.91	80.39	**80.84**
	50	79.19	70.51	73.40	81.98	**82.59**
	5	70.70	62.21	66.21	72.12	**72.17**
	3	**69.96**	61.36	66.64	69.05	69.57
	2	67.08	60.14	65.58	67.26	**67.83**
Line	80	78.98	53.55	53.82	**86.08**	84.47
	70	80.61	53.44	53.80	84.09	**84.54**
	60	79.04	53.45	53.85	82.92	**84.59**
	50	78.07	53.46	53.99	**82.90**	81.88
	5	63.47	53.46	53.96	65.70	**66.91**
	3	61.59	53.50	53.97	63.61	**64.69**
	2	59.75	53.57	53.95	61.61	**62.07**
Hard	80	82.30	80.13	81.54	81.79	**82.44**
	70	**81.79**	80.17	81.20	80.85	81.71
	60	**82.76**	79.68	81.79	82.31	82.37
	50	**81.08**	80.00	80.67	78.67	79.13
	5	79.95	79.56	79.76	80.00	**80.11**
	3	**79.78**	79.73	79.25	79.65	**79.78**
	2	**80.09**	79.36	79.72	79.98	79.88

For the Serve dataset, in F1 scores, T-DIFF-REG and/or T-DIFF NORM REG perform better between roughly 3% - 7% than traditional Text GCN. F1-score results are more significant than the accuracy scores to compare methods because of the unbalanced distribution of senses. Nevertheless, T-DIFF REG and/or T-DIFF NORM REG perform better than traditional Text GCN in accuracy.

For the Line dataset, in accuracy and/or F1 scores, T-DIFF REG and/or T-DIFF NORM REG perform better between roughly 3% - 10% than traditional Text GCN, in general.

For the Hard dataset, in general, differences in results of accuracy scores between the group of diffusion methods and traditional Text GCN are roughly under 1%, which is insignificant. Conversely, in F1 scores, T-DIFF NORM or T-DIFF-NORM REG performs better between roughly 3% - 7% on some labeling ratios. According to the results, generally, there is no dominance in favor of the group of diffusion methods or traditional Text GCN for the Hard dataset.

According to the results, the best method can be specified as T-DIFF NORM REG among the methods of this group of experiments.

D. EXPERIMENTAL SETUP FOR THRESHOLDING

In the threshold search, for each dataset separately, we created cleaned adjacency matrices by applying Algorithm 2 to non-cleaned adjacency matrices. We used Algorithm 3 and Algorithm 4 to create our threshold search space that includes candidate threshold values for being the best threshold value. Adjacency matrices and node feature matrices were created (see section III-C.), and models were trained according to the method DIFF NORM GCN (see section IV-F.). For each dataset separately, the best threshold values were specified by sorting threshold search results concerning the choice of metrics on the validation set. Then the best threshold values were used for experiments of method DIFF NORM GCN.

Every unique element in the set of edge weights of the adjacency matrix could be a threshold value. In other words, These are possible candidates. But, trying every unique element as a threshold value to find the best threshold value is

TABLE 3. F1 scores of exchanging PMI versus versions of diffusion algorithm on text GCN (%).

Dataset	Label Ratio	Text GCN	T-DIFF	T-DIFF NORM	T-DIFF REG	T-DIFF NORM REG
Interest	80	43.48	30.25	34.24	**59.54**	47.63
	70	42.93	29.68	32.84	56.37	**57.96**
	60	36.48	29.53	33.41	37.12	**38.09**
	50	44.40	28.49	32.05	53.25	**55.34**
	5	32.47	24.59	29.48	**35.38**	35.18
	3	30.74	24.24	27.54	**33.65**	28.99
	2	28.45	24.00	26.78	**30.39**	29.87
Serve	80	70.22	51.16	54.65	**77.43**	76.67
	70	75.24	49.79	54.09	76.97	**77.45**
	60	73.14	49.45	52.93	75.31	**75.85**
	50	70.59	49.91	53.51	76.45	**77.74**
	5	61.58	43.45	44.95	**64.76**	62.53
	3	60.81	43.53	52.92	60.87	**62.82**
	2	53.18	38.86	51.35	57.03	**59.01**
Line	80	68.63	11.62	12.49	**78.37**	76.09
	70	71.31	11.61	12.80	75.59	**76.04**
	60	68.14	11.61	12.97	73.32	**76.53**
	50	66.81	11.61	13.43	**73.65**	72.14
	5	38.61	11.61	13.40	43.50	**45.84**
	3	34.96	11.72	13.31	39.60	**41.72**
	2	31.01	11.89	13.17	35.26	**36.19**
Hard	80	50.73	33.86	40.54	51.33	**53.55**
	70	46.86	34.17	39.10	39.65	**50.28**
	60	**54.51**	30.42	45.94	50.94	50.32
	50	54.39	35.34	36.26	**55.02**	54.87
	5	**40.37**	29.84	40.31	34.62	37.12
	3	30.68	29.57	**37.75**	30.64	30.67
	2	**41.41**	29.51	35.98	33.59	32.73

a computationally expensive way. For this reason, Threshold values corresponding to our desired percentage of the number of edges were selected as candidates.

