



# Graph neural networks for text classification: a survey

Kunze Wang<sup>1</sup> · Yihao Ding<sup>1,2</sup> · Soyeon Caren Han<sup>1,2</sup>

Published online: 1 July 2024

© The Author(s) 2024

## Abstract

Text Classification is the most essential and fundamental problem in Natural Language Processing. While numerous recent text classification models applied the sequential deep learning technique, graph neural network-based models can directly deal with complex structured text data and exploit global information. Many real text classification applications can be naturally cast into a graph, which captures words, documents, and corpus global features. In this survey, we bring the coverage of methods up to 2023, including corpus-level and document-level graph neural networks. We discuss each of these methods in detail, dealing with the graph construction mechanisms and the graph-based learning process. As well as the technological survey, we look at issues behind and future directions addressed in text classification using graph neural networks. We also cover datasets, evaluation metrics, and experiment design and present a summary of published performance on the publicly available benchmarks. Note that we present a comprehensive comparison between different techniques and identify the pros and cons of various evaluation metrics in this survey.

**Keywords** Graph neural networks · Text classification · Representation learning

## 1 Introduction

Text classification aims to classify a given document into certain pre-defined classes, and is considered to be a fundamental task in natural language processing (NLP). It includes a large number of downstream tasks, such as topic classification (Zhang et al. 2015), and sentiment analysis (Tai et al. 2015). Traditional text classification methods build representation on the text using N-gram (Cavnar et al. 1994) or Term Frequency-Inverse Document Frequency (TF-IDF) (Hakim et al. 2014) and apply traditional machine learning models, such as SVM (Joachims 2005), to classify the documents. With the development of neural networks, more deep learning models have been applied to text classification, including convolutional neural networks (CNN) (Kim 2014), recurrent neural networks (RNN)

---

✉ Soyeon Caren Han  
caren.han@unimelb.edu.au

<sup>1</sup> School of Computer Science, The University of Sydney, City Rd, Sydney, NSW 2006, Australia

<sup>2</sup> Faculty of Engineering and IT, The University of Melbourne, Southbank 234, Melbourne, VIC 3052, Australia

(Tang et al. 2015) and attention-based (Vaswani et al. 2017) models and large language models (Devlin et al. 2018).

However, these methods are either unable to handle the complex relationships between words and documents (Yao et al. 2019), and can not efficiently explore the contextual-aware word relations (Zhang et al. 2020). Graph neural networks (GNN) are introduced to resolve such obstacles. GNN is used with graph-structure datasets, so a graph needs to be built for text classification. There are two main approaches to constructing graphs: corpus-level and document-level graphs. The datasets are either built into single or multiple corpus-level graphs representing the whole corpus or numerous document-level graphs and each of them represents a document. The corpus-level graph can capture the global structural information of the entire corpus, while the document-level graph can explicitly capture the word-to-word relationships within a document. Both ways of applying graph neural networks to text classification achieve good performance.

This paper mainly focuses on GNN-based text classification techniques, datasets, and their performance. The graph construction approaches for both corpus-level and document-level graphs are addressed in detail. Papers on the following aspects will be reviewed:

- GNNs-based text classification approaches. Papers that design GNN-based frameworks to enhance the feature representation or directly apply GNNs to conduct sequence text classification tasks will be summarized, described and discussed. GNNs applied for token-level classification (Natural Language Understanding) tasks, including NER, slot filling, etc, will not be discussed in this work.
- Text classification benchmark datasets and their performance applied by GNN-based models. The text classification datasets with commonly used metrics used by GNNs-based text classification models will be summarized and categorized based on task types and the model performance on these datasets.

## 1.1 Related surveys and our contribution

Before 2019, the text classification survey papers (Xing et al. 2010; Khan et al. 2010; Harish et al. 2010; Aggarwal and Zhai 2012; Vijayan et al. 2017) have focused on covering traditional machine learning-based text classification models. Recently, with the rapid development of deep learning techniques, (Minaee et al. 2021; Zulqarnain et al. 2020; Zhou 2020; Li et al. 2022) review the various deep learning-based approaches. In addition, some papers not only review the SoTA model architectures but summarize the overall workflow (Jindal et al. 2015; Kadhim 2019; Mirończuk and Protasiewicz 2018; Kowsari et al. 2019; Bhavani and Kumar 2021) or specific techniques for text classification including word embedding (Selva Birunda and Kanniga 2021), feature selection (Deng et al. 2019; Shah and Patel 2016; Pintas et al. 2021), term weighting (Patra and Singh 2013; Alsaeedi 2020) and etc. Meanwhile, some growing potential text classification architectures are surveyed, such as CNNs (Yang et al. 2016), attention mechanisms (Mariyam et al. 2021). Since the powerful ability to represent non-Euclidean relation, GNNs have been used in multiple practical fields and reviewed e.g. financial application (Wang et al. 2021), traffic prediction (Liu and Tan 2021), bio-informatics (Zhang et al. 2021), power system (Liao et al. 2021), recommendation system (Gao et al. 2022; Liang et al. 2021; Yang et al. 2021). Moreover, (Bronstein et al. 2017; Battaglia et al. 2018; Zhang et al. 2019; Zhou et al. 2020; Wu et al. 2020) comprehensively review the general algorithms and applications of GNNs, as well as certain surveys mainly focus on specific perspectives

including graph construction (Skarding et al. 2021; Thomas et al. 2022), graph representation (Hamilton et al. 2017), training (Xie et al. 2022), pooling (Liu et al. 2022) and more. However, only (Minaee et al. 2021; Li et al. 2022) briefly introduce certain SoTA GNN-based text classification models. A recent short review paper (Malekzadeh et al. 2021) reviews the concept of GNNs and four SoTA GNN-based text classification models. However, our study focuses explicitly on applying GNN techniques in text classification tasks. We delve into various GNN-related methodologies, including graph construction, node and edge representation, and training approaches commonly employed in text classification. Unlike (Malekzadeh et al. 2021) that typically review a limited number of models, our survey encompasses around 30 models categorised into document-level and corpus-level classifications, enabling a comprehensive analysis for comparing and contrasting these approaches. Additionally, our study goes beyond merely examining models by providing an in-depth analysis of metrics and datasets commonly used in GNN-based text classification tasks, aiming to offer valuable insights for future research in similar areas.

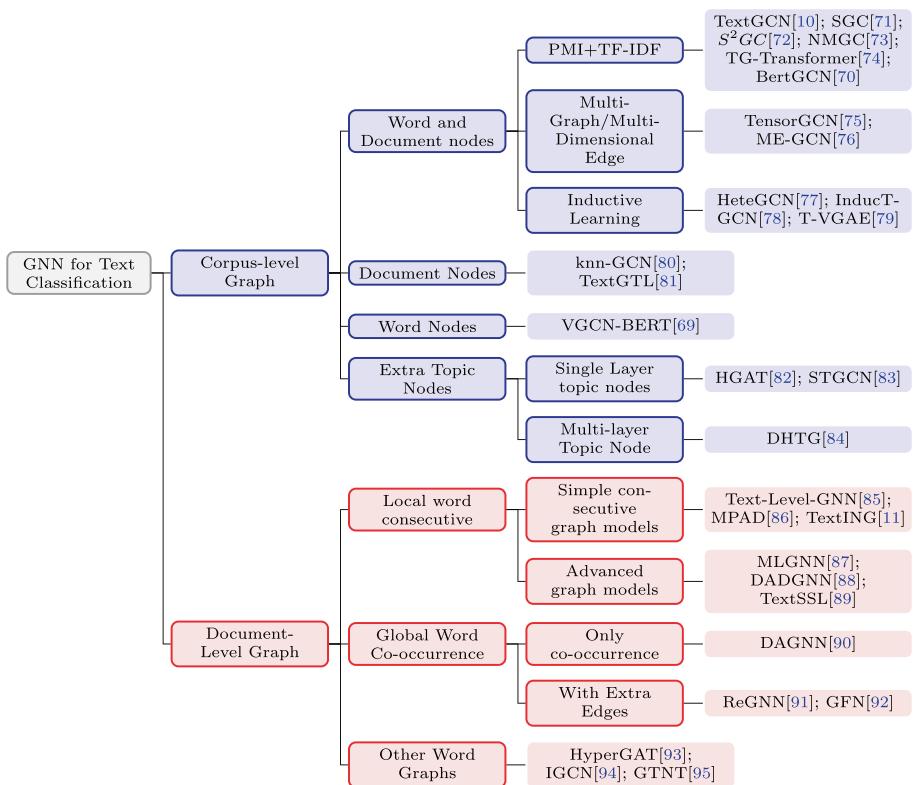
The contribution of this survey includes:

- This is the first survey focused only on graph neural networks for text classification with a comprehensive description and critical discussion on more than twenty GNN text classification models.
- We categorize the existing GNN text classification models into two main categories with multiple sub-categories, and the tree structure of all the models shows in Fig. 1.
- We compare these models in terms of graph construction, node embedding initialization, and graph learning methods. And we also compare the performance of these models on the benchmark datasets and discuss the key findings.
- We discuss the existing challenges and some potential future work for GNN text classification models.

