Knowledge-Based Systems 236 (2022) 107659

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Graph Fusion Network for Text Classification

Yong Dai [a], Linjun Shou [b], Ming Gong [b], Xiaolin Xia [a], Zhao Kang [a], Zenglin Xu [c,d,*], Daxin Jiang [b]

[a] SMILE Lab, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610031, China
[b] STCA NLP Group, Microsoft, Beijing, China
[c] School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China
[d] Center of Artificial Intelligence. Peng Cheng Lab, Shenzhen, Guangdong, China

ARTICLE INFO

Article history:
Received 22 June 2021
Received in revised form 22 August 2021
Accepted 28 October 2021
Available online 10 November 2021

Keywords:
Graph Neural Networks
Text classification
External knowledge
Graph fusion

ABSTRACT

Text classification is an important and classical problem in natural language processing. Recently, Graph Neural Networks (GNNs) have been widely applied in text classification and achieved outstanding performance. Despite the success of GNNs on text classification, existing methods are still limited in two main aspects. On the one hand, transductive methods cannot easily adapt to new documents. Since transductive methods incorporate all documents into their text graph, they need to reconstruct the whole graph and retrain their system from scratch when new documents come. However, this is not applicable to real-world situations. On the other hand, many state-of-the-art algorithms ignore the quality of text graphs, which may lead to sub-optimal performance. To address these problems, we propose a Graph Fusion Network (**GFN**), which can overcome these limitations and boost text classification performance. In detail, in the graph construction stage, we build homogeneous text graphs with word nodes, which makes the learning system capable of making inference on new documents without rebuilding the whole text graph. Then, we propose to transform external knowledge into structural information and integrate different views of text graphs to capture more structural information. In the graph reasoning stage, we divide the process into three steps: graph learning, graph convolution, and graph fusion. In the graph learning step, we adopt a graph learning layer to further adapt text graphs. In the graph fusion step, we design a multi-head fusion module to integrate different opinions. Experimental results on five benchmarks demonstrate the superiority of our proposed method.

1. Introduction

Text classification has been widely studied in many real-world applications, such as sentiment analysis [1–3], recommendation systems [4,5], community detection [6], and anomaly detection [7]. Over years, with the remarkable domain-expert-free feature-learning ability, various neural network models have been successfully applied to the text classification task. Among them, Recurrent Neural Networks (RNNs) [8], Convolutional Neural Networks (CNNs) [9,10], Transformer [11] and their variants [12] are representative deep learning paradigms. Specially, powered with a latent memory, RNNs are good at processing sequential data and widely adopted in natural language processing [13,14]. CNNs are able to exploit the shift-invariance, local connectivity, and compositionality of data, which make them popular in computer vision and natural language processing [15–17]. Some other works manage to combine RNNs and CNNs to capture local features and global features [18]. These models are proved to be effective for capturing hidden patterns in the Euclidean space.

Recently, there is an increasing number of applications where data are generated from non-Euclidean domains and represented in the form of graphs. For example, in e-commerce, users and product items can be considered as two kinds of nodes. Together with the relations between them, they are expressed in a heterogeneous graph. In the citation network, authors and their publications can be regarded as two kinds of nodes. Together with the author-to-publication, publication-to-author, publication-to-publication, and author-to-author relations, they are represented as a complex graph. However, previous neural networks cannot be applied to non-Euclidean domains directly. Driven by the strong practical application needs and the potential research value, graph neural networks (GNNs) have been widely studied and employed in real life [19–22]. For example, in community detection, GNNs are adopted to model complex graph structures

* Corresponding author at: School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China.
E-mail address: xuzenglin@hit.edu.cn (Z. Xu).

https://doi.org/10.1016/j.knosys.2021.107659
0950-7051/

between user nodes and their relations in an inherently convenient way [6]. In anomaly detection, GNNs are successfully applied to capture graph information to help identify anomalous graph objects (i.e., nodes, edges, and sub-graphs) [7].

When applied to text classification, GNNs allow the features of nodes at any location to directly attend to each other according to structural information (e.g., the relations between nodes). In detail, GNN-based methods first explicitly introduce the relations (i.e., adding edges) between words or terms in the graph construction stage. Then they encode this structural information into the learning process in the graph reasoning stage. For example, TextGCN [23] considers words and documents as two kinds of nodes to build a corpus-level heterogeneous text graph. [24] extends TextGCN and employs a tensor to model different graphs. This line of works trains their system in the transductive setting, which means they can exploit the mutual relationships between training and test examples [25]. Nonetheless, they need to pre-establish all documents (including test documents) in their graphs. This leads to the need of reconstructing the whole text graph and retraining the system from scratch when new documents come. This is inefficient and unrealistic when applied to real-world scenarios. To address this problem, [26] adopts the message passing framework [21] and mini-batch-based feature propagation mechanism. The drawback of this method is that it ignores the quality of the graph which could result in sub-optimal performance. To summarize, there remain two major limitations in GNN methods: (1) these methods are unable to easily adapt to new documents; (2) most of them ignore the quality of text graphs. These impede its wide application in practical scenarios. In this paper, we propose a Graph Fusion Network (GFN), which attempts to overcome these limitations and further boost system performance on text classification.

GFN consists of a graph construction stage and a graph reasoning stage. In the graph construction stage, GFN manage to overcome the two limitations mentioned above. In detail, For the first limitation, instead of pre-defining all documents as nodes in the text graph, GFN discards document-level nodes and builds pure word-level text graphs. To generate document embeddings, GFN fuses word embeddings according to the document-level structural information on the fly. In this way, the system does not need to reconstruct the text graphs and retrain the learning system when facing new documents. For the second limitation, it is known that constructing an ideal text graph, which can exactly capture all structural knowledge, is a nontrivial task, while constructing text graphs by different methods may contain different views of information [24,26]. Thus, GFN constructs different corpus-level graphs by transforming external knowledge (i.e., pre-calculated co-occurrence statistic and pre-trained embedding) as structural information, and integrate them to compensate each other.

In the graph reasoning process, GFN adopts three steps (i.e. graph learning, graph convolution, and graph fusion) to boost system performance. First, GFN adds a graph learning step to adjust the initialized edge weights to better serve the task. Then, GFN adopts the message passing mechanism [21] for graph convolution computing. At last, GFN applies a late-fusion paradigm (i.e. fuse the logits or final decision results), which consists of an elaborately designed multi-head-based fusion module, to integrate the results coming from each graph in the graph fusion step. Importantly, different from the early-fusion paradigm (i.e. feature-level fusion), late fusion brings in a number of advantages. First, it avoids the early cross-contamination due to noise and irrelevant information contained in each graph. Second, it inherits the robustness and empirical good performance of ensemble learning. Third, the late fusion paradigm retains diverse representation ability. The algorithm flow chart of GFN is shown in Fig. 1.

In sum, our contributions can be concluded as follows:

- To overcome the two limitations that remained in the GNN-based text classification task, in the graph construction stage, (1) we propose to exploit different external-knowledge-induced text graphs to better capture the structural information between words and across documents, and (2) we build homogeneous text graphs, which remove the dependency between the acquisition of document embedding and graph construction process, to make the system easily adapt to new documents.
- To further boost the system performance, we propose to adopt three steps in the graph reasoning stage: a graph learning step, a graph convolution step, and a fusion step. On the one hand, the learning ability in the graph learning step can derive more flexible and task-oriented graphs. On the other hand, the integration of multiple views of the graph information at the graph fusion stage can further boost the performance.
- This paper presents a unified Graph Fusion Network (**GFN**) for text classification. Extensive experiments on benchmark datasets validate the superiority of our framework.