To apply our approach, we used Algorithm 3 with parameter Step as 0.0001 to create a threshold search pool that includes possible candidates of the best threshold value and the number of edges corresponding to those. To specify our candidates from the threshold search pool, we created our threshold search space by Algorithm 4 with parameter R as [0.01, 0.015, 0.02, 0.025, 0.03, 0.035, 0.04, 0.045, 0.05, 0.055, 0.06, 0.065, 0.07, 0.075, 0.08, 0.085, 0.09, 0.095 0.1, 0.12, 0.14, 0.16, 0.18, 0.2, 0.3, 0.4, 0.5, 0.6, 0.8, 1.0]. Note that the R represents a list that includes the desired percentage of the number of edges of the adjacency matrix. The parameter Step in Algorithm 3 and Algorithm 4 is the same, and it can be explained as a sensitivity value to search our candidates among possible candidates.

We set the number of the convolutional layers as 2, the number of neurons as 128, the dropout rate as 0.6, the learning rate as 0.01, and weight decay as 0.01 for all datasets. For threshold search, models were trained with a single random seed (random seed was 42.) and cleaned adjacency matrices on the 5% label ratio. Parameters for diffusion algorithms, optimizer, scheduler, and their parameters were the same as in section IV-F. Thresholding results are visualized in Fig. 2.

For Algorithm 2, let N denote the number of nodes of adjacency matrix A (A has non-negative elements.), and V denotes the threshold value.

For Algorithm 3, L denotes the threshold search pool, U denotes the sorted (ascending order) array of unique elements in the set of edge weights of the adjacency matrix A (A has non-negative elements.), M denotes the length of U, and V denotes the threshold value.

For Algorithm 4, let R denote a list that includes percentages of a desired number of edges, J denotes the length of R, K denotes the number of edges of non-cleaned adjacency matrix A (A has non-negative elements.), B denotes the desired number of edges, and O denotes the threshold search space.

Algorithm 2 Thresholding

Require: Adjacency matrix A, Parameter V.
Ensure: Cleaned adjacency matrix.
for $i = 0$ to $N - 1$ **do**
  for $j = 0$ to $N - 1$ **do**
    if $(A[i][j] \geq V$ and $A[i][j] > 0)$ **then**
      continue
    else if $(A[i][j] < V)$ **then**
      $A[i][j] := 0$
    end if
  end for
end for
return A

Algorithm 3 Threshold Search Pool

Require: Adjacency matrix A, Parameter Step.
Conditions:
$0 <$ Step ≤ 1
Ensure: Threshold Search Pool
Initialization:
$Q := 0$, $L :=$ Empty List
while $Q < M$ **do**
  $V := U[Q]$
  $E :=$ Number of edges of Thresholding(A, V)
  L.append($[Q, V, E]$)
  $Q := Q + \lceil M \times \text{Step} \rceil$
end while
return L

Algorithm 4 Threshold Search Space

Require: Adjacency matrix A, Parameter Step, R.
Conditions:
$0 <$ Step ≤ 1
$0 <$ Elements of $R \leq 1$
Ensure: Threshold Search Space
Initialization:
P: Reversed array of Threshold Search Pool(A, *Step*)
C: Length of P, $O :=$ Empty List.
for $i = 0$ to $J - 1$ **do**
  $B := \lceil R[i] \times K \rceil$
  for $j = 0$ to $C - 1$ **do**
    if $(P[j][2] \geq B)$ **then**
      O.append($P[j]$)
      break
    end if
  end for
end for
return O