## 1.2 Text classification tasks

Text classification involves assigning a pre-defined label to a given text sequence. The process typically involves encoding pre-processed raw text into numerical representations and using classifiers to predict the corresponding categories. Typical sub-tasks include sentiment analysis, topic labelling, news categorization, and hate speech detection. Specific frameworks can be extended to advanced applications such as information retrieval, summarising, question answering, and natural language inference. This paper focuses specifically on GNN-based models used for typical text classification.

- **Sentiment analysis** is a task that aims to identify the emotional states and subjective opinions expressed in the input text, such as reviews, micro-blogs, etc. This can be achieved through binary or multi-class classification. Effective sentiment analysis can aid in making informed business decisions based on user feedback.
- **Topic classification** is a supervised deep learning task to automatically understand the text content and classify it into multiple domain-specific categories, typically more than two. The data sources may be gathered from different domains, including Wikipedia pages, newspapers, scientific papers, etc.
- **Junk information detection** involves detecting inappropriate social media content. Social media providers commonly use approaches like hate speech, abusive language, advertising or spam detection to remove such content efficiently.



**Fig. 1** Categorizing the graph neural network text classification models

### 1.3 Text classification development

Many traditional machine learning methods and deep learning models are selected as baselines for comparison with the GNN-based text classifiers. We mainly summarized those baselines into three types:

**Traditional machine learning:** In earlier years, traditional methods such as Support Vector Machines (SVM) (Zhang et al. 2011) and Logistic Regression (Genkin et al. 2007) utilized sparse representations like Bag of Words (BoW) and TF-IDF. However, recent advancements (Lilleberg et al. 2015; Yin and Jin 2015; Ren et al. 2016) have focused on dense representations, such as Word2vec, GloVe, and Fasttext, to mitigate the limitations of sparse representations. These dense representations are also used as inputs for sophisticated methods, such as Deep Averaging Networks (DAN) (Iyyer et al. 2015) and Paragraph Vector (Doc2Vec) (Le and Mikolov 2014), to achieve new state-of-the-art results.

**Sequential models:** RNNs and CNNs have been utilized to capture local-level semantic and syntactic information of consecutive words from input text bodies. The upgraded models, such as LSTM (Graves 2012) and GRU (Cho et al. 2014), have been proposed to address the vanishing or exploding gradient problems caused by vanilla RNN. CNN-based structures have been applied to capture N-gram features by using one or more convolution and pooling layers, such as Dynamic CNN (Kalchbrenner et al. 2014) and TextCNN

(Kim 2014). However, these models can only capture local dependencies of consecutive words. To capture longer-term or non-Euclidean relations, improved RNN structures, such as Tree-LSTM (Tai et al. 2015) and MT-LSTM (Liu et al. 2015), and global semantic information, like TopicRNN (Dieng et al. 2016), have been proposed. Additionally, graph (Peng et al. 2018) and tree structure (Mou et al. 2015) enhanced CNNs have been proposed to learn more about global and long-term dependencies.

**Attentions and transformers:** attention mechanisms (Bahdanau et al. 2014) have been widely adopted to capture long-range dependencies, such as hierarchical attention networks (Abreu et al. 2019) and attention-based hybrid models (Yang et al. 2016). More attention-based text classification frameworks are summarized by (Minaee et al. 2021). Self-attention-based transformer architectures have achieved state-of-the-art performance on many text classification benchmarks via unsupervised pre-training tasks to generate strong contextual word representations (Devlin et al. 2018; Liu et al. 2019). However, although those large-scale models implicitly store general domain knowledge and are widely used to generate more representative textual representations, they only focus on learning the relation between input text bodies and ignore the global and corpus-level information (Lu et al. 2020; Lin et al. 2021).

## 1.4 Outline

The outline of this survey is as follows:

- Section 1 presents the research questions and provides an overview of applying Graph Neural Networks to text classification tasks, along with the scope and organization of this survey.
- Section 2 provides background information on text classification and graph neural networks and introduces the key concepts of applying GNNs to text classification from a designer's perspective.
- Section 3 and Sect. 4 discuss previous work on Corpus-level Graph Neural Networks and Document-level Graph Neural Networks, respectively, and provide a comparative analysis of the strengths and weaknesses of these two approaches.
- Section 5 introduces the commonly used datasets and evaluation metrics in GNN for text classification.
- Section 6 reports the performance of various GNN models on a range of benchmark datasets for text classification and discusses the key findings.
- The challenges for the existing methods and some potential future works are discussed in Sect. 7.
- In Sect. 8, we present the conclusions of our survey on GNN for text classification and discuss potential directions for future work.

## 2 Backgrounds of GNN

### 2.1 Definition of graph

A graph in this paper is represented as  $G = (V, E)$ , where  $V$  and  $E$  represent a set of nodes (vertices) and edges of  $G$ , respectively. A single node in the node set is represented  $v_i \in V$ , as well as  $e_{ij} = (v_i, v_j) \in E$  donates an edge between node  $v_i$  and  $v_j$ . The adjacent matrix of

graph  $G$  is represented as  $A$ , where  $A \in \mathbb{R}^{n \times n}$  and  $n$  is the number of nodes in graph  $G$ . If  $e_{ij} \in E$ ,  $A_{ij} = 1$ , otherwise  $A_{ij} = 0$ . In addition, we use  $X$  and  $E$  to represent the nodes and edges representations in graph  $G$ , where  $X \in \mathbb{R}^{n \times m}$  and  $E \in \mathbb{R}^{n \times c}$ .  $x_i \in \mathbb{R}^m$  represents the  $m$ -dimensional vector of node  $v_i$  and  $e_{ij} \in \mathbb{R}^c$  represents the  $c$ -dimensional vector of edge  $e_{ij}$  (most of the recent studies set  $c = 1$  to represent a weighting scalar).  $A$  donates the edge feature weighted adjacent matrix.

## 2.2 Traditional graph-based algorithms

Before GNNs were broadly used for representing irregular relations, traditional graph-based algorithms have been applied to model the non-Euclidean structures in text classification e.g. Random Walk (Szummer and Jaakkola 2001; Zhou and Li 2005), Graph Matching (Schenker et al. 2004; Silva et al. 2014), Graph Clustering (Matsuo et al. 2006) which has been well summarized in (Wu et al. 2021). There are three common limitations of those traditional graph-based algorithms. Firstly, most of those algorithms mainly focus on capturing graph-level structure information without considering the significance of node and edge features. For example, Random Walk-based approaches (Zhou and Li 2005; Szummer and Jaakkola 2001) mainly focus on using distance or angle between node vectors to calculate transition probability while ignoring the information represented by node vectors. Secondly, since the traditional graph-based algorithms are only suitable for specific tasks, there is no unified learning framework for addressing various practical tasks. For example, Kaur and Kumar (2018) proposes a graph clustering method that requires a domain knowledge-based ontology graph. Lastly, the traditional graph-based methods are comparative time inefficient like the Graph Edit Distance-based graph matching methods have exponential time complexity (Silva et al. 2014).

## 2.3 Foundations of GNN

To tackle the limitation of traditional graph-based algorithms and better represent non-Euclidean relations in practical applications, Graph Neural Networks are proposed by Scarselli et al. (2008). GNNs have a unified graph-based framework and simultaneously model the graph structure, node, and edge representations. This section will provide the general mathematical definitions of Graph Neural Networks. The general forward process of GNN can be summarised as follows:

$$\mathbf{H}^{(l)} = \mathcal{F}(\mathbf{A}, \mathbf{H}^{(l-1)}) \quad (1)$$

where  $\mathbf{A} \in \mathbb{R}^{n \times n}$  represents the weighted adjacent matrix and  $\mathbf{H}^{(l)} \in \mathbb{R}^{n \times d}$  is the updated node representations at the  $l$ -th GNN layers by feeding  $l - 1$ -th layer node features  $\mathbf{H}^{(l-1)} \in \mathbb{R}^{n \times k}$  ( $k$  is the dimensions of previous layers node representations) into pre-defined graph filters  $\mathcal{F}$ .