The rest of this paper is organized as follows. Section 2 introduces the related work and its relation with our work. Section 3 demonstrates how to construct text graphs by incorporating external knowledge. Section 4 illustrates the reasoning process over the text graphs. Section 5 introduces the experimental datasets and implementation details and then elaborates on the experimental results. Section 6 concludes this paper and discusses future work.

2. Related work

In this section, we first discuss the role of two kinds of knowledge (i.e. co-occurrence statistic and word embedding) played in the previous works. Then, we review the recent text classification techniques, introduce the GNN-based method, and compare their relations with our work.

2.1. Role of co-occurrence statistic and word embedding

Within social sciences, word co-occurrence analysis is widely used in various forms of research concerning the domains of content analysis, text mining, construction of thesauri and ontologies, etc. [27–29]. In general, the aim of co-occurrence analysis is to find similarities in meaning between word pairs or similarities in meaning among/within word patterns, also to discover latent structures of text representations [30]. Based on the co-occurrence analysis, neural language models can derive low-dimensional and dense vectors (a.k.a. distributed representation) to implicitly represent the syntactic or semantic features of the language. For example, given a set of documents, [31] consider the word-to-word co-occurrence matrix as their objective and map each word to a vector in a relatively low dimensional space, such that the relationships between the meaning of the words roughly correspond to the spatial relationships of the vectors. The same adoptions can be seen in works by [32,33]. Also, word-to-word co-occurrences have been applied to traditional topic modeling tasks [34,35]. By re-weighting the co-occurrence matrix, Point-wise Mutual Information (PMI) [36] performs better in some situations. On the other hand, the pre-trained word embeddings, no matter trained based on the Language Modeling [31,37] or Masked Language Modeling [38,39], contain rich syntax and semantic relations among words [40,41].

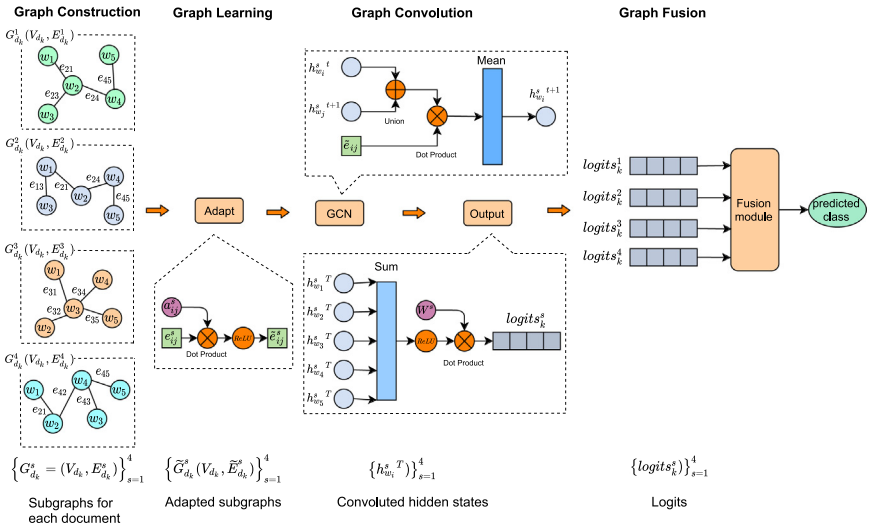

Fig. 1. The algorithm flow chart. We first build text graphs $\{G_{s,d_k}(V_{d_k}, E_{s,d_k})\}_{s=1}^{4}$ for each document d_k, then adjust the document-level text graphs to better serve the classification task, implement the graph convolution, and at last integrate all logits to obtain a better decision boundary.

2.2. Text classification approaches

Recent research on text classification tasks [42,43] makes use of neural networks have shown to outperform methods based on the bag-of-words model [12,44,45]. Specially, RNN, together with its popular variants LSTM [46,47] and GRU [48,49], considers information of previous nodes in a sophisticated way and allows for better semantic analysis of a data's structure. They are powerful methods for text, string, and sequential data classification [8, 50]. CNNs, which are originally built for image processing, have also been effectively used for text classification in a hierarchical manner [9,51]. Capsule networks [52], which can be powered with different routing mechanisms, are also attempted on this task [53,54]. Transformer, together with its variants, is also widely adopted depending on its powerful self-attention mechanism recently [11]. Apart from these single network architectures, hybrid models, which combine the merits of different kinds of neural models, are also studied. For example, [55] proposes a deep recurrent belief network to learning word dependencies in text. [56,57] employs hybrid networks to bring in more semantics or external knowledge. [58] combines reinforcement learning, generative adversarial networks, and recurrent neural networks to build a new model, which not only generates category sentences to enlarge the original dataset but also helps improve its generalization capability during supervised training.

Most recently, GNNs are introduced to natural language processing. They can derive more informative representation by capturing diverse structural information, which is verified beneficial in many applications [6,7,19,59,60]. The relations of a text graph may contain many types, such as word-to-word, word-to-document, or word-to-label relations [19]. Different kinds of relations can be exploited to advance the system performance in different ways. For example, [61] exploits a heterogeneous graph to build the correlation between labels and words, and obtain label-aware text representation for the multi-label text classification task. [23] first introduces the graph neural network to the multi-class text classification task and proposes TextGCN. TextGCN considers documents and words as two kinds of nodes and builds their relations into the text graph. In the graph reasoning process, two layers of GCN [20] are used to extract the document features. This method needs to pre-establish test documents into their text graph, which is impractical for real applications. To mitigate this situation, [26] abandons the document nodes and builds relations between words nearby. [62] introduces the gate mechanism to mitigate the over-smoothing problem. [63] unify the best aspects of PTE [64] and TextGCN [23] together to make their models efficient and inductive. Nevertheless, these works neglect the quality of text graphs. TensorGCN [24] builds different text graphs and adopts a tensor to model the different kinds of structural information, which is most similar to our method. The main differences include: (1) we specialize in the inductive setting, while they are in the transductive setting; (2) the text graph construction methods are different: ours are homogeneous, while theirs are heterogeneous; (3) we implement the late fusion paradigm (i.e. logit-level fusion), while they employ the early fusion (i.e. feature-level fusion).

3. Text graph construction

In this section, we present how to transform the syntactic and semantic information, which is contained in co-occurrence statistics and pre-trained word embedding, into structural information and construct text graphs. In the following, we first give out the problem settings and notations, then demonstrate our proposed graph construction methods.