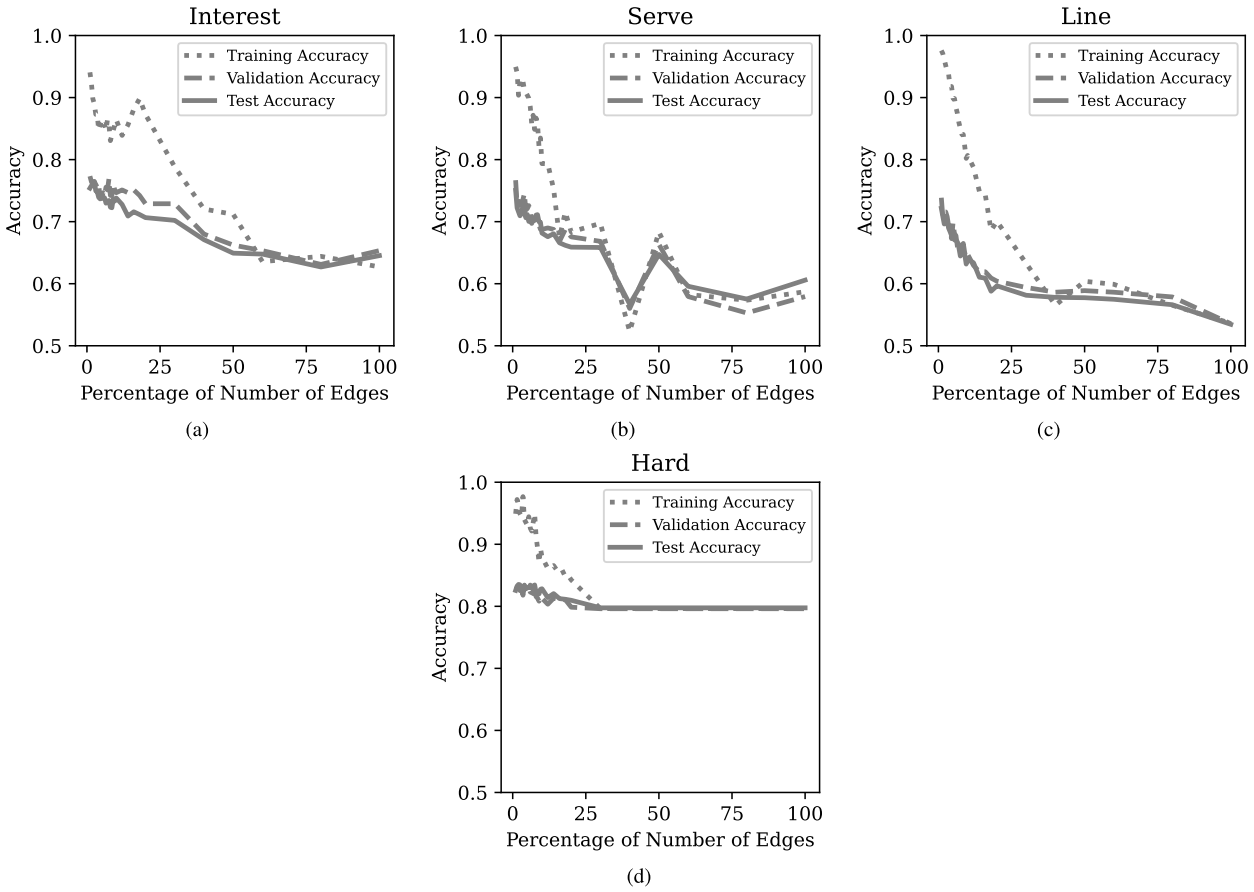

FIGURE 2. Accuracy scores of each dataset respect to percentage of number of edges.

E. HYPERPARAMETER TUNING

After the threshold search, hyperparameter tuning was done with cleaned adjacency matrices of method DIFF NORM GCN for each dataset separately. For hyperparameter tuning, we used a single random seed as 42 and the following parameter spaces. Space of the number of convolutional layers as [2] and [3], space of the number of neurons as [64,128,256], space of weight decay values as [0.0001, 0.001, 0.01], space of learning rate as [0.001, 0.005, 0.05, 0.01], space of dropout rate as [0.3, 0.6]. Hyperparameter tuning was done on the 5% label ratio. Parameters for diffusion algorithms, optimizer, scheduler, and their parameters were the same as in section IV-F. The best hyperparameters were specified with evaluations on the validation set. The best hyperparameter values are shown in Table 4. The same hyperparameters were used for all methods with the related dataset.

F. EXPERIMENTS ON GCN/GAT WITH DIFFUSION

Experiments of diffusion with GCN denoted as DIFF NORM GCN. Experiments of diffusion with GAT denoted as DIFF NORM GAT. For experiments of methods BOW GCN, and BOW GAT, we used the graph-based adjacency matrix defined in (1), and we set the node feature matrix as the BoW matrix. Experiments of BOW GCN and BOW GAT were done with GCN and GAT, respectively.

TABLE 4. Hyperparameters for each dataset.

Dataset	# Layers	# Neurons	Weight Decay	Dropout Rate	Learning Rate
Interest	2	256	0.01	0.3	0.01
Serve	2	128	0.0001	0.6	0.01
Line	3	128	0.001	0.3	0.01
Hard	2	128	0.001	0.6	0.01

For each dataset, we used the Taylor Step as 2 and $\lambda = 0.02$ to create semantic matrices. In all experiments, models were trained with an AdamW optimizer. We used Scheduler[2] with the parameter "patience" as 100 and the parameter "factor" as 0.9 by mode of minimizing validation loss. Learning rate, weight decay, the number of layers, the number of neurons, and dropout rate values were specified by hyperparameter searching. Dropout layers were located after GCN/GAT layers, except for the last layer. The dropout rate value was mutual for every dropout layer. Ten different random seeds were used for experiments. Random seeds were [42,78,153,12,109,553,201,67,32,138]. Then, scores are averaged.

[2]Scheduler named "ReduceLROnPlateau" that belongs to Pytorch.

The number of edges of graphs that are created from some datasets is very large. This causes an out-of-memory problem in GPU. It forced us to apply Algorithm 2 to reduce memory consumption instead of noise elimination in some experiments. Percentages of the number of edges corresponding to a used threshold value that we used for every experiment are shown in Table 5.

TABLE 5. Information about graphs used for diffusion GCN/GAT experiments.