The most commonly used graph filtering method is defined as follows:

$$\mathbf{H}^{(l)} = \phi(\tilde{\mathbf{A}}\mathbf{H}^{(l-1)}\mathbf{W}) \quad (2)$$

where  $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$  is the normalized symmetric adjacency matrix.  $\mathbf{A} \in \mathbb{R}^{n \times n}$  is the adjacent matrix of graph  $G$  and  $\mathbf{D}$  is the degree matrix of  $\mathbf{A}$ , where  $D_{ii} = \sum_j A_{ij}$ .  $\mathbf{W} \in \mathbb{R}^{k \times d}$  is the weight matrix and  $\phi$  is the activation function. If we design a two layers of GNNs based

**Table 1** Commonly used notations in Graph Neural Networks

Notations	Descriptions
$G$	A graph
$V$	The set of nodes in a graph
$E$	The set of edges in a graph
$e_{ij}$	An edge between node $i$ and node $j$
$N_i$	The neighbors of a node $i$
$A$	The graph adjacency matrix
$\tilde{A}$	The normalized matrix $A$
$\tilde{A}^k, k \in \mathbb{Z}$	The $k^{th}$ power of $\tilde{A}$
$[A  B]$	The concatenation of $A$ and $B$
$D$	The degree matrix of $A$ . $D_{ii} = \sum_{j=1}^n A_{ij}$
$W^{(l)}$	The weight matrix of layer $l$
$H \in \mathbb{R}^{n \times d}$	The feature matrix of a graph
$H^{(l)} \in \mathbb{R}^{n \times d}$	The feature matrix of a graph at layer $l$
$h_i \in \mathbb{R}^n$	The feature vector of the node $i$
$h_i^{(l)} \in \mathbb{R}^n$	The feature vector of the node $i$ at layer $l$
$Z \in \mathbb{R}^{n \times d}$	The output feature matrix of a graph
$z_i \in \mathbb{R}^n$	The output feature vector of the node $i$

on the above filter could get a vanilla Graph Convolutional Network (GCN) (Welling and Kipf 2016) framework for text classification:

$$\mathbf{Y} = \text{softmax}(\tilde{A}(\text{ReLU}(\tilde{A}H\mathbf{W}^{(0)}))\mathbf{W}^{(1)}) \quad (3)$$

where  $\mathbf{W}^0$  and  $\mathbf{W}^1$  represent different weight matrix for different GCN layers and  $H$  is the input node features.  $\text{ReLU}$  function is used for non-linearization and  $\text{softmax}$  is used to generated predicted categories  $\mathbf{Y}$ . The notation of GNN can be found in Table 1.

## 2.4 GNN for text classification

This paper mainly discusses how GNNs are applied in Text Classification tasks. Before we present the specific applications in this area, we first introduce the key concepts of applying GNNs to text classification from a designer's view. We suppose for addressing a text classification task, we need to design a graph  $G = (V, E)$ . The general procedures include *Graph Construction*, *Initial Node Representation*, *Edge Representations*, and *Training Setup*.

### 2.4.1 Graph construction

Some applications have explicit graph structures, including constituency or dependency graphs (Tang et al. 2020), knowledge graphs (Ostendorff et al. 2019; Marin et al. 2014), social networks (Dai et al. 2022) without constructing graph structure and defining

corresponding nodes and edges. However, for text classification, the most common graph structures are implicit, which means we need to define a new graph structure for a specific task, such as designing a word-word or word-document co-occurrence graph. In addition, for text classification tasks, the graph structure can be generally classified into two types:

- ***Corpus-level/document-level***: Corpus-level graphs intend to construct the graph to represent the whole corpus, such as Yao et al. (2019); Liu et al. (2020); Lin et al. (2021); Wu et al. (2019), while the document-level graphs focus on representing the non-Euclidean relations existing in a single text body like Chen et al. (2020); Nikolentzos et al. (2020); Zhang et al. (2020). Supposing a specific corpus  $\mathcal{C}$  contains a set of documents (text bodies)  $\mathcal{C} = \{D_1, D_2, \dots, D_j\}$  and each  $D_i$  contains a set of tokens  $D_i = \{t_{i_1}, t_{i_2}, \dots, t_{i_l}\}$ . The vocabulary of  $\mathcal{C}$  can be represented as  $\mathcal{D} = \{t_1, t_2, \dots, t_l\}$ , where  $l$  is the length of  $\mathcal{D}$ . For the most commonly adopted corpus-level graph  $G_{corpus} = (V_{corpus}, E_{corpus})$ , a node  $v_i$  in  $V_{corpus}$  follows  $v_i \in \mathcal{C} \cup \mathcal{D}$  and the edge  $e_{ij} \in E_{corpus}$  is one kind of relations between  $v_i$  and  $v_j$ . Regarding the document level graph  $G_{doc_i} = (V_{doc_i}, E_{doc_i})$ , a node  $v_{i_j}$  in  $V_{doc_i}$  follows  $v_{i_j} \in D_i$ .

After designing the graph scale for the specific tasks, specifying the graph types is also important to determine the nodes and their relations. For text classification tasks, the commonly used graph construction ways can be summarized into:

- ***Homogeneous/heterogeneous graphs***: homogeneous graphs have the same node and edge type while heterogeneous graphs have various node and edge types. For a graph  $G = (V, E)$ , we use  $\mathcal{N}^v$  and  $\mathcal{N}^e$  to represent the number of types of  $V$  and  $E$ . If  $\mathcal{N}^v = \mathcal{N}^e = 1$ ,  $G$  is a homogeneous graph. If  $\mathcal{N}^v > 1$  or  $\mathcal{N}^e > 1$ ,  $G$  is a heterogeneous graph.
- ***Static/dynamic graphs***: Static graphs aim to use the constructed graph structure by various external or internal information to leverage to enhance the initial node representation such as dependency or constituency graph (Tang et al. 2020), co-occurrence between word nodes (Zhang et al. 2020), TF-IDF between word and document nodes (Yao et al. 2019; Wu et al. 2019; Lei et al. 2021) and so on. However, compared with the static graph, the dynamic graph initial representations or graph topology are changing during training without certain domain knowledge and human efforts. The feature representations or graph structure can jointly learn with downstream tasks to be optimised together. For example, Wang et al. (2020) proposed a novel topic-aware GNN text classification model with dynamically updated edges between topic nodes with others (e.g. document, word). Piao et al. (2021) also designed a dynamic edge-based graph to update the contextual dependencies between nodes. Additionally, Chen et al. (2020) propose a dynamic GNN model to jointly update the edge and node representation simultaneously. We provide more details about the above-mentioned models in Sect. 3 and Sect. 4.

Another widely used pair of graph categories are ***directed*** or ***undirected*** graphs based on whether the directions of edges are bi-directional or not. For text classification, most of the GNN designs follow the unidirectional way. In addition, those graph-type pairs are not parallel, which means they can be combined.

### 2.4.2 Initial node representation

Based on the pre-defined graph structure and specified graph type, selecting the appropriate initial node representations is the key procedure to ensure the proposed graph structure can effectively learn node. According to the node entity type, the existing node representation approaches for text classification can be generally summarized into:

- **Word-level representation:** non-context word embedding methods such as GloVe (Pennington et al. 2014), Word2vec (Mikolov et al. 2013), FastText (Bojanowski et al. 2017) are widely adopted by many GNN-based text classification framework to represent the node features numerically. However, those embedding methods are restricted to capturing only syntactic similarity and fail to represent the complex semantic relationships between words. They cannot capture the meaning of out-of-vocabulary (OOV) words, and their representations are fixed. Therefore, there are some recent studies selecting ELMo (Peters et al. 2018), BERT (Devlin et al. 2018), GPT (Radford et al. 2018) to get contextual word-level node representation. Notably, even if the one-hot encoding is the simplest word representation method, many GNN-based text classifiers use one-hot encoding and achieve state-of-the-art performance. Few frameworks use randomly initialised vectors to represent the word-level node features.
- **Document-level representation:** similar to other NLP applications, document-level representations are normally acquired by aggregating the word-level representation via some deep learning frameworks. For example, some researchers select by extracting the last-hidden state of LSTM or using the [CLS] token from BERT to represent the input text body numerically. Furthermore, it is also a commonly used document-level node representation way to use TF-IDF-based document vectors.