3.1. Notations and settings

We represent text graphs as $\{G_s = (V, E_s)\}_{s=1}^{n_G}$, where $V(|V| = n)$ and E_s are node set and edge set of the corresponding text graph. $s \in [1, n_G]$ is the text graph index and $n_G = 4$, which implies we construct four kinds of corpus-level text graphs in our paper. V is composed of all words in the corpus. $e_{s,ij}$ (the element of E_s) denotes the relation between word w_i and w_j, which is calculated by metrics that are introduced in the following subsection. In addition, we denote the node feature matrix as $X \in \mathbb{R}^{n\times d}$ and a node feature vector as $x_i \in \mathbb{R}^d$ (i.e. the embedding of word w_i), where d is the feature dimension. There is an adjacency matrix $A_s \in \mathbb{R}^{n\times n}$ corresponding to each E_s, which includes all word-to-word relations. E_s can be regarded as a subset of A_s, which is filtered and retained under some conditions. For example, if the value of an element $a_{s,ij}$ in A_s, which is calculated by a certain metric, equals zero, this means the relation between word i and word j is weak and it will not be included into E_s.

The learning process of our system is to: (1) update each edge set E_s, elements of which are initialized and calculated by

formulations in the following section, (2) update node feature matrix X, and (3) update other network parameters. In practice, we build subgraphs $\left\{G_{s,d_k} = (V_{d_k}, E_{s,d_k})\right\}_{s=1}^{4}$ for each document d_k and update our system gradually and iteratively, where nodes in V_{d_k} are words within the document and edge set E_{s,d_k} is a subset of E_s.

3.2. Graph construction

In this subsection, we demonstrate how to construct each text graph $G_s(V, E_s)$ based on the following knowledge and four different metrics.

3.2.1. Graph construction based on the co-occurrence statistics

Co-occurrence statistics allow practitioners to understand what is being said in a given set of text documents at a high level, which is verified by many pre-train models, such as word2vec [32], GloVe [31], etc. Each entry of the word-to-word co-occurrence matrix is the maximum likelihood estimate (MLE) for the probability of a word pair (w_i, w_j) conditioned on the appearance of word w_i, which can be formulated as following [65]:

$$P_{ML}\left(w_j|w_i\right) = \frac{\#\left(w_i, w_j\right)}{\#\left(w_i\right)}, \tag{1}$$

which represents the relation of the two words (w_1, w_2) globally. In this formulation, $\#(w_i)$ is the number of occurrences of word w_i and $\#(w_i, w_j)$ is the number of times two words appear together in a pre-defined and fix-sized window L. As mentioned before, word co-occurrence analysis finds similarities in meaning between word pairs, so we let $e_{ij} = P_{ML}\left(w_j|w_i\right)$, of which the value is bigger the relation is closer.

3.2.2. Graph construction based on positive point-wise mutual information

Introduced by [36], Point-wise Mutual Information (PMI) is widely adopted for word similarity tasks [66–68]. It can be estimated empirically by considering the actual number of observations in a corpus [33]:

$$PMI(w_i, w_j) = \log \frac{\#(w_i, w_j) \cdot |D|}{\#(w_i) \cdot \#(w_j)}, \tag{2}$$

where D is the collection of word pairs and $|D|$ is the count of the word pairs. $PMI(w_i, w_j)$ measures the association between the word pair (w_i, w_j) by calculating the log of the ratio between their joint probability (the frequency in which they occur together) and their marginal probabilities (the frequency in which they occur independently). The negative PMI often means a weak association between words, so the positive PMI (PPMI) metric was adopted:

$$PPMI(w_i, w_j) = \max(PMI(w_i, w_j), 0). \tag{3}$$

PPMI metric can be considered as the re-weighted variant of the co-occurrence metric [36]. In some scenarios, PPMI performs better, like semantic similarity tasks [28]. So we can set $e_{ij} = PPMI(w_i, w_j)$.

3.2.3. Graph construction based on the cosine similarity of pre-trained embedding

A good representation could capture the implicit linguistic rules and common sense knowledge hiding in text data, such as lexical meanings, syntactic structures, semantic roles, and even pragmatics [31,32,38]. So another reasonable way to measure the relationships between words is to leverage the pre-trained word embedding. To distill knowledge from word embeddings and measure relations between word pairs, we choose the cosine similarity and Euclidean distance as two kinds of metrics, because the cosine similarity and Euclidean distance have already been widely adopted to measure sentence pair relations and word pair relations [43,69–71]. Euclidean distance focuses more on the magnitude differences while the cosine similarity focuses more on the orientation (the angle).

To be specific, the cosine similarity of a word pair is formulated by:

$$cos(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{|\mathbf{x}_i| \cdot |\mathbf{x}_j|}, \tag{4}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ is the embedding of word w_i and w_j separately. The bigger the value is, the more close the two words are.

3.2.4. Graph construction based on the euclidean distance of pre-trained embedding

The Euclidean distance of a word pair is formulated as:

$$euc(\mathbf{x}_i, \mathbf{x}_j) = \left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2. \tag{5}$$

3.2.5. Building subgraphs for each document

We can construct four kinds of corpus-level text graphs based on the four metrics. To avoid the noise from irrelevant relations, we exclude the relations with negative or small values calculated by the four metrics.

Considering the memory consumption, we train GFN on a mini-batch of documents, not the whole documents every step. For this, we build subgraphs $\left\{G_{s,d_k} = (V_{d_k}, E_{s,d_k})\right\}_{s=1}^{4}$ for each document d_k. The node set V_{d_k} is composed of m words within d_k and k is the document index. The edge set E_{s,d_k} is fetched from E_s. In addition, we add an edge between w_i and w_j to G_{s,d_k} when w_j is in the p-neighborhood of w_i, because this kind of information can benefit the representation of the central word w_i [72,73]. As a whole, document-level subgraphs include two kinds of relations: the corpus-level relations and the neighborhood relations.

4. Graph reasoning

In this section, we show the reasoning process of GFN. GFN reasons over document-level subgraphs through the following three steps: graph learning, graph convolution, and graph fusion.

4.1. Graph learning

We assume the pre-calculated E_s may be not optimal, it needs to be adapted according to the downstream task. For an edge $e_{s,ij}$ between w_i and w_j, the learning process can be formulated as:

$$\tilde{e}_{s,ij} = \mathrm{ReLU}\left(a_{s,ij} \cdot e_{s,ij}\right), \tag{6}$$

where $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$ and $a_{s,ij}$ is a learnable parameter. ReLU guarantees the non-negativity of $\tilde{e}_{s,ij}$. Through Eq. (6), we can derive adapted text subgraphs $\tilde{G}_{s,d_k}(V_{d_k}, \tilde{E}_{s,d_k})$ for document d_k.