Dataset	Method	# Edges in Non-Cleaned Graph	# Edges in Cleaned Graph	Percentage
Interest	DIFF NORM GCN	2,802,528	28,277	1.01%
	DIFF NORM GAT	2,802,528	2,802,528	100.00%
	BOW GCN	2,218,556	2,218,556	100.00%
	BOW GAT	2,218,556	2,218,556	100.00%
Serve	DIFF NORM GCN	9,585,631	96,471	1.01%
	DIFF NORM GAT	9,585,631	3,834,795	40.01%
	BOW GCN	4,171,210	4,171,210	100.00%
	BOW GAT	4,171,210	4,171,210	100.00%
Line	DIFF NORM GCN	8,596,731	86,171	1.00%
	DIFF NORM GAT	8,596,731	8,596,731	100.00%
	BOW GCN	6,163,984	6,163,984	100.00%
	BOW GAT	6,163,984	6,163,984	100.00%
Hard	DIFF NORM GCN	9,389,611	470,399	5.01%
	DIFF NORM GAT	9,389,611	3,755,909	40.00%
	BOW GCN	7,188,614	7,188,614	100.00%
	BOW GAT	7,188,614	7,188,614	100.00%

During the calculations, we observed that some decimal errors occurred. Although these errors seem insignificant, Algorithm 2 is very sensitive to them. This sensitivity results in a non-symmetric adjacency matrix when comparing numbers. To avoid such a problem, we forced symmetry by assigning the upper triangular of the adjacency matrix to the lower triangular.

Accuracy scores for Diffusion GCN/GAT experiments are shown in Table 6. F1 scores for Diffusion GCN/GAT experiments are shown in Table 8. Comparative results for Text GCN and Diffusion GCN/GAT experiments are shown in Table 7.

G. RESULT ANALYSIS FOR EXPERIMENTS ON GCN/GAT WITH DIFFUSION

In this sub-section, we analyze the experiment results in Table 6 and Table 8.

For the Interest dataset, in general, DIFF NORM GCN performs better by at least 3% - 12% than other methods in accuracy and/or F1 scores, which is significant. Results show a significant difference between DIFF NORM GCN and other methods.

For the Serve dataset, on the labeling ratios of 80% and 50%, DIFF NORM GCN performs better by at least roughly 4.5% - 7% than other methods in accuracy and/or F1 scores. For other labeling ratios, experiment results show that the performances of BOW GCN and DIFF NORM GCN are very close to each other, and they outperform the other methods.

For the Line dataset, DIFF NORM GCN outperforms the other methods by at least roughly 8.5% - 46% differences in accuracy and/or F1 scores. Results show a significant difference between methods in favor of DIFF NORM GCN.

For the Hard dataset, DIFF NORM GCN generally has better accuracy scores by at least 2% - 3% differences than the other methods. F1 scores show that DIFF NORM GCN outperforms other methods by at least roughly 4.5% - 18%.

According to the results, the best method can be specified as DIFF NORM GCN among the methods of this group of experiments.

TABLE 6. Accuracy scores for diffusion GCN/GAT experiments (%).

Dataset	Label Ratio	BOW GCN	BOW GAT	DIFF NORM GCN	DIFF NORM GAT
Interest	80	73.22	64.23	**83.17**	52.82
	50	73.53	63.04	**81.54**	52.86
	5	70.17	61.36	**74.59**	52.92
	3	70.09	61.38	**72.92**	52.88
	2	67.93	61.74	**70.05**	52.90
Serve	80	75.61	68.21	**81.84**	67.70
	50	75.91	66.72	**80.40**	68.25
	5	72.56	58.98	**73.90**	62.21
	3	70.60	58.26	**72.39**	60.73
	2	69.10	54.95	**69.97**	60.08
Line	80	59.41	54.35	**81.57**	53.55
	50	58.57	55.09	**80.99**	53.48
	5	57.70	55.60	**70.36**	53.46
	3	57.13	55.02	**67.72**	53.47
	2	56.78	54.70	**65.49**	53.49
Hard	80	79.74	79.36	**82.71**	80.13
	50	79.74	79.33	**82.70**	79.27
	5	79.81	79.76	**81.95**	79.70
	3	79.87	79.54	**81.02**	79.67
	2	79.71	79.43	**80.78**	79.76

H. COMPARISON BETWEEN T-DIFF NORM REG, DIFF NORM GCN, AND TEXT GCN

In this sub-section, we analyze the experiment results in Table 7.

For Serve and Hard datasets on the labeling ratio of 80%, DIFF NORM GCN and T-DIFF NORM REG performances are very close.

DIFF NORM GCN outperforms T-DIFF NORM REG on the labeling ratio of 80% for the Interest dataset. Conversely, for the Line dataset, T-DIFF NORM REG outperforms DIFF NORM GCN on the labeling ratio of 80%.

Conversely to the Serve dataset, DIFF NORM GCN outperforms T-DIFF NORM REG for the Interest dataset on the labeling ratio of 50%. For the Line dataset on the labeling ratio of 50%, DIFF NORM GCN and T-DIFF NORM REG performances are very close. For the Hard dataset on the labeling ratio of 50%, DIFF NORM GCN performs better than T-DIFF NORM REG in accuracy and precision scores.

In general, DIFF NORM GCN performs significantly better than T-DIFF NORM REG and traditional Text GCN on the other labeling ratios (5%, 3%, 2%) for all datasets. This result shows that if models are trained using a relatively small amount of data, DIFF NORM GCN performs better than the other methods.