Most GNN-based text classification frameworks will compare the performance between different node representation methods to conduct quantitative analysis, as well as provide reasonable justifications for demonstrating the effectiveness of the selected initial node representation based on a defined graph structure.

### 2.4.3 Edge features

Well-defined edge features can effectively improve the graph representation learning efficiency and performance to exploit more explicit and implicit relations between nodes. Based on the predefined graph types, the edge feature types can be divided into **structural features** and **non-structural features**. The structural edge features are acquired from explicit relations between nodes, such as dependency or constituency relation between words, word-word adjacency relations, etc. That relationship between nodes is explicitly defined and widely employed in other NLP applications. However, more commonly used edge features are non-structural features which implicitly exist between the nodes and are specifically applied to specific graph-based frameworks. The typically non-structural edge features are firstly defined by Kim (2014) for GNNs-based text classification tasks, including:

- **PMI (point-wise mutual information)** measures the co-occurrence between two words in a sliding window  $W$  and is calculated as:

$$\text{PMI}(i,j) = \log \frac{p(i,j)}{p(i)p(j)}; \quad (4)$$

$$p(i,j) = \frac{\#W(i,j)}{\#W}; \quad (5)$$

$$p(i) = \frac{\#W(i)}{\#W}. \quad (6)$$

where  $\#W$  is the number of windows in total, and  $\#W(i)$ ,  $\#W(i,j)$  shows the number of windows containing word  $i$  and both word  $i$  and  $j$  respectively.

- **TF-IDF (term frequency-inverse document frequency)** is the broadly used weight of the edges between document-level nodes and word-level nodes.

Except for those two widely used implicit edge features, some specific edge weighting methods are proposed to meet the demands of particular graph structures for exploiting more information of input text bodies.

#### 2.4.4 Training setup

After specifying the graph structure and types, the graph representation learning tasks and training settings also need to be determined to decide how to optimise the designed GNNs. Generally, the graph representation learning tasks can be categorized into three levels, including **Node-level**, **Graph-level** and **Edge-level**. Node-level and graph-level tasks involve node or graph classification, clustering, regression, etc., while edge-level tasks include link prediction or edge classification for predicting the existence of the relation between two nodes or the corresponding edge categories.

Similar to other deep learning model training settings, GNNs also can be divided into **supervised**, **semi-supervised** and **unsupervised training settings**. Supervised training provides labelled training data, while unsupervised training utilises unlabeled data to train the GNNs. However, compared with supervised or unsupervised learning, semi-supervised learning methods are broadly used by GNNs designed for text classification applications, which could be classified into two types:

- **Inductive learning** adjusts the weights of proposed GNNs based on a labelled training set for learning the overall statistics to induce the general trained model for following processing. The unlabeled set can be fed into the trained GNNs to compute the expected outputs.
- **Transductive learning** intends to exploit labelled and unlabeled sets simultaneously for leveraging the relations between different samples to improve the overall performance.

### 2.4.5 Evolution of GNNs for text classification

TextGCN (Yao et al. 2019) and Text-Level-GNN (Huang et al. 2019) were the first to frame a text classification task as a node or graph classification task, achieved by constructing graphs based on textual data. Following these works, the field witnessed a proliferation of methodologies, exploring various avenues: (1) advancements in graph learning models, (2) improved graph construction strategies, (3) integration with State-of-the-Art text classification methods like Bert (Devlin et al. 2018).

In terms of the advancements in graph learning models, SGC (Wu et al. 2019) simplifies the Graph Convolutional Network (GCN) architecture, thereby conserving computational resources, S2GC (Zhu and Koniusz 2020) and NMGC (Lei et al. 2021) mitigate over-smoothing challenges by integrating skip-connection mechanisms, TensorGCn (Liu et al. 2020), TextGTL (Li et al. 2021) and ME-GCN (Wang et al. 2022) direct their efforts towards the acquisition of enriched edge information, T-VGAE (Xie et al. 2021) employs graph auto-encoder methodologies to enhance representation learning. HGAT (Linmei et al. 2019), ReGNN (Li et al. 2019), HyperGAT (Ding et al. 2020), MLGNN (Liao et al. 2021) and DADGNN (Liu et al. 2021) leverage attention mechanisms for model enhancement. A detailed exposition of these Graph Neural Network (GNN) models can be found in Sect. 3 and 4.

## 3 Corpus-level GNN for text classification

We define a corpus-level Graph Neural Network as “constructing a graph to represent the whole corpus”; thus, only one or several graphs will be built for the given corpus. We categorize Corpus-level GNN into four subcategories based on the types of nodes shown in the graph.

### 3.1 Document and word nodes as a graph

Most corpus-level graphs include word nodes and document nodes, and there are word-document edges and word-word edges. By applying  $K$  (normally  $K=2$  or 3) layer GNN, word nodes will serve as a bridge to propagate the information from one document node to another.

#### 3.1.1 PMI and TF-IDF as graph edges: TextGCN, SGC, S<sup>2</sup>GC, NMGC, TG-transformer, bertgCN

**TextGCN** (Yao et al. 2019) (Yao et al. 2019) builds a corpus-level graph with training document nodes, test document nodes and word nodes. Before constructing the graph, a common preprocessing method (Kim 2014) has been applied, and words shown fewer than five times or in NLTK (Bird et al. 2009) stopwords list have been removed. The edge value between the document node and the word node is TF-IDF, and that between the word nodes is PMI. The adjacency matrix of this graph is shown as follows.

$$A_{ij} = \begin{cases} \text{PMI}(i,j) & i, j \text{ are words, PMI}(i,j) > 0; \\ \text{TF-IDF}_{ij} & i \text{ is document, } j \text{ is word;} \\ 1 & i = j; \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

A two-layer GCN is applied to the graph, and the dimension of the second layer output equals the number of classes in the dataset. Formally, the forward propagation of TextGCN shows as follows:

$$\mathbf{Z} = \text{softmax}(\tilde{\mathbf{A}}(\text{ReLU}(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)}))\mathbf{W}^{(1)}) \quad (8)$$

where  $\tilde{\mathbf{A}}$  is the normalized adjacency of  $A$  and  $X$  is one-hot embedding.  $W_0$  and  $W_1$  are learnable parameters of the model. The representation on training documents is used to calculate the loss, and that on test documents is used for prediction. TextGCN is the first work that treats a text classification task as a node classification problem by constructing a corpus-level graph and has inspired many following works.

Based on TextGCN, several works follow the same graph construction method and node initialization but apply different graph propagation models.

**SGC** (Wu et al. 2019) To make GCN efficient, SGC (Simple Graph Convolution) removes the nonlinear activation function in GCN layers; therefore, the K-layer propagation of SGC is shown as follows:

$$\mathbf{Z} = \text{softmax}(\tilde{\mathbf{A}} \dots (\tilde{\mathbf{A}}(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)})\mathbf{W}^{(1)}) \dots \mathbf{W}^{(K)}) \quad (9)$$

which can be reparameterized into

$$\mathbf{Z} = \text{softmax}(\tilde{\mathbf{A}}^K \mathbf{X} \mathbf{W}) \quad (10)$$

and  $K$  is 2 when applied to text classification tasks. With a smaller number of parameters and only one feedforward layer, SGC saves computation time and resources while improving performance.

**S<sup>2</sup>GC** (Zhu and Koniusz 2020) To solve the over smoothing issues in GCN, (Zhu and Koniusz 2020) propose Simple Spectral Graph Convolution (S<sup>2</sup>GC), which includes self-loops using Markov Diffusion Kernel. The output of S<sup>2</sup>GC is calculated as:

$$\mathbf{Z} = \text{softmax}\left(\frac{1}{K} \sum_{k=0}^K \tilde{\mathbf{A}}^k \mathbf{X} \mathbf{W}\right) \quad (11)$$

And can be generalized into:

$$\mathbf{Z} = \text{softmax}\left(\frac{1}{K} \sum_{k=0}^K \left(1 - \alpha\right) \tilde{\mathbf{A}}^k \mathbf{X} + \alpha \mathbf{X}\right) \mathbf{W} \quad (12)$$

Similarly,  $K = 2$  on text classification tasks and  $\alpha$  denotes the trade-off between self-information of the node and consecutive neighbourhood information. S<sup>2</sup>GC can also be viewed as introducing skip connections into GCN.