4.2. Graph convolution

After the graph learning, we adopt the message passing mechanism [21] to aggregate information from neighbors and update node features by absorbing the aggregated information. If the message passing phase runs for T time steps, the aggregation function of the intermediate time step can be formulated as:

$$\mathbf{m}_{s,i}^{t+1} = \frac{1}{|Nei(w_i)|} \sum_{j=1}^{|Nei(w_i)|} \left(\tilde{e}_{s,ij} \cdot \mathbf{h}_{s,j}^{t+1}\right), \tag{7}$$

where $|Nei(w_i)|$ is the number of w_i's neighbors, $\mathbf{h}_{s,j}^{t+1} \in \mathbb{R}^{1\times d_h}$ is the intermediate hidden state of w_i's neighbors, j is the index of

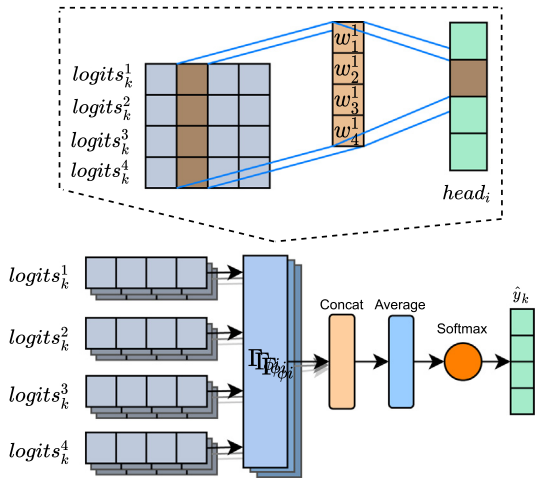

Fig. 2. Graph fusion process. Each head takes in four logits, which represent different views of the unnormalized probability belonging to each class, and outputs the integrated decision. Multiple heads obtain different decisions, which will be further averaged and normalized by the softmax function.

w_i's neighborhood words and d_h is the dimension of the hidden state.

The hidden state $\mathbf{h}_{s,i}^{t+1} \in \mathbb{R}^{1\times d_h}$ of each node w_i is updated by absorbing the neighborhood message according to:

$$\mathbf{h}_{s,i}^{t+1} = \mathbf{h}_{s,i}^{t} + \mathbf{m}_{s,i}^{t+1}. \tag{8}$$

Eq. (8) means $\mathbf{h}_{s,i}^{t+1}$ (the hidden state of w_i) consists of the last time's hidden state and the message from its neighbors. After the word-level feature updating, we can obtain the representation and the last logits (i.e. the unnormalized probabilities before softmax) for each document by:

$$\mathbf{d}_{s,k} = \frac{1}{|d_k|}\sum_{i=1}^{|d_k|}(\mathbf{h}_{s,i}^{T}|\forall w_i \in d_k), \tag{9}$$

$$\mathbf{logits}_{s,k} = \mathrm{ReLU}(\mathbf{d}_{s,k})\mathbf{W}_s, \tag{10}$$

where $\mathbf{d}_{s,k} \in \mathbb{R}^{1\times d_h}$, $\mathbf{W}_s \in \mathbb{R}^{d_h\times c}$, $\mathbf{logits}_{s,k} \in \mathbb{R}^{1\times c}$ and c denotes the number of classes. Eq. (9) means $\mathbf{d}_{s,k}$ (the embedding of a document) is obtained by averaging the last hidden states of all the words within the document d_k. Eq. (10) projects $\mathbf{d}_{s,k}$ to the label space to obtain the final logits $\mathbf{logits}_{s,k}$.

4.3. Graph fusion

The subscript $s(s \in [1, 4])$ means that there are 4 views of opinions for each document. To integrate different views and search for a consistent decision boundary, we design a fusion module. In detail, we first concatenate $\mathbf{logits}_{s,k}$:

$$\mathbf{logits}_k = \mathop{\Vert}_{s=1}^{4} \mathbf{logits}_{s,k}, \tag{11}$$

where $\Vert$ is the concatenation operator and $\mathbf{logits}_k \in \mathbb{R}^{1\times c\times 4}$. Inspired by [11,74], we develop a multi-head variant as the fusion operator to make good use of the model's capacity, which can be formulated as:

$$\mathrm{head}_i = \Gamma_{\phi_i}(\mathbf{logits}_k), \tag{12}$$

where Γ_{ϕ_i} is a fusion operator parametrized by ϕ_i and $\mathrm{head}_i \in \mathbb{R}^{1\times c}$ is ith $(i \in [1, |h|])$ head of fused opinions. The purpose of the fusion operator Γ_{ϕ_i} is to fuse 4 different opinions into a consistent opinion, which is key to the performance of the entire system. In practical terms, we use a convolutional layer with a kernel size $[4, |h|, 1]$ to aggregate different views of opinions. The convolution behaves like a weighted pooling that reduces the multi-view opinions into a single decision.

Then, $|h|$ heads of decisions are averaged to generate the final decision for document d_k:

$$\widehat{\mathbf{logits}}_k = \frac{1}{|h|}\sum_{i=1}^{|h|}\mathrm{head}_i, \tag{13}$$

$$\hat{\mathbf{y}}_k = \mathrm{Softmax}\left(\widehat{\mathbf{logits}}_k\right), \tag{14}$$

where $|h|$ is the number of heads, $\widehat{\mathbf{logits}}_k \in \mathbb{R}^{1\times c}$, and $\hat{\mathbf{y}}_k \in \mathbb{R}^{1\times c}$. We break down the fusion process and draw it in Fig. 2.

At last, we minimize the cross-entropy loss between the ground truth label $\mathbf{y}_k$ and the predicted label $\hat{\mathbf{y}}_k$ for model optimization:

$$\mathcal{L} = -\sum_{k}\mathbf{y}_k \log \hat{\mathbf{y}}_k. \tag{15}$$

5. Experiments

In this section, we evaluate GFN on text classification task and answer the following questions:

(1) Can GFN achieve superior performance when compared with state-of-the-art models? (Section 5.2)
(2) How is the effectiveness of each text graph? (Section 5.3)
(3) How is the effectiveness of the fusion module? (Section 5.4)
(4) How many heads are enough for the system? (Section 5.5)
(5) Is our model memory- and time-efficient? (Section 5.6)
(6) How does our system make mistakes? (Section 5.7)

5.1. Experimental setup

To comprehensively evaluate the performance of GFN, we did two groups of experiments on 5 benchmark datasets. One group of experiments are evaluated on the accuracy, and another group of experiments is evaluated on Micro-F1 and Macro-F1 scores [75]. Accuracy emphasizes true positives and true negatives, while F1-score emphasizes false negatives and false positives. Micro-F1 computes F1 by considering total true positives, false negatives, and false positives. Macro-F1 computes F1 for each label and returns the average without considering the proportion for each label in the dataset. These metrics measure different aspects of system performance.

In this subsection, we first introduce datasets, then give out the baselines for different groups of experiments separately, and at last specify the implementation details.

5.1.1. Datasets

We ran our experiments on five widely adopted benchmark datasets:

- **20-Newsgroups (20NG)**[1] is a collection of 18846 newsgroup documents.
- **Ohsumed (Oh)**[2] is a bibliographic database of important medical literature;
- **R8** and **R52**[3] are two subsets of the Reuters dataset with 8 and 52 categories respectively;
- **Movie Review** (MR)[4] is a movie review dataset for binary sentiment.

[1] http://qwone.com/jason/20Newsgroups/.
[2] http://disi.unitn.it/moschitti/corpora.htm.
[3] https://www.cs.umb.edu/smimarog/textmining/datasets/.
[4] https://github.com/mnqu/PTE/tree/master/data/mr.

Table 1
Statistics of datasets.

Dataset	#Docs	#Train	#Dev	#Test	#Words	#Frequent	#Classes	Average Length
20NG	18,846	10,183	1,131	7,532	16,375	10	20	221.26
R8	7,674	4,937	548	2,189	2,923	5	8	65.72
R52	9,100	5,879	653	2,568	5,364	5	52	69.82
Oh	7,400	3,022	335	4,043	7,998	5	23	135.82
MR	10,662	6,398	710	3,554	13,904	1	2	20.39

Following [23], we pre-process all documents by removing stop words and lemmatizing all words by NLTK.[5] The statistics of preprocessed datasets are summarized in Table 1.