According to the results, DIFF NORM GCN and T-DIFF NORM REG outperform the traditional Text GCN (baseline) in general.

I. FURTHER DISCUSSIONS

The most significant aspect of this study is that our models perform significantly well with tiny label ratios, such as 2%

TABLE 7. Comparative table (%).

Dataset	Label Ratio	DIFF NORM GCN				T-DIFF NORM REG				Text GCN			
		Accuracy	F1 Score	Precision	Recall	Accuracy	F1 Score	Precision	Recall	Accuracy	F1 Score	Precision	Recall
Interest	80	**83.17**	**61.79**	**65.45**	**60.00**	80.75	47.63	48.86	47.24	77.46	43.48	48.03	42.08
	50	**81.54**	**58.47**	**62.68**	**56.31**	79.34	55.34	61.44	52.21	75.77	44.40	61.72	41.66
	5	**74.59**	**45.24**	**55.00**	**43.77**	68.53	35.18	37.94	34.73	66.75	32.47	53.73	31.97
	3	**72.92**	**41.30**	52.48	**40.81**	66.54	28.99	47.38	29.58	64.60	30.74	**54.37**	30.17
	2	**70.05**	**39.53**	**50.64**	**38.19**	65.45	29.87	38.19	29.35	63.73	28.45	37.17	28.15
Serve	80	**81.84**	76.07	**77.89**	75.49	81.60	**76.67**	76.63	**76.89**	79.06	70.22	76.83	68.32
	50	80.40	73.92	76.66	72.80	**82.59**	**77.74**	78.20	**77.38**	79.19	70.59	**78.41**	68.35
	5	**73.90**	**65.43**	**69.55**	**65.05**	72.17	62.53	66.52	61.62	70.70	61.58	64.92	60.78
	3	**72.39**	**64.46**	**68.92**	**63.85**	69.57	62.82	63.64	63.12	69.96	60.81	64.04	60.32
	2	**69.97**	**59.98**	**65.69**	**60.31**	67.83	59.01	61.59	58.80	67.08	53.18	62.64	53.26
Line	80	81.57	72.85	76.47	70.90	**84.47**	**76.09**	79.34	**74.36**	78.98	68.63	**83.41**	62.22
	50	80.99	71.83	**75.11**	**70.01**	**81.88**	**72.14**	76.99	68.80	78.07	66.81	76.99	61.20
	5	**70.36**	**52.85**	**58.38**	**51.70**	66.91	45.84	54.85	42.06	63.47	38.61	53.24	35.05
	3	**67.72**	**47.46**	**55.08**	**47.26**	64.69	41.72	52.52	37.96	61.59	34.96	50.65	31.86
	2	**65.49**	**43.55**	51.53	**42.29**	62.07	36.19	**52.26**	32.97	59.75	31.01	50.27	28.65
Hard	80	**82.71**	52.15	**76.22**	47.69	82.44	**53.55**	72.75	**48.52**	82.30	50.73	74.89	46.20
	50	**82.70**	52.87	**74.22**	48.24	79.13	**54.87**	58.81	**52.38**	81.08	54.39	63.64	50.39
	5	**81.95**	**51.04**	**68.87**	**47.43**	80.11	37.12	61.34	37.14	79.95	40.37	58.76	39.08
	3	**81.02**	**44.30**	**68.62**	**42.32**	79.78	30.67	59.95	33.84	79.78	30.68	65.50	33.85
	2	**80.78**	41.36	**63.85**	**40.64**	79.88	32.73	57.24	34.82	80.09	**41.41**	60.79	39.75

TABLE 8. F1 scores for diffusion GCN/GAT experiments (%).

Dataset	Label Ratio	BOW GCN	BOW GAT	DIFF NORM GCN	DIFF NORM GAT
Interest	80	49.82	27.80	**61.79**	11.52
	50	48.54	27.59	**58.47**	11.53
	5	40.74	22.33	**45.24**	11.53
	3	40.81	22.66	**41.30**	11.53
	2	**39.60**	21.63	39.53	11.53
Serve	80	69.12	58.16	**76.07**	56.16
	50	68.97	55.66	**73.92**	58.38
	5	64.35	45.89	**65.43**	49.98
	3	62.62	45.78	**64.46**	46.64
	2	59.82	38.00	**59.98**	46.78
Line	80	26.83	15.78	**72.85**	11.62
	50	25.63	16.80	**71.83**	11.61
	5	25.31	19.08	**52.85**	11.61
	3	24.76	17.42	**47.46**	11.61
	2	23.37	16.29	**43.55**	11.62
Hard	80	29.58	39.23	**52.15**	47.54
	50	29.58	35.71	**52.87**	42.90
	5	30.44	29.58	**51.04**	32.98
	3	30.95	31.19	**44.30**	32.57
	2	30.32	30.13	**41.36**	30.95

training data, as seen in Table 7. Considering the value of human labeling, study areas such as few-shot intent detection in chatbots can benefit from our methodology immensely. Moreover, reduced label utilization is crucial for the machine translation task to or from low-resource languages. Our study proposes a novel graph algorithm that can be used in various text classification tasks with very scarce labels besides WSD.