**NMGC** (Lei et al. 2021) Other than using the sum of each GCN layer in S<sup>2</sup>GC, NMGC applies min pooling using the Multi-hop neighbour Information Fusion (MIF) operator to address over-smoothing problems. A MIF function is defined as:

$$\text{MIF}(K) = \min(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}, \tilde{\mathbf{A}}^2\mathbf{X}\mathbf{W}, \dots, \tilde{\mathbf{A}}^K\mathbf{X}\mathbf{W}) \quad (13)$$

NMGC-K firstly applies a MIF ( $K$ ) layer, then a GCN layer, and  $K$  is 2 or 3. For example, when  $K = 3$ , the output is:

$$\mathbf{Z} = \text{softmax}(\tilde{\mathbf{A}}(\text{ReLU} \min(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)}, \tilde{\mathbf{A}}^2\mathbf{X}\mathbf{W}^{(0)}, \tilde{\mathbf{A}}^3\mathbf{X}\mathbf{W}^{(0)}))\mathbf{W}^{(1)}) \quad (14)$$

NMGC can also be treated as a skip-connection in Graph Neural Networks, making the shallow layer of GNN directly contribute to the final representation.

**TG-Transformer** (Zhang and Zhang 2020) TextGCN treats the document nodes and word nodes as the same type of nodes during propagation, and to introduce heterogeneity into the TextGCN graph, TG-Transformer (Text Graph Transformer) adopts two sets of weights for document nodes and word nodes, respectively. To cope with a large corpus graph, subgraphs are sampled from the TextGCN graph using PageRank algorithm (Page et al. 1999). The input embedding of is the sum of three types of embedding: pretrained GloVe embedding, node type embedding, and Weisfeiler-Lehman structural encoding (Niepert et al. 2016). During propagation, self-attention (Vaswani et al. 2017) with graph residual (Zhang and Meng 2019) is applied.

**BertGCN** (Lin et al. 2021) To combine BERT (Devlin et al. 2018) and TextGCN, BertGCN enhances TextGCN by replacing the document node initialization with the BERT [CLS] output of each epoch and replacing the word input vector with zeros. BertGCN trains BERT and TextGCN jointly by interpolating the output of TextGCN and BERT:

$$\mathbf{Z} = \lambda \mathbf{Z}_{GCN} + (1 - \lambda) \mathbf{Z}_{BERT} \quad (15)$$

where  $\lambda$  is the trade-off factor. To optimize the memory during training, a memory bank is used to track the document input and a smaller learning rate is set to BERT module to remain the consistency of the memory bank. BertGCN shows that with the help of TextGCN, BERT can achieve better performance.

### 3.1.2 Multi-graphs/multi-dimensional edges: tensorGCN, ME-GCN

**TensorGCN** (Liu et al. 2020) Instead of constructing a single corpus-level graph, TensorGCN builds three independent graphs: A semantic-based graph, a Syntactic-based graph, and a Sequential-based graph to incorporate semantic, syntactic and sequential information, respectively and combines them into a tensor graph.

Three graphs share the same set of TF-IDF values for the word-document edge but different values for word-word edges. Semantic-based graph extracts the semantic features from a trained Long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) model and connects the words sharing high similarity. The syntactic-based graph uses Stanford CoreNLP parser (Manning et al. 2014) and constructs edges between words when they have a larger probability of having a dependency relation. For the Sequential-based graph, the PMI value is applied as TextGCN does.

The propagation includes intra-graph propagation and inter-graph propagation. The model first applies the GCN layer on three graphs separately as intra-graph propagation. Then, the same nodes on three graphs are treated as a virtual graph, and another GCN layer is applied as inter-graph propagation.

**ME-GCN** (Wang et al. 2022) To fully utilize the corpus information and analyze rich relational information of the graph, ME-GCN (Multi-dimensional Edge-Embedded GCN) builds a graph with multi-dimensional word-word, word-document and document-document edges. Word2vec and Doc2vec embedding is firstly trained on the given corpus

and the similarity of each dimension of trained embedding is used to construct the multi-dimensional edges. The trained embedding also serves as the input embedding of the graph nodes. During propagation, GCN is firstly applied on each dimension and representations on different dimensions are either concatenated or fed into a pooling method to get the final representations of each layer.

### 3.1.3 Making textGCN inductive: heteGCN, InducT-GCN, T-VGAE

**HeteGCN** (Ragesh et al. 2021) HeteGCN (Heterogeneous GCN) optimizes the TextGCN by decomposing the TextGCN undirected graph into several directed subgraphs. Several subgraphs from the TextGCN graph are combined sequentially as different layers: feature graph (word-word graph), feature-document graph (word-document graph), and document-feature graph (document-word graph). Different combinations were tested and the best model is shown as:

$$\mathbf{Z} = \text{softmax}(\mathbf{A}_{w-d}(\text{ReLU}(\mathbf{A}_{w-w}\mathbf{X}_w\mathbf{W}^{(0)}))\mathbf{W}^{(1)}) \quad (16)$$

where  $\mathbf{A}_{w-w}$  and  $\mathbf{A}_{w-d}$  show the adjacency matrix for the word-word subgraph and word-document subgraph. Since the input of HeteGCN is the word node embeddings without using document nodes, it can also work in an inductive way while the previous corpus-level graph text classification models are all transductive models.

**InducT-GCN** (Wang et al. 2022) InducT-GCN (InducTive Text GCN) aims to extend the transductive TextGCN into an inductive model. Instead of using the whole corpus to build the graph, InducT-GCN builds a training corpus graph and makes the input embedding of the document the TF-IDF vectors, aligning with the one-hot word embeddings. The weights are learned following TextGCN but InducT-GCN builds virtual subgraphs for prediction on new test documents.

**T-VGAE** (Xie et al. 2021) T-VGAE (Topic Variational Graph Auto-Encoder) applies Variational Graph Auto-Encoder on the latent topic of each document to make the model inductive. A vocabulary graph  $A_v$  connects the words using PMI values, is constructed while each document is represented using the TF-IDF vector. All the document vectors are stacked into a matrix which can also be treated as a bipartite graph  $A_d$ . Two graph auto-encoder models are applied on  $A_v$  and  $A_d$ , respectively. The overall workflow shows as:

$$\mathbf{Z}_v = \text{Encoder}_{GCN}(\mathbf{A}_v, \mathbf{X}_v); \quad (17)$$

$$\mathbf{Z}_d = \text{Encoder}_{UDMP}(\mathbf{A}_d, \mathbf{Z}_v); \quad (18)$$

$$\mathbf{A}_v^* = \text{Decoder}(\mathbf{Z}_v); \quad (19)$$

$$\mathbf{A}_d^* = \text{Decoder}(\mathbf{Z}_d, \mathbf{Z}_v). \quad (20)$$

where  $X^v$  is an Identity Matrix. The Encoder<sub>GCN</sub> and the decoders are applied following VGAE (Kipf and Welling 2016) while Encoder<sub>UDMP</sub> is an unidirectional message passing variant of Encoder<sub>GCN</sub>. The training objective is to minimise the reconstruction error, and  $Z_d$  is used for the classification task.

### 3.2 Document nodes as a graph

To show the global structure of the corpus directly, some models only adopt document nodes in the non-heterogeneous graph.

**knn-GCN** (Benamira et al. 2019) knn-GCN constructs a k-nearest-neighbours graph by connecting the documents with their  $K$  nearest neighbours using Euclidean distances of the embedding of each document. The embedding is generated in an unsupervised way: either using the mean of pretrained GloVe word vectors or applying LDA (Blei et al. 2003). Both GCN and Attention-based GNN (Thekumparampil et al. 2018) are used as the graph model.

**TextGTL** (Li et al. 2021) Similar to TensorGCN, TextGTL (Text-oriented Graph-based Transductive Learning) constructs three different document graphs: Semantics Text Graph, Syntax Text Graph, and Context Text Graph, while all the graphs are non-heterogeneous. Semantics Text Graph uses Generalized Canonical Correlation Analysis (Bach and Jordan 2002) and trains a classifier to determine the edge values between two document nodes. Syntax Text Graph uses the Stanford CoreNLP dependency parser (Manning et al. 2014) to construct units and also trains a classifier. Context Text Graph defines the edge values by summing up the PMI values of the overlapping words in two documents. Two GCN layers are applied, and the output of each graph is mixed as the output of this layer and input for the next layer for all three graphs:

$$\mathbf{H}^{(1)} = \sigma(\mathbf{A}\mathbf{H}^{(0)}\mathbf{W}^{(0)}); \quad (21)$$

$$\mathbf{H}^{(2)} = \sigma(\mathbf{A}[\mathbf{H}_{sem}^{(1)} || \mathbf{H}_{syn}^{(1)} || \mathbf{H}_{seq}^{(1)}]\mathbf{W}^{(1)}); \quad (22)$$

$$\mathbf{Z} = \text{Pooling}_{mean}(\mathbf{H}_{sem}^{(2)}, \mathbf{H}_{syn}^{(2)}, \mathbf{H}_{seq}^{(2)}). \quad (23)$$

where  $\mathbf{H}^{(0)}$  is the TF-IDF vector of the documents. Data augmentation with super nodes is also applied in TextGTL to strengthen the information in graph models.