5.1.2. Baselines

The first group of baselines comparing with GFN on accuracy are classified into three classes, which including sequence deep learning models, word embedding based methods, and graph convolutional methods.

- **CNN-rand** and **CNN-non-static** [76] are two kinds of Convolutional Neural Networks with randomly initialized word embeddings and pre-trained word embeddings.
- **LSTM** is defined in [1] and commonly used in text classification. LSTM considers the last hidden state as the representation of the whole text.
- **Bi-LSTM** is a bi-directional LSTM, which can exploit two directions of text information.
- **PV-DBOW** and **PV-DM**[77] are Paragraph Vector models, one of which considers the word order and one not. We used Logistic Regression as the classifier.
- **fastText** [78] averages the ngram word embedding as their document embedding and adopts the linear classifier.
- **PTE** [64] learns the word embedding by building a heterogeneous text network and then averages the whole word embedding as the document embedding.
- **LEAM** [79] embeds the labels and words in the same embedding space.
- **SWEM** [80] employs simple pooling strategies over word embeddings.
- **Graph-CNN-C/S/F** are Graph CNN models that adopt Chebyshev (C) [81], Spline (S) [82] and Fourier (F) [83] filter, respectively.
- **TextGCN** [23] is a graph-convolution-based model. It considers documents and words as their nodes and builds a heterogeneous graph to capture the structure information.
- **TLGCN**[6] [26] is a graph convolution based mode which is memory-friendly.

For the second group of experiments, we follow the experimental setting of HETEGCN [63] for a fair comparison. The baseline methods (not including methods that have been introduced above) are as follows:

- **LR** (Logistic Regression model) is a model that trained on the TF–IDF transformed word representation.
- **HETEGCN** [63] consists of a series of models, which adopt different text graphs and neural layers to encode features. These models include (F-X), (X-TX-X), (X-TX-X)*, (TX-X), and (TX-X)*.

5.1.3. Implementation details

For the first series of experiments, we reproduced results of TLGCN based on the open-source code and other results are the same as the results in [23,24,63]. For the second series of experiments, we followed the most setting of HETEGCN in the large labeled data scenario and other main results are taken from their paper. In the following, we specify the training details.

Training acceleration. Inspired by the merit of the pre-training paradigm, we adopt a two-stage training paradigm to speed up the training process. In detail, we first train each sub-system (i.e. each row-wise system in Fig. 1 without fusion module) in parallel using independent classifiers. Then, we fix the parameters of each sub-system and fine-tuning other parameters. Since no storage of gradient-related values is needed for each sub-system, the memory and time consumption is trivial. Benefiting from this pre-training and fine-tuning practice, the system can converge in a few iterations.

Parameter searching. To keep the balance of the vocabulary size and the semantic loss of documents, we set a different threshold for each dataset to discard seldom appeared words and build the vocabulary. In detail, we keep all the words for MR, because every example of this dataset has just one sentence and discarding any word may lose its semantic. We set a threshold of 10 for 20NG and 5 for other datasets. Following the setting of [23], we set window-size L 20 to obtain the co-occurrence statistic. To derive embedding-based distances, we adopt 300d GloVe[7] [31] as our word embeddings and set hidden state size to 300. To avoid introducing too much noise, we roughly retain $min(0.005 * n, 100)$ (n is the size of vocabulary) number of edges for each word. To train each sub-system, we employ AdamW [84] with the default parameter setting and stop training if the validation loss does not decrease for 10 consecutive epochs. To train the fusion module, we reset the learning rate to 0.05 and keep other parameters default. If the validation loss does not decrease for 100 iterations, we stop training. For other parameters searching, we adopt Neural Network Intelligence (NNI)[8] to implement grid searching. We search the dropout rate in [0.3, 0.5, 0.7], batch size in [16, 32, 64, 128] and P in [1, 3, 5, 7, 9]. In terms of code implementation, we employ Deep Graph Library (DGL)[9] to realize graph reasoning and scikit-learn.[10] to measure the Micro-f1 and Macro-f1 At last, all results reported in this study are the mean values of three independent runs with different random initializations.

5.2. Main results

In this subsection, we analyze the performance of GFN with two groups of baselines on five benchmark datasets. The results which are tested on accuracy are presented in Table 2. As a whole, the performance of GFN is consistently superior to other baselines. In detail, GFN obtains 0.8701, 0.7020, and 0.7804 on 20NG, Oh, and MR respectively, which are obvious boosts compared with the previous works. The improvements on R8 and R52 are not so obvious, we suppose this is because: (1) the task on these datasets is simpler than the task on other datasets, so they

5 http://www.nltk.org/
6 https://github.com/HuangLianzhe/TextLevelGCN.
7 http://nlp.stanford.edu/data/glove.6B.zip.
8 https://github.com/microsoft/nni.
9 https://docs.dgl.ai/index.html.
10 https://scikit-learn.org/stable/.

Table 2
Test performance comparison with baselines measured by accuracy.

Model	20NG	R8	R52	Oh	MR
CNN-rand	0.7693	0.9402	0.8537	0.4387	0.7498
CNN-non-static	0.8215	0.9571	0.8759	0.5844	0. 7775
LSTM	0.7543	0.9609	0.9048	0.5110	0. 7733
Bi-LSTM	0.7318	0.9631	0.9054	0.4927	0. 7768
PV-DBOW	0.7436	0.8587	0.7829	0.4665	0. 6109
PV-DM	0.5114	0.5207	0.4492	0.2950	0. 5947
FastTest	0.7938	0.9613	0.9281	0.5770	0. 7514
PTE	0.7674	0.9669	0.9071	0.5358	0. 7023
LEAM	0.8191	0.9331	0.9184	0.5858	0.7695
SWEM	0.8516	0.9532	0.9294	0.6312	0. 7665
Graph-CNN-C	0.8142	0.9699	0.9275	0.6386	0. 7722
Graph-CNN-S	/	0.9680	0.9274	0.6282	0. 7699
Graph-CNN-F	/	0.9689	0.9320	0.6304	0. 7674
TextGCN	0.8634	0.9707	0.9356	0.6836	0. 7674
TLGCN	0.8592	0.9769	0.9441	0.6914	0. 7638
GFN	**0.8701**	**0.9822**	**0.9531**	**0.7020**	**0. 7804**

Table 3
Test performance comparison with baselines measured by Micro-F1 and Macro-F1. The models with * denote that HETEGCN adopts normalized transpose of X. The models without * denote HETEGCN adopt transpose of normalized X.