The current architecture relies heavily on the heuristically defined threshold value for removing the weak links. This is possibly the main bottleneck of our system regarding time and resource utilization. A theoretical approach based on the graph structure and diffusion process to replace the current heuristic thresholding would drastically improve the design. Along with hyperparameter optimization, this is the current direction of our research. Such an improvement in the design will make the training faster and the integration of our system to other systems, such as machine translation models, more effortless. On the hand, it will give us more information on the topology of the data.

Our observation suggests that the accuracy and F1 Score changes seem to be very little for the dataset Hard compared to the other datasets. The possible reason for that might be the number of different senses for the word hard is the smallest compared to the others, and sense distribution in the data is very skewed, as seen in Fig 1. Other than that, the F1 score differences of DIFF NORM GCN in Table 7 compared to conventional Text GCN for label ratio 2% is between circa 6% - 12% for all datasets excluding the Hard dataset, which is very significant. Considering machine translation as a two-stage model with WSD and word mapping. The propagated error that uses Text GCN instead of DIFF NORM GCN will possibly be 6% - 12% less accurate. The error rate would possibly increase significantly for a word such as 'interest' with five different meanings. On the other hand, how WSD models are integrated into the system matters as much as the error rates of WSD systems and the architecture of the systems. A more detailed study of how a WSD model affects a machine translation tool or any other NLP task precisely is beyond the scope of this paper, and it is a possible future work. For further information on how WSD affects machine translation, we recommend a very detailed study [45] to the reader.

For hyperparameter tuning, our system limitations forced us to use a relatively small search space and a single random seed because of a longer experimental time. Conversely to our experimental setup, it has to be said that using more than one random seed is a better way for experiments for hyperparameter tuning. Additionally, our hyperparameter search space was not the absolute one. Different spaces could be used for searching also. Therefore, doing experiments for hyperparameter searching using multiple random seeds and different search spaces is one of our possible future works.

J. SYSTEM SPECS FOR EXPERIMENTS

All computations were done on the Pro and Pro+ versions of Google Colab[3] systems. Tesla T4 16 GB[4] and A100-SXM4 40 GB[5] GPUs were used for computations. Computations were done by using Pytorch[6] with its CUDA Module.[7] The Nltk[8] library was used to clean datasets. Datasets were split as stratified by using the Sklearn[9] library. Many matrix operations were done by using Numpy.[10] For reproducibility, random seed values were used with convenient random seed functions that belong to Pytorch[6], the standard random library of Python,[11] Pytorch Lightning,[12] and Numpy[10] libraries, before splitting datasets and creating models.

In experiments with Diffusion GCN/GAT models, threshold search, hyperparameter tuning, and all experiments were done using Pytorch Lightning[12]. Pytorch Geometric [47] was used to create models and data objects.

In experiments with the Text GCN algorithm, models were created using Pytorch[6]. CUDA Module[7] of Pytorch[6] was used for computations.

For all experiments, source codes are available on our repository.[13]

V. CONCLUSION AND FUTURE DIRECTIONS

In NLP, WSD is the process of identifying the meaning of a word considering the context in which the word is used. Considering the limited number of studies on GAT and GCN for WSD, we attempt to develop an improved version of GNN architecture with regularization, normalization, thresholding effect, and diffusion process for enhancing the performance of the WSD task. We proposed two architectures to merge the semantic diffusion process with the GCN and Text GCN algorithms.

We conducted comprehensive experiments on four benchmark sub-datasets of the SensEval dataset: Interest, Line, Hard, and Serve. The accuracy and F1 scores from the experiments, shown in the tables, motivated us to believe that the enhanced GAT, GCN, and Text GCN algorithms with the diffusion process could increase the classification performance on WSD tasks. These results are consistent with the results Gastgeiger et al. presented in [20].

In the future, to make further analysis, we will perform our experiment on larger datasets. We plan to conduct the suggested system in Turkish WSD corpora. Furthermore, building an ensemble learning classifier including these GAT and GCN algorithms may be a good idea. Another item on our agenda is to develop different normalization and regularization schemas in our classifier algorithm. Exploring the effect of the diffusion process on different graph structures, such as bipartite graphs or complex network settings, can be a good future work direction. Additionally, for hyperparameter tuning, we plan to do more comprehensive experiments with multiple random seeds and larger search spaces.

[3]https://colab.research.google.com/
[4]https://www.nvidia.com/en-us/data-center/tesla-t4/
[5]https://www.nvidia.com/en-us/data-center/a100/
[6]https://pytorch.org/
[7]https://pytorch.org/docs/stable/cuda.html
[8]https://www.nltk.org/
[9]https://scikit-learn.org/
[10]https://numpy.org/
[11]https://www.python.org/
[12]https://www.pytorchlightning.ai/
[13]https://github.com/eren-elma/diffusion-experiments

ACKNOWLEDGMENT

Points of view in this document are those of the authors and do not necessarily represent the official position or policies of TÜBİTAK.