### 3.3 Word nodes as a graph

By neglecting the document nodes in the graph, a graph with only word nodes shows good performance in deriving the graph-based embedding and is used for downstream tasks. Since no document nodes are included, this method can be easily adapted as an inductive learning model.

**VGCN-BERT** (Lu et al. 2020) VGCN-BERT enhances the input embedding of BERT by concatenating it with the graph embedding. It first constructs a vocabulary graph and uses PMI as the edge value. A variant of the GCN layer called VGCN (Vocabulary GCN) is applied to derive the graph word embedding:

$$\mathbf{X}_{Graph} = \text{ReLU}(\mathbf{X}_{BERT}\mathbf{A}\mathbf{W}^{(0)})\mathbf{W}^{(1)} \quad (24)$$

Where BERT embedding is used as the input, the graph word embeddings are concatenated with BERT embedding and fed into the BERT as extra information.

### 3.4 Extra topic nodes in the graph

Topic information of each document can also provide extra information in corpus-level graph neural networks. Several models also include topic nodes in the graph.

#### 3.4.1 Single layer topic nodes: HGAT, STGCN

**HGAT** (Linmei et al. 2019) HGAT (Heterogeneous GAT) applies LDA (Blei et al. 2003) to extract topic information for each document; top  $P$  topics with the largest probabilities are selected as connected with the document. Instead of using the words directly to utilize the external knowledge, HGAT applies the entity linking tool TAGME<sup>1</sup> to identify the entities in the document and connect them. The semantic similarity between entities using pre-trained Word2vec with threshold is used to define the connectedness between entity nodes. Since the graph is a heterogeneous graph, a HIN (heterogeneous information network) model is implemented, which propagates solely on each sub-graph depending on the type of node. An HGAT model is applied by considering type-level attention and node-level attention. For a given node, the type-level attention learns the weights of different types of neighbouring nodes while node-level attention captures the importance of different neighbouring nodes when ignoring the type. By using the dual attention mechanism, HGAT can capture the information of type and node at the same time.

**STGCN** (Yan et al. 2013) In terms of short text classification, STGCN (Short-Text GCN) applies BTM to get topic information to avoid the data sparsity problem from LDA. The graph is constructed following TextGCN while extra topic nodes are included. Word-topic and document-topic edge values are from BTM, and a classical two-layer GCN is applied. The word embeddings learned from STGCN are concatenated with BERT embeddings and a bi-LSTM model is applied for final prediction.

#### 3.4.2 Multi-layer topic nodes: DHTG

**DHTG** (Wang et al. 2020) To capture different levels of information, DHTG (Dynamic Hierarchical Topic Graph) introduces hierarchical topic-level nodes in the graph from fine-grain to coarse. Poisson gamma belief network (PGBN) (Zhou et al. 2015) is used as a probabilistic deep topic model. The first-layer topics are from the combination of words, while deeper layers are generated by previous layers' topics with the weights of PGBN, and the weights serve as the edge values of each layer of topics. The cosine similarity is chosen as the edge value for the topics on the same layer. A two-layer GCN is applied, and the model is learned jointly with PGBN, which makes the edge of the topics dynamic.

## 3.5 Critical analysis

Compared with sequential models like CNN and LSTM, corpus-level GNN is able to capture the global corpus structure information with word nodes as bridges between document nodes and shows great performance without using external resources like pre-trained embedding or pre-trained model. However, the improvement in performance is marginal

<sup>1</sup> <https://sobigdata.d4science.org/group/tagme/>.

**Table 2** Models' detailed comparison in terms of whether using external resources, how to construct the edge and node input, and whether transductive learning or inductive learning

Graph	Model	Year	External Resource	Edge Construction	Node Initialization	Learning
Corpus-level	TextGCN (Yao et al. 2019)	2019	N/A	pmi, tf-idf	One-hot	Transductive
	SGC (Wu et al. 2019)	2019	N/A	pmi, tf-idf	One-hot	Transductive
	S2GC (Zhu and Koniusz 2020)	2020	N/A	pmi, tf-idf	One-hot	Transductive
	NMGCF (Lei et al. 2021)	2021	N/A	pmi, tf-idf	One-hot	Transductive
	TG-transformer (Zhang and Zhang 2020)	2020	GloVe	pmi, tf-idf	GloVe	Transductive
	BERTGCN (Lin et al. 2021)	2021	BERT	pmi, tf-idf	Doc: 0 word; BERT emb	Transductive
	TensorGCN (Liu et al. 2020)	2020	GloVe, CoreNLP	emb sim, dep graph, pmi, tf-idf	One-hot	Transductive
	ME-GCN (Wang et al. 2022)	2022	N/A	emb sim, tf-idf	Trained Word2vec/doc2vec	Transductive
	HeteGCN (Ragesh et al. 2021)	2021	N/A	pmi, tf-idf	One-hot	Inductive
	InductT-GCN (Wang et al. 2022)	2022	N/A	pmi, tf-idf	One-hot, tf-idf vectors	Inductive
Text-level	T-VGAE (Xie et al. 2021)	2021	N/A	pmi	One-hot	Inductive
	VGCN-BERT (Lu et al. 2020)	2020	BERT	pmi	BERT emb	Transductive
	knn-GCN (Benamira et al. 2019)	2019	GloVe	emb sim	GloVe	Transductive
	TextGTL (Li et al. 2021)	2021	CoreNLP	Dep graph, pmi	tf-idf vectors	Transductive
	HGAT (Linmei et al. 2019)	2019	TAGME, Word2vec	LDA, entity link, emb sim	tf-idf, LDA, Word2vec	Transductive
	STGCN (Ye et al. 2020)	2020	BERT	pmi, tf-idf, BTM	BERT emb	Transductive
	DHTG (Wang et al. 2020)	2020	N/A	PGBN, pmi, tf-idf	One-hot	Transductive

**Table 2** (continued)

Graph	Model	Year	External Resource	Edge Construction	Node Initialization	Learning
Doc-level	Text-Level-GNN (Huang et al. 2019)	2019	GloVe	Consecutive words	GloVe	Inductive
	MPAD (Nikolentzos et al. 2020)	2020	Word2vec	Consecutive words	Word2vec	Inductive
	TextING (Zhang et al. 2020)	2020	GloVe	Consecutive words	GloVe	Inductive
	MLGNN (Liao et al. 2021)	2021	Word2vec	Consecutive words	Word2vec	Inductive
	DADGNN (Liu et al. 2021)	2021	Word2vec/GloVe	Consecutive words	Word2vec/GloVe	Inductive
	TextSSL (Piao et al. 2021)	2021	GloVe	Consecutive words	GloVe	Inductive
	DAGNN (Wu et al. 2019)	2019	GloVe	pmi	GloVe	Inductive
	ReGNN (Li et al. 2019)	2019	GloVe	Consecutive words, pmi	GloVe	Inductive
	GFN (Dai et al. 2022)	2022	GloVe	pmi, emb sim	GloVe	Inductive
	HyperGAT (Ding et al. 2020)	2020	N/A	LDA, consecutive words	One-hot	Inductive
	IGCN (Tang et al. 2020)	2020	spaCy	Dep graph	LSTM emb	Inductive
	GTNT (Mei et al. 2021)	2021	Word2vec/GloVe	tf-idf sorted value	Word2vec/GloVe	Inductive

GloVe and Word2vec are pretrained if not specified. ‘emb sim’ is short for ‘embedding similarity’, ‘dep graph’ is short for ‘dependency graph’

when pretrained embedding is included. Another issue is that most corpus-level GNN is transductive learning, which is not applicable in the real world. Meanwhile, constructing the whole corpus into a graph requires large memory space, especially when the dataset is large.