Model	20NG	R8	R52	Oh	MR
LR	0.8476	0.9726	0.9338	0.6587	0.7701
PTE	0.8437	0.9466	0.9065	0.6	0.7107
GCN	0.7632	0.941	0.9041	0.6223	0.7413
HETEGCN (F-X)	**0.8715**	0.9724	0.9435	0.6811	0.7671
HETEGCN (X-TX-X)	0.8439	0.9709	0.9205	0.6144	0.7548
HETEGCN (X-TX-X)*	0.8625	0.9733	0.9321	0.6667	0.7756
HETEGCN (TX-X)	0.8418	0.9698	0.9287	0.6614	0.7521
HETEGCN (TX-X)*	0.8660	0.9746	0.9376	0.6894	0.7648
GFN (Micro-F1)	0.8687	**0.9819**	**0.9517**	**0.701**	**0.7784**
LR	0.8409	0.9332	0.682	0.5377	0.7101
PTE	0.8370	0.8691	0.5921	0.5367	0.7106
GCN	0.7607	0.8529	0.6177	0.5487	0.7412
HETEGCN (F-X)	**0.8659**	0.9295	0.6842	**0.6062**	0.7671
HETEGCN (X-TX-X)	0.8386	0.9099	0.5275	0.4823	0.7546
HETEGCN (X-TX-X)*	0.8563	0.9340	0.5661	0.5799	0.7755
HETEGCN (TX-X)	0.8371	0.9184	0.5769	0.5852	0.7520
HETEGCN (TX-X)*	0.8600	0.9390	0.6533	0.6184	0.7647
GFN (Macro-F1)	0.8628	**0.9547**	**0.7458**	0.6025	**0.7780**

Table 4
The effectiveness of each text graph.

Graph	20NG	R8	R52	Oh	MR	Avg.
pmi	**0.8673**	**0.9812**	**0.9492**	0.6942	0.7707	0.8525
cos.	0.8623	0.9793	**0.9492**	0.6922	0.7719	0.8510
euc.	0.8653	0.9811	0.9488	**0.6954**	**0.7744**	**0.8530**
co.	0.8620	0.9775	0.9465	0.6927	0.7659	0.8489

have no strong requirement for a complex feature encoder. (2) the accuracy on these two datasets is too high to be improved. The results on Micro-F1 and Macro-F1 are presented in Table 3, from which we can also observe an obvious boost comparing with baselines. We suppose our superior mainly locate in two aspects, one is the accommodation of better text graphs, another is the fusion of multiple graphs, which will be interpreted in the following subsections.

5.3. The effectiveness of each text graph

In this subsection, we explore the effectiveness of each text graph. For this, we take turns to accommodate GFN with every single text graph (i.e. PMI, cosine similarity, Euclidean distance, and co-occurrence) and test the system performance without the fusion module. System performances by adopting different graphs (denoted "*pmi*", "*cos.*", "*euc.*", and "*co.*") are presented in Table 4. We can conclude that the structural information captured by each text graph is different, which leads to different performances. GFN powered with PMI-induced text graphs and Euclidean-distance-induced text graphs performs good, while GFN with co-occurrence-induced graphs performs comparably poor.

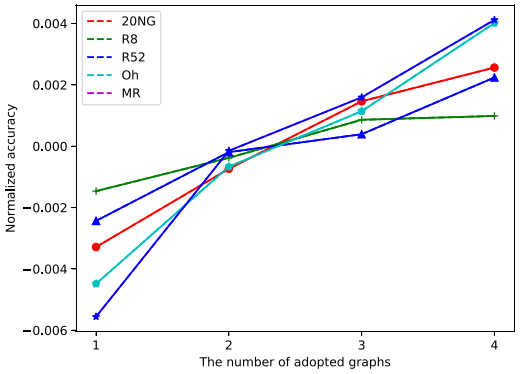

Fig. 3. Analysis of fusion performance. Each line demonstrates the trend by fusing one/two/three/four graphs on each dataset.

Table 5
Performance analysis with different heads.

Kernel_size	20NG	R8	R52	Ohsumed	MR	Avg.
1	0.8670	**0.9822**	0.9513	0.6994	**0.7804**	0.8561
2	**0.8701**	0.9819	0.9520	0.7018	0.7786	0.8569
3	0.8697	0.9819	**0.9531**	**0.7020**	0.7792	**0.8572**
4	0.8689	0.9805	0.9513	0.6994	0.7767	0.8554

5.4. Analysis of fusion performance

In this subsection, we examine the performance of fusing a different number of graphs.

For this, we take different combinations of text graphs to accommodate GFN and evaluate performance. The results of GFN powered with the different number of graphs are averaged as a result for each dataset. To prevent the uneven distribution of values from affecting our description of the trend, we subtracted the mean accuracy from all values to get the normalized accuracy. Take 20NG for example, the obtained results are 0.86425, 0.8668, 0.869, and 0.8701, which denote the results of fusing one, two, three, and four text graphs. These results are subtracted by their mean accuracy of 0.8675325. The final normalized results are −0.0032875, −0.0007375, 0.0014625, and 0.0025625. At last, we obtain four sets of results, which are drawn as a trend graph in Fig. 3, for all datasets.

From Fig. 3, we can observe: (1) a clear upward trend, which indicates that the more the number of fused graphs, the better the system performance. (2) a gradually slowing upward trend, which is caused by the information overlap between graphs we suppose.

5.5. Analysis of multi-head mechanism

In this subsection, we evaluate how many heads are enough for our system. We did experiments using GFN with different heads. The results are shown in Table 5, from which we can see that GFN performs best by adopting three heads.

Fig. 4. Error examples from test set on MR.

Table 6
The time- and memory-consumption.

	Model	R8	R52	Oh
Memory consumption (MiB)	TLGCN	623	697	721
	GFN	908	907.5	1195
Time consumption (S)	TLGCN	839	1175	1685
	GFN	907	1572	1833

5.6. Analysis of efficiency of our system

As described before, we train our system in two stages to accelerate the training speed. In this subsection, we compare the memory consumption and time consumption of GFN with TLGCN, which is more efficient comparing with other GNN-based text classification models. Results on three datasets are shown in Table 6. From the memory consumption perspective, our system is slightly heavy due to the adoption of multiple graphs. From the time consumption perspective, our system is as fast as TLGCN due to the adoption of the two-stage training paradigm.

5.7. Error analysis

In this subsection, we analyze how does our system makes mistakes? To this end, we did the error analysis on MR. The task on MR is to predict the sentimental polarity (i.e. positive or negative) of comments for movies, which is easy to understand and illustrate. We randomly sampled 50 error cases from the test set for analysis. Through our analysis, we roughly categorize the error into five types.

- Misfocusing. It means one expresses two or more polarities in one sentence simultaneously, and the system often focuses more on the wrong part of the sentence and gives out a wrong prediction. This type of error roughly accounts for 52% of the total error cases.
- Irony. This means the literal meaning of a sentence is positive, but what one actually expresses the negative sentiment. This type of error roughly accounts for 10% of the total error cases.
- Confusion. It means it is confused to infer the sentiment polarity of a comment for a system or even for a human. This type of error roughly accounts for 8% of the total error cases.

- Negative prefix. This means that the system cannot recognize the emotional reversal brought about by the negative prefix. This type of error roughly accounts for 5% of the total error cases.
- Others. This type of error roughly accounts for 25% of the total error cases and we cannot find the reason why GFN makes mistakes.

We provide some examples for each type of error in Fig. 4. It is worth noting that, there are about 25% of cases in the "*Others*" error type that GFN makes wrong predictions, but a human can discriminate their sentimental tendency.

6. Conclusion and future work

In this paper, we proposed a Graph Fusion Network (**GFN**), which supports the efficient inference for the new documents without retraining the system and can better capture structural information by integrating different views of text graphs. Experimental results illustrate the superiority of our proposed method. Specially, different views of graphs are complementary and the elaborately designed multi-head fusion module can further boost the system performance.