REFERENCES

[1] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, "Introduction to WordNet: An on-line lexical database," *Int. J. Lexicogr.*, vol. 3, no. 4, pp. 235–244, 1990.
[2] R. Navigli, "Word sense disambiguation: A survey," *ACM Comput. Surv.*, vol. 41, no. 2, pp. 1–69, 2009.
[3] E. Agirre and P. Edmonds, *Word Sense Disambiguation: Algorithms and Application*, vol. 33. Berlin, Germany: Springer, 2007.
[4] N. Ide and J. Véronis, "Introduction to the special issue on word sense disambiguation: The state of the art," *Comput. Linguist.*, vol. 24, no. 1, pp. 1–40, 1998.
[5] S. Banerjee and T. Pedersen, "Extended gloss overlaps as a measure of semantic relatedness," in *Proc. IJCAI*, vol. 3, 2003, pp. 805–810.
[6] P. Basile, A. Caputo, and G. Semeraro, "An enhanced Lesk word sense disambiguation algorithm through a distributional semantic model," in *Proc. 25th Int. Conf. Comput. Linguistics, Tech. Papers*, 2014, pp. 1591–1600.
[7] J. Boyd-Graber, D. Blei, and X. Zhu, "A topic model for word sense disambiguation," in *Proc. EMNLP-CoNLL*, 2007, pp. 1024–1033.
[8] Y. Wang, M. Wang, and H. Fujita, "Word sense disambiguation: A comprehensive knowledge exploitation framework," *Knowl.-Based Syst.*, vol. 190, Feb. 2020, Art. no. 105030.
[9] S. Kwon, D. Oh, and Y. Ko, "Word sense disambiguation based on context selection using knowledge-based word similarity," *Inf. Process. Manag.*, vol. 58, no. 4, 2021, Art. no. 102551.
[10] I. Iacobacci, M. T. Pilehvar, and R. Navigli, "Embeddings for word sense disambiguation: An evaluation study," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 897–907.
[11] S. Papandrea, A. Raganato, and C. D. Bovi, "SupWSD: A flexible toolkit for supervised word sense disambiguation," in *Proc. Conf. Empirical Methods Natural Lang. Process., Syst. Demonstrations*, 2017, pp. 103–108.
[12] M. E. Peters, "Deep contextualized word representations," in *Proc. NAACL HLT*, 2018, pp. 2227–2237.
[13] O. Melamud, J. Goldberger, and I. Dagan, "context2vec: Learning generic context embedding with bidirectional LSTM," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.*, 2016, pp. 51–61.
[14] C.-X. Zhang, R. Liu, X.-Y. Gao, and B. Yu, "Graph convolutional network for word sense disambiguation," *Discrete Dyn. Nature Soc.*, vol. 2021, pp. 1–12, Sep. 2021.
[15] D. Le, M. Thai, and T. Nguyen, "Multi-task learning for metaphor detection with graph convolutional neural networks and word sense disambiguation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 8139–8146.
[16] G. Zhao, T. Wang, Y. Li, Y. Jin, and C. Lang, "Entropy-aware self-training for graph convolutional networks," *Neurocomputing*, vol. 464, pp. 394–407, Nov. 2021.
[17] D. Tufis, D. Cristea, and S. Stamou, "BalkaNet: Aims, methods, results and perspectives. A general overview," *Romanian J. Inf. Sci. Technol.*, vol. 7, nos. 1–2, pp. 9–43, 2004.
[18] R. Navigli and S. P. Ponzetto, "An overview of BabelNet and its API for multilingual language processing," in *Proc. People's Web Meets NLP, Collaboratively Constructed Lang. Resour.*, 2013, pp. 177–197.
[19] Z. Zhong and H. T. Ng, "It makes sense: A wide-coverage word sense disambiguation system for free text," in *Proc. ACL Syst. Demons.*, 2010, pp. 78–83.