A detailed comparison of corpus-level GNN is displayed in Table 2.

## 4 Document-level GNN for text classification

By constructing the graph based on each document, a graph classification model can be used as a text classification model. Since each document is represented by one graph and new graphs can be built for test documents, the model can easily work in an inductive way.

### 4.1 Local word consecutive graph

The simplest way to convert a document into a graph with words as nodes is by connecting the consecutive words within a sliding window.

#### 4.1.1 Simple consecutive graph models: text-Level-GNN, MPAD, TextING

**Text-Level-GNN** (Huang et al. 2019) Text-Level-GNN applies a small sliding window and constructs the graph with a small number of nodes and edges in each graph, which saves memory and computation time. The edge value is trainable and shared across the graphs when connecting the same two words, which also brings global information.

Unlike corpus-level graph models, Text-Level-GNN applies a message passing mechanism (MPM) (Gilmer et al. 2017) instead of GCN for graph learning. For each node, the neighbour information is aggregated using max-pooling with trainable edge values as the AGGREGATE function and then the weighted sum is used as the COMBINE function. Sum-pooling and an MLP classifier are applied as the READOUT function to get the representation of each graph. The propagation shows as:

$$\mathbf{h}_i^{(l+1)} = (1 - \alpha)(\max_{n \in \mathcal{N}_i} e_{ni} \mathbf{h}_n^{(l)}) + \alpha \mathbf{h}_i^{(l)}; \quad (25)$$

$$z_i = \text{softmax}(\mathbf{W} \Sigma_i \mathbf{h}_i + \mathbf{b}). \quad (26)$$

where  $\mathbf{h}_i^{(l)}$  is  $i$ th word node presentation of layer  $l$ ,  $e_{ni}$  is edge weight from node  $n$  to node  $i$ . A two-layer MPM is applied, and the input of each graph is pretrained GloVe vectors.

**MPAD** (Nikolentzos et al. 2020) MPAD (Message Passing Attention Networks) connects words within a sliding window of size 2 but also includes an additional master node connecting all nodes in the graph. The edge only shows the connectedness of each pair of word nodes and is fixed. A variant of Gated Graph Neural Networks is applied where the AGGREGATE function is the weighted sum and the COMBINE function is GRU (Chung et al. 2014). Self-attention is applied in the READOUT function.

To learn the high-level information, the master node is directly concatenated with the READOUT output, working as a skip connection mechanism. Each layer's READOUT results are concatenated to capture multi-granularity information to get the final representation. Pretrained Word2vec is used as the initialization of word nodes input.

**TextING** (Zhang et al. 2020) To simplify MPAD, TextING ignores the master node in the document-level graphs, which makes the graph sparser. Compared with Text-Level-GNN, TextING has fixed edges. A similar AGGREGATE and COMBINE function are applied under the concept of e-gated Graph Neural Networks (GGNN) (Li et al. 2016) with the weighted sum and GRU. However, for the READOUT function, soft attention is used and both max-pooling and mean-pooling are applied to make sure that "every word plays a role in the text and the keywords should contribute more explicitly".

#### 4.1.2 Advanced graph models: MLGNN, TextSSL, DADGNN

**MLGNN** (Liao et al. 2021) MLGNN (Multi-level GNN) builds the same graph as TextING but introduces three levels of MPM: bottom-level, middle-level and top-level. In the bottom-level MPM, the same method with Text-Level-GNN is applied with pretrained Word2vec as input embedding but the edge is non-trainable. A larger window size is adopted in the middle level, and Graph Attention Networks (GAT) (Veličković et al. 2018) is applied to learn distant word node information. In the top-level MPM, all word nodes are connected, and multi-head self-attention (Vaswani et al. 2017) is applied. By applying three different levels of MPM, MLGNN learns multi-granularity information well.

**DADGNN** (Liu et al. 2021) DADGNN (Deep Attention Diffusion GNN) constructs the same graph as TextING but uses attention diffusion to overcome the over-smoothing issue. Pretrained word embedding is used as the input of each node and an MLP layer is applied. Then, the graph attention matrix is calculated based on the attention to the hidden states of each node. The diffusion matrix is calculated as

$$\mathbf{T} = \sum_{n=0}^{\infty} \epsilon_n \mathbf{A}^n \quad (27)$$

where  $\mathbf{A}$  is the graph attention matrix and  $\epsilon$  is the learnable coefficients.  $\mathbf{A}^n$  plays a role of connecting  $n$ -hop neighbours and (Liu et al. 2021) uses  $n \in [4, 7]$  in practice. A multi-head diffusion matrix is applied for layer propagation.

**TextSSL** (Piao et al. 2021) To solve the word ambiguity problem and show the word synonymy and dynamic contextual dependency, TextSSL (Sparse Structure Learning) simultaneously learns the graph using intra-sentence and inter-sentence neighbours. The local syntactic neighbour is defined as the consecutive words, and trainable edges across graphs are also included by using Gumbel-softmax. By applying sparse structure learning, TextSSL manages to select edges with dynamic contextual dependencies.

## 4.2 Global word co-occurrence graph

Similar to the TextGCN graph, document-level graphs can also use PMI as the word-word edge values.

### 4.2.1 Only global word co-occurrence: DAGNN

**DAGNN** (Wu et al. 2019) To address the long-distance dependency, hierarchical information and cross-domain learning challenges in domain-adversarial text classification tasks, (Wu et al. 2019) propose DAGNN (Domain-Adversarial Graph Neural Network). Each document is represented by a graph with content words as nodes and PMI values as edge values, which can capture long-distance dependency information. Pretrained FastText is chosen as the input word embeddings to handle the out-of-vocabulary issue and a GCN model with skip connection is used to address the over-smoothing problem. The propagation is formulated as:

$$\mathbf{H}^{(l+1)} = (1 - \alpha)\tilde{\mathbf{A}}\mathbf{H}^{(l)} + \alpha\mathbf{H}^{(0)} \quad (28)$$

To learn the hierarchical information of documents, DiffPool (Ying et al. 2018) is applied to assign each document into a set of clusters. Finally, adversarial training minimises the loss on source tasks and maximises the differentiation between source and target tasks.

### 4.2.2 Combine with extra edges: ReGNN, GFN

**ReGNN** (Li et al. 2019) ReGNN (Recursive Graphical Neural Network) uses PMI together with consecutive words as the word edges to capture global and local information. The graph propagation function is the same as GGNN while additive attention (Bahdanau et al. 2015) is applied in aggregation. Pretrained GloVe is the input embedding of each word node.

**GFN** (Dai et al. 2022) GFN (Graph Fusion Network) builds four types of graphs using the word co-occurrence statistics, PMI, the similarity of pretrained embedding and Euclidean distance of pretrained embedding. Although four corpus-level graphs are built, the graph learning happens on each document's subgraphs, making the method a document-level GNN. For each subgraph, each type of graph is learned separately using the graph convolutional method, and then a fusion method of concatenation is used. After an MLP layer, average pooling is applied to get the document representation.

## 4.3 Other word graphs

Some other ways of connecting words in a document have been explored.

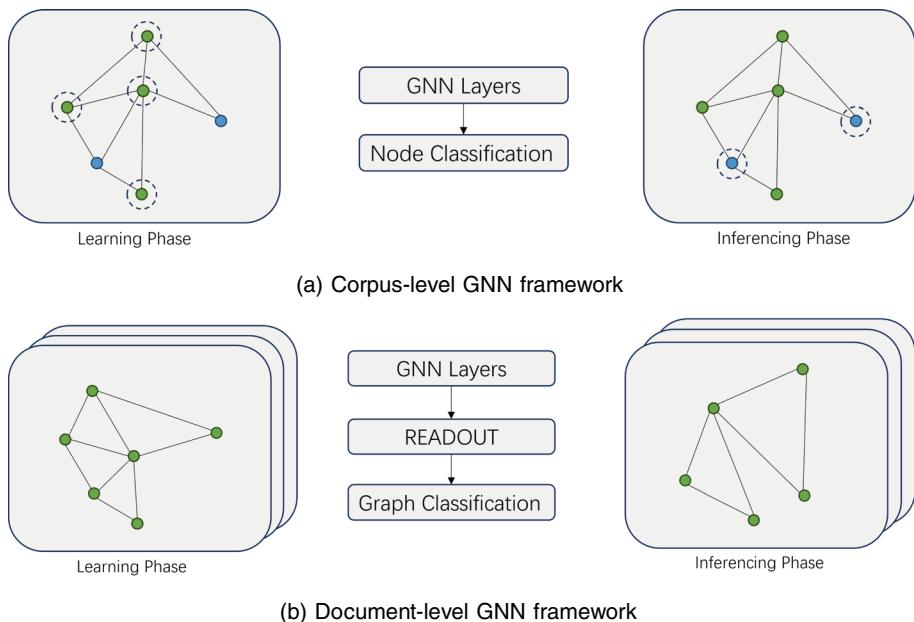
**HyperGAT** (Ding et al. 2020) (Ding et al. 2020) proposes HyperGAT (Hypergraph Attention Networks), which builds hypergraphs for each document to capture high-level interaction between words. Two types of hyperedges are included: sequential hyperedges connecting all words in a sentence and semantic hyperedges connecting top-K words after getting the topic of each word using LDA. Like traditional hypergraph propagations, HyperGAT follows the same two steps of updating but with an attention mechanism to highlight the key information: Node-level attention is applied to learn hyperedges representations, and edge-level attention is used to update node representations.