In the future, we think that the interpretability of deep learning models is worthy of in-depth study, which will make deep learning systems more robust and believable. GFN transforms two kinds of knowledge into structural information to capture different aspects of information, but the problem of constructing an ideal text graph is also unsolved. We may continue to explore other methods to construct text graphs with more information or automatically. On the other side, the pre-trained models (e.g., BERT [38]) have dominated many fields including text classification due to their outstanding performance, while GNN-based models also have their advantages. So, the comprehensive comparative analysis between BERT-style models and GNN-based models is valuable.

CRediT authorship contribution statement

Yong Dai: Conceptualization, Methodology, Software, Investigation, Writing – original draft, Formal analysis, Validation. **Linjun Shou:** Supervision, Writing - review & editing, Resources. **Ming Gong:** Project administration, Funding acquisition. **Xiaolin Xia:** Software, Investigation, Visualization. **Zhao Kang:** Writing - review & editing. **Zenglin Xu:** Supervision, Writing – review & editing. **Daxin Jiang:** Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This paper was partially supported by the National Key Research and Development Program of China (Nos. 2018YFB1005100, 2018YFB1005104), and a key program of fundamental research from Shenzhen Science and Technology Innovation Commission, China (No. JCYJ20200109113403826).

References

[1] P. Liu, X. Qiu, X. Huang, Recurrent neural network for text classification with multi-task learning, 2016, arXiv preprint arXiv:1605.05101.
[2] L. Zhang, S. Wang, B. Liu, Deep learning for sentiment analysis: A survey, Wiley Interdiscip. Rev.: Data Min. Knowl. Dis. 8 (4) (2018) e1253.
[3] Y. Dai, J. Liu, X. Ren, Z. Xu, Adversarial training based multi-source unsupervised domain adaptation for sentiment analysis, in: AAAI, 2020, pp. 7618–7625.
[4] M. De Gemmis, P. Lops, C. Musto, F. Narducci, G. Semeraro, Semantics-aware content-based recommender systems, in: Recommender Systems Handbook, Springer, 2015, pp. 119–159.
[5] H. Bai, Z. Chen, M.R. Lyu, I. King, Z. Xu, Neural relational topic models for scientific article analysis, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22–26, 2018, ACM, 2018, pp. 27–36.
[6] X. Su, S. Xue, F. Liu, J. Wu, J. Yang, C. Zhou, W. Hu, C. Paris, S. Nepal, D. Jin, et al., A comprehensive survey on community detection with deep learning, 2021, arXiv preprint arXiv:2105.12584.
[7] X. Ma, J. Wu, S. Xue, J. Yang, Q.Z. Sheng, H. Xiong, A comprehensive survey on graph anomaly detection with deep learning, 2021, arXiv preprint arXiv:2106.07178.
[8] S. Lai, L. Xu, K. Liu, J. Zhao, Recurrent convolutional neural networks for text classification, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.
[9] M. Jaderberg, K. Simonyan, A. Vedaldi, A. Zisserman, Reading text in the wild with convolutional neural networks, Int. J. Comput. Vis. 116 (1) (2016) 1–20.
[10] P. Wang, B. Xu, J. Xu, G. Tian, C.-L. Liu, H. Hao, Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification, Neurocomputing 174 (2016) 806–814.
[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.
[12] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, D. Brown, Text classification algorithms: A survey, Information 10 (4) (2019) 150.
[13] C. Yin, Y. Zhu, J. Fei, X. He, A deep learning approach for intrusion detection using recurrent neural networks, IEEE Access 5 (2017) 21954–21961.
[14] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, S. Valaee, Recent advances in recurrent neural networks, 2017, arXiv preprint arXiv:1801.01078.
[15] A. Khan, A. Sohail, U. Zahoora, A.S. Qureshi, A survey of the recent architectures of deep convolutional neural networks, Artif. Intell. Rev. 53 (8) (2020) 5455–5516.
[16] S. Khan, H. Rahmani, S.A.A. Shah, M. Bennamoun, A guide to convolutional neural networks for computer vision, Synth. Lect. Comput. Vis. 8 (1) (2018) 1–207.
[17] Y. Zhang, B. Wallace, A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification, 2015, arXiv preprint arXiv:1510.03820.
[18] G. Chen, D. Ye, Z. Xing, J. Chen, E. Cambria, Ensemble application of convolutional and recurrent neural networks for multi-label text categorization, in: 2017 International Joint Conference on Neural Networks, IJCNN, IEEE, 2017, pp. 2377–2383.
[19] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks, 2019, arXiv preprint arXiv:1901.00596.
[20] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint arXiv:1609.02907.
[21] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 1263–1272.
[22] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems, 2017, pp. 1024–1034.
[23] L. Yao, C. Mao, Y. Luo, Graph convolutional networks for text classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 7370–7377.
[24] X. Liu, X. You, X. Zhang, J. Wu, P. Lv, Tensor graph convolutional networks for text classification, 2020, arXiv preprint arXiv:2001.05313.
[25] G. Ciano, A. Rossi, M. Bianchini, F. Scarselli, On inductive-transductive learning with graph neural networks, IEEE Trans. Pattern Anal. Mach. Intell. (2021).
[26] L. Huang, D. Ma, S. Li, X. Zhang, H. Wang, Text level graph neural network for text classification, in: K. Inui, J. Jiang, V. Ng, X. Wan (Eds.), Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3–7, 2019, Association for Computational Linguistics, 2019, pp. 3442–3448.