[20] J. Gasteiger, S. Weißenberger, and S. Gunnemann, "Diffusion improves graph learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–13.
[21] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10684–10695.
[22] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. G. Lopes, B. K. Ayan, T. Salimans, and J. Ho, "Photorealistic text-to-image diffusion models with deep language understanding," 2022, *arXiv:2205.11487*.
[23] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models," 2021, *arXiv:2112.10741*.
[24] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 7370–7377.
[25] R. I. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete structures," in *Proc. 19th Int. Conf. Mach. Learn.*, 2002, pp. 315–322.
[26] T. Wang, W. Li, F. Liu, and J. Hua, "Sprinkled semantic diffusion kernel for word sense disambiguation," *Eng. Appl. Artif. Intell.*, vol. 64, pp. 43–51, Sep. 2017.
[27] T. Wang, J. Rao, and Q. Hu, "Supervised word sense disambiguation using semantic diffusion kernel," *Eng. Appl. Artif. Intell.*, vol. 27, pp. 167–174, Jan. 2014.
[28] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 8780–8794.
[29] G. Kocaman, B. Sipal, A. Gerek, B. Altinel, and M. C. Ganiz, "Diffused label propagation based transductive classification algorithm for word sense disambiguation," in *Proc. IEEE Int. Symp. Innov. Intell. Syst. Appl. (INISTA)*, Jul. 2019, pp. 1–6.
[30] P. Edmonds and S. Cotton, "SENSEVAL-2: Overview," in *Proc. 2nd Int. Workshop Eval. Word Sense Disambiguation Syst.*, 2001, pp. 1–5.
[31] Q. Yang, R. Li, Y. Li, and X. Gu, "Word sense disambiguation based on sequence topic model using sense dependency," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–8.
[32] S. Sousa, E. Milios, and L. Berton, "Word sense disambiguation: An evaluation study of semi-supervised approaches with word embeddings," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.
[33] N. Rahman and B. Borah, "An unsupervised method for word sense disambiguation," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 9, pp. 6643–6651, 2022.
[34] H. Kohli, "Transfer learning and augmentation for word sense disambiguation," in *Proc. Eur. Conf. Inf. Retr.*, 2021, pp. 303–311.
[35] J. M. Duarte, S. Sousa, E. Milios, and L. Berton, "Deep analysis of word sense disambiguation via semi-supervised learning and neural word representations," *Inf. Sci.*, vol. 570, pp. 278–297, Sep. 2021.
[36] H. Calvo, A. P. Rocha-Ramirez, M. A. Moreno-Armendariz, and C. A. Duchanoy, "Toward universal word sense disambiguation using deep neural networks," *IEEE Access*, vol. 7, pp. 60264–60275, 2019.
[37] A. M. Butnaru and R. T. Ionescu, "ShotgunWSD 2.0: An improved algorithm for global word sense disambiguation," *IEEE Access*, vol. 7, pp. 120961–120975, 2019.
[38] E. A. Correa, A. A. Lopes, and D. R. Amancio, "Word sense disambiguation: A complex network approach," *Inf. Sci.*, vols. 442–443, pp. 103–113, 2018.
[39] T. C. Silva and D. R. Amancio, "Word sense disambiguation via high order of learning in complex networks," *Europhys. Lett.*, vol. 98, no. 5, p. 58001, Jun. 2012.
[40] S. V. N. Vishwanathan, N. N. Schraudolph, I. R. Kondor, and K. M. Borgwardt, "Graph kernels," *J. Mach. Learn. Res.*, vol. 11, pp. 1201–1242, Mar. 2010.
[41] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, 2017, pp. 1–14.
[42] Y. Huang, Y. Weng, S. Yu, and X. Chen, "Diffusion convolutional recurrent neural network with rank influence learning for traffic forecasting," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./13th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2019, pp. 678–685.
[43] C. Vignac, I. Krawczuk, A. Siraudin, B. Wang, V. Cevher, and P. Frossard, "DiGress: Discrete denoising diffusion for graph generation," 2022, *arXiv:2209.14734*.
[44] M. Kim and J. Kim, "Collaborative filtering on bipartite graphs using graph convolutional networks," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Jan. 2022, pp. 304–307.
[45] Q.-P. Nguyen, A.-D. Vo, J. C. Shin, and C. Y. Ock, "Effect of word sense disambiguation on neural machine translation: A case study in Korean," *IEEE Access*, vol. 6, pp. 38512–38523, 2018.
[46] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. ICLR*, 2018, pp. 1–12.
[47] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *Proc. ICLR*, 2019, pp. 1–9.

BİLGE ŞİPAL SERT received the B.S. degree in computer science and mathematics from İstanbul Kültür University, İstanbul, Turkey, in 2003, the M.Sc. degree in mathematics from Koç University, İstanbul, in 2007, and the Ph.D. degree in symbolic computation (algorithmic algebraic geometry) from Passau University, Passau, Germany, in 2017.

From 2007 to 2014, she was a Research Assistant with the Chair of Symbolic Computation, Passau University. In 2018, she worked on the TÜBİTAK Project 1001 of Dr. M. C. Ganiz for text classification as a Postdoctoral Researcher. Between 2018 and 2020, she was an Assistant Professor with İstanbul Kültür University, and later, she switched to a Researcher with Afiniti, in 2020. Currently, she is the Lead Research Scientist of the NLP Group in the Decisioning Project of Afiniti.

Dr. Sert was a recipient 2022 H2 to Afiniti and Beyond Inventor Award with her patented work on the few-shot learning model.

EREN ELMA is currently pursuing the B.S. degree in computer engineering with Istanbul University—Cerrahpaşa, İstanbul, Turkey. He was a Research Intern on a project supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK), Marmara University, İstanbul. His research interests include computer vision, natural language processing, neural networks, and graph neural networks.

AYŞE BERNA ALTINEL received the B.S. degree in computer engineering from Yeditepe University, İstanbul, Turkey, in 2004, the M.S. degree in computer engineering from İstanbul Technical University, İstanbul, in 2007, and the Ph.D. degree in computer engineering from Yıldız Technical University, İstanbul, in 2016. She was a software and test engineer with Turkey İş Bank, between 2006 and 2009, and a software analyst and a researcher with the Scientific and Technological Research Council of Turkey (TÜBİTAK), between 2009 and 2012. She joined Marmara University, İstanbul in Spring 2013. Currently she is an Assistant Professor with the Computer Engineering Department, Faculty of Technology. Her research interests include textual data mining, natural language processing, social network analysis, machine learning, and bioinformatics. She is also the founder and the director of the Textual Data Analysis Research Laboratory (MetLAB), Computer Engineering Department, Faculty of Technology, Marmara University.