[27] Y. Matsuo, M. Ishizuka, Keyword extraction from a single document using word co-occurrence statistical information, Int. J. Artif. Intell. Tools 13 (01) (2004) 157–169.
[28] J.A. Bullinaria, J.P. Levy, Extracting semantic representations from word co-occurrence statistics: A computational study, Behav. Res. Methods 39 (3) (2007) 510–526.
[29] A. Dhar, H. Mukherjee, N.S. Dash, K. Roy, Cess-a system to categorize bangla web text documents, ACM Trans. Asian Low-Resour. Lang. Inf. Process. (TALLIP) 19 (5) (2020) 1–18.
[30] F. Figueiredo, L. Rocha, T. Couto, T. Salles, M.A. Gonçalves, W. Meira Jr., Word co-occurrence features for text classification, Inf. Syst. 36 (5) (2011) 843–858.
[31] J. Pennington, R. Socher, C. Manning, GloVe: Global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP, Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1532–1543.
[32] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013, arXiv preprint arXiv:1301.3781.
[33] O. Levy, Y. Goldberg, Neural word embedding as implicit matrix factorization, in: Advances in Neural Information Processing Systems, 2014, pp. 2177–2185.
[34] M. Lee, D. Bindel, D. Mimno, Robust spectral inference for joint stochastic matrix factorization, in: Advances in Neural Information Processing Systems, 2015, pp. 2710–2718.
[35] S. Arora, R. Ge, Y. Halpern, D. Mimno, A. Moitra, D. Sontag, Y. Wu, M. Zhu, A practical algorithm for topic modeling with provable guarantees, in: International Conference on Machine Learning, 2013, pp. 280–288.
[36] K.W. Church, P. Hanks, Word association norms, mutual information, and lexicography, Comput. Linguist. 16 (1) (1990) 22–29.
[37] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems, 2013, pp. 3111–3119.
[38] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv preprint arXiv:1810.04805.
[39] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, X. Huang, Pre-trained models for natural language processing: A survey, 2020, arXiv preprint arXiv:2003.08271.
[40] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[41] Z. Liu, Y. Lin, M. Sun, Representation learning and NLP, in: Representation Learning for Natural Language Processing, Springer, 2020, pp. 1–11.
[42] A. Dhar, H. Mukherjee, N.S. Dash, K. Roy, Text categorization: past and present, Artif. Intell. Rev. 54 (4) (2021) 3007–3054.
[43] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, J. Gao, Deep learning–based text classification: A comprehensive review, ACM Comput. Surv. 54 (3) (2021) 1–40.
[44] Y. Dai, J. Liu, J. Zhang, H. Fu, Z. Xu, Unsupervised sentiment analysis by transferring multi-source knowledge, 2021, arXiv preprint arXiv:2105.11902.
[45] Y. Gou, Y. Lei, L. Liu, P. Zhang, X. Peng, A dynamic parameter enhanced network for distant supervised relation extraction, Knowl.-Based Syst. 197 (2020) 105912.
[46] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[47] G. Liu, J. Guo, Bidirectional LSTM with attention mechanism and convolutional layer for text classification, Neurocomputing 337 (2019) 325–338.
[48] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014, arXiv preprint arXiv:1412.3555.
[49] A. Elnagar, R. Al-Debsi, O. Einea, Arabic text classification using deep learning models, Inf. Process. Manage. 57 (1) (2020) 102121.
[50] W. Ali, N. Ali, Y. Dai, J. Kumar, S. Tumrani, Z. Xu, Creating and evaluating resources for sentiment analysis in the low-resource language: Sindhi, in: Proceedings of the Eleventh Workshop on Computational Approaches To Subjectivity, Sentiment and Social Media Analysis, 2021, pp. 188–194.
[51] K. Shimura, J. Li, F. Fukumoto, HFT-CNN: Learning hierarchical category structure for multi-label short text categorization, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 811–816.
[52] S. Sabour, N. Frosst, G.E. Hinton, Dynamic routing between capsules, 2017, arXiv preprint arXiv:1710.09829.
[53] J. Kim, S. Jang, E. Park, S. Choi, Text classification using capsules, Neurocomputing 376 (2020) 214–221.
[54] W. Zhao, J. Ye, M. Yang, Z. Lei, S. Zhang, Z. Zhao, Investigating capsule networks with dynamic routing for text classification, 2018, arXiv preprint arXiv:1804.00538.
[55] I. Chaturvedi, Y.-S. Ong, I.W. Tsang, R.E. Welsch, E. Cambria, Learning word dependencies in text by means of a deep recurrent belief network, Knowl.-Based Syst. 108 (2016) 144–154.
[56] Y. Ma, H. Peng, T. Khan, E. Cambria, A. Hussain, Sentic LSTM: a hybrid network for targeted aspect-based sentiment analysis, Cogn. Comput. 10 (4) (2018) 639–650.
[57] N. Zainuddin, A. Selamat, R. Ibrahim, Hybrid sentiment classification on twitter aspect-based sentiment analysis, Appl. Intell. 48 (5) (2018) 1218–1232.
[58] Y. Li, Q. Pan, S. Wang, T. Yang, E. Cambria, A generative model for category text generation, Inform. Sci. 450 (2018) 301–315.
[59] Z. Kang, H. Pan, S.C. Hoi, Z. Xu, Robust graph learning from noisy data, IEEE Trans. Cybern. 50 (5) (2019) 1833–1843.
[60] Z. Kang, Z. Lin, X. Zhu, W. Xu, Structured graph learning for scalable subspace clustering: From single view to multiview, IEEE Trans. Cybern. (2021).
[61] H. Guo, X. Li, L. Zhang, J. Liu, W. Chen, Label-aware text representation for multi-label text classification, in: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2021, pp. 7728–7732.
[62] W. Li, S. Li, S. Ma, Y. He, D. Chen, X. Sun, Recursive graphical neural networks for text classification, 2019, arXiv preprint arXiv:1909.08166.
[63] R. Ragesh, S. Sellamanickam, A. Iyer, R. Bairi, V. Lingam, HeteGCN: Heterogeneous graph convolutional networks for text classification, 2020, arXiv preprint arXiv:2008.12842.
[64] J. Tang, M. Qu, Q. Mei, Pte: Predictive text embedding through large-scale heterogeneous text networks, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 1165–1174.
[65] I. Dagan, L. Lee, F.C. Pereira, Similarity-based models of word cooccurrence probabilities, Mach. Learn. 34 (1–3) (1999) 43–69.
[66] I. Dagan, F. Pereira, L. Lee, Similarity-based estimation of word cooccurrence probabilities, in: Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, 1994, pp. 272–278.
[67] P.D. Turney, Mining the web for synonyms: PMI-IR versus LSA on TOEFL, in: European Conference on Machine Learning, Springer, 2001, pp. 491–502.
[68] P.D. Turney, P. Pantel, From frequency to meaning: Vector space models of semantics, J. Artificial Intelligence Res. 37 (2010) 141–188.
[69] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, 2019, arXiv preprint arXiv:1908.10084.
[70] V. Yadav, S. Bethard, M. Surdeanu, Unsupervised alignment-based iterative evidence retrieval for multi-hop question answering, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 4514–4525.
[71] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, W.-t. Yih, Dense passage retrieval for open-domain question answering, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP, Association for Computational Linguistics, Online, 2020, pp. 6769–6781.
[72] I. Beltagy, M.E. Peters, A. Cohan, Longformer: The long-document transformer, 2020, arXiv:2004.05150.
[73] M. Zaheer, G. Guruganesh, K.A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, et al., Big bird: Transformers for longer sequences, Adv. Neural Inf. Process. Syst. 33 (2020).
[74] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, I. Titov, Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned, 2019, arXiv preprint arXiv:1905.09418.
[75] H. Schütze, C.D. Manning, P. Raghavan, Introduction to Information Retrieval, vol. 39, Cambridge University Press Cambridge, 2008.
[76] Y. Kim, Convolutional neural networks for sentence classification, 2014, arXiv preprint arXiv:1408.5882.
[77] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: International Conference on Machine Learning, 2014, pp. 1188–1196.
[78] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of tricks for efficient text classification, 2016, arXiv preprint arXiv:1607.01759.
[79] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, L. Carin, Joint embedding of words and labels for text classification, 2018, arXiv preprint arXiv:1805.04174.
[80] D. Shen, G. Wang, W. Wang, M.R. Min, Q. Su, Y. Zhang, C. Li, R. Henao, L. Carin, Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms, 2018, arXiv preprint arXiv:1805.09843.
[81] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in Neural Information Processing Systems, 2016, pp. 3844–3852.
[82] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, 2013, arXiv preprint arXiv:1312.6203.
[83] M. Henaff, J. Bruna, Y. LeCun, Deep convolutional networks on graph-structured data, 2015, arXiv preprint arXiv:1506.05163.
[84] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, 2017, arXiv preprint arXiv:1711.05101.