**IGCN** (Tang et al. 2020) Contextual dependency helps in understanding a document, and the graph neural network is no exception. IGCN constructs the graph with the dependency graph to show the connectedness of each pair of words in a document. Then, the word representation learned from Bi-LSTM using POS embedding and word embedding is used to calculate the similarity between each pair of nodes. Attention is used for the output to find the important relevant semantic features.

**GTNT** (Mei et al. 2021) Words with higher TF-IDF values should connect to more word nodes, with this in mind, GTNT (Graph Transformer Networks based Text representation) uses sorted TF-IDF value to determine the degree of each node and applies the Havel-Hakimi algorithm (Hakami 1962) to determine the edges between word nodes. A variant of GAT is applied during model learning. Despite the fact that GAT’s attention score is mutual for two nodes, GTNT uses relevant importance to adjust the attention score from one node to another. Pretrained Word2vec is applied as the input of each node.

**Table 3** A Comparison between Corpus-level GNN and Document-level GNN

	Corpus-level GNN	Document-level GNN
Number of Graphs	One or multiple graph (s) per corpus	One graph per document
Learning Methods	Mostly transductive	Inductive
Performance	Normally performs better under the same setting	Normally performs worse under the same setting
Generalisation	Bad due to the transductive learning	Good
Scalability	OOM issues when the number of documents grows	OOM issues when the number of words in a document grows
Efficiency	High, one iteration to generate all predictions	Low, batch predictions



**Fig. 2** Corpus-level GNN usually builds a single graph per corpus and learns the node representation while Document-level GNN usually builds one graph per document and learns the graph representation

## 4.4 Critical analysis

Most document-level GNNs connect consecutive words as edges in the graph and apply a graph neural network model, which makes them similar to CNN, where the receptive field enlarges when graph models go deeper. Also, the major differences among document-level GNNs are the details of graph models, e.g. different pooling methods and different attention calculations, which diminishes the impact of the contribution of these works. Compared with corpus-level GNN, document-level GNN adopts more complex graph models and also suffers from the out-of-memory issue when the number of words in a document is large. A detailed comparison of document-level GNN is displayed in Table 2.

## 4.5 Comparison between corpus-level and document-level GNN

The comparison of the framework between Corpus-level and Document-level GNN's learning is shown in Fig. 2. A comprehensive comparison between corpus-level GNN and document-level GNN can be found in Table 3.

## 5 Datasets and metrics

### 5.1 Datasets

There are many popular text classification benchmark datasets, while this paper mainly focuses on the datasets used by GNN-based text classification applications. Based on the purpose of applications, we divided the commonly adopted datasets into three types including *Topic Classification*, *Sentiment Analysis* and *Other*. Most of these text classification datasets contain a single target label of each text body. The key information of each dataset is listed in Table 4.

#### 5.1.1 Topic classification

Topic classification models aim to classify input text bodies from diverse sources into pre-defined categories. News categorization is a typical topic classification task to obtain key information from news and classify them into corresponding topics. The input text bodies normally are paragraphs or whole documents especially for news categorization, while there are still some short text classification datasets from certain domains such as micro-blogs, bibliography, etc. Some typical datasets are listed:

- **Ohsumed** (Joachims 1998) is acquired from the MEDLINE database and further processed by Yao et al. (2019) via selecting certain documents (abstracts) and filtering out the documents belonging to multiple categories. Those documents are classified into 23 cardiovascular diseases. The statistics of Yao et al. (2019) processed Ohsumed dataset is represented in Table 4, which is directly employed by other related works.

**Table 4** Commonly used text classification datasets by GNN-based models

Task	Name	Time	Domain	# Cat.	# Docs	# Train	# Test	# Words	# Len.	Ave	Models
Topic classification	Ohsuned	2004	Bibliography	23	7400	3357	4043	14,157	135,82	Linmei et al. (2019); Wang et al. (2020); Xie et al. (2021); Yao et al. (2019); Lin et al. (2021); Piao et al. (2021); Liu et al. (2020); Lei et al. (2021); Zhang and Zhang (2020); Ding et al. (2020); Zhang et al. (2020); Li et al. (2021); (2019)	
R8	1997	News		8	7674	5485	2189	7688	65.72	Liu et al. (2021); Wu et al. (2019); Zhang and Zhang (2020); Yao et al. (2019); Wang et al. (2020); Xie et al. (2021); Lin et al. (2021); Piao et al. (2021); Liu et al. (2020); Lei et al. (2021); Linmei et al. (2019); Ding et al. (2020); Zhang et al. (2020); Li et al. (2021); Huang et al. (2021); Li et al. (2019)	
R52	2002	News		52	9100	6532	2568	8892	69.82	Liu et al. (2021); Wu et al. (2019); Zhang and Zhang (2020); Yao et al. (2019); Wang et al. (2020); Xie et al. (2021); Lin et al. (2021); Piao et al. (2021); Liu et al. (2020); Lei et al. (2021); Ding et al. (2020); Zhang et al. (2019); Linmei et al. (2019); (Huang et al. 2019); Li et al. (2019, 2021)	
20NG	2003	News		20	18,846	11,314	7532	42,757	221.26	Wang et al. (2020); Wu et al. (2019); Xie et al. (2021); Yao et al. (2019); Lin et al. (2021); Piao et al. (2021); Liu et al. (2020); Lei et al. (2021); Linmei et al. (2019); Ding et al. (2020); Li et al. (2021); (Liu et al. 2021))	
AG-News	2004	News		4	127,600	120,000	7600	128,515	44.03		
WebKB	1998	Web Page		7	4199	2803	1396	7771	133.37		
TREC	2002	Questions		6	5952	5452	500	9593	10.06	Liu et al. (2021)	
DBLP	2017	Bibliography		6	81,479	61,479	20,000	25,549	8.51	Liu et al. (2021)	
DBpedia	2016	Wikipedia		14	630000	560000	70000	—	—	Linmei et al. (2019)	

**Table 4** (continued)

Task	Name	Time	Domain	# Cat.	# Docs	# Train	# Test	# Words	# Len.	Ave	Models
Sentiment analysis	MR	2005	Movie review	2	10,662	7108	3554	18,764	20,39	Lu et al. (2020); Wang et al. (2020); Xie et al. (2021); Yao et al. (2019); Liu et al. (2021); Lin et al. (2021); Piao et al. (2021); Liu et al. (2020); Lei et al. (2021); Linmei et al. (2019); Ding et al. (2020); Zhang et al. (2020); Li et al. (2021); Huang et al. (2019); Li et al. (2019); Ye et al. (2020)	
AAR		2019	Product review	2	3150	1575	1575	—	—	Tang et al. (2020)	
TUA		2015	Airline comments	2	14640	7320	7320	—	—	Tang et al. (2020)	
SST-1		2005	Movie review	5	11,855	9465	2210	19,524	20,17	Liu et al. (2021)	
SST-2		2005	Movie review	2	9613	7792	1821	17339	19,67	Liu et al. (2021); Lu et al. (2020)	
IMDB		2011	Movie review	2	50,000	25,000	25,000	71,278	232,77	Liu et al. (2021); Tang et al. (2020); Zhang and Zhang (2020)	
Yelp 2014		2015	Review rating	5	1,125,386	900,309	225,077	476,191	148,8	Zhang and Zhang (2020)	
SentTube-A		2014	Youtube Comment	2	7400	3357	4043	14,157	28,54	Liao et al. (2021)	
SentTube-T		2014	Youtube Comment	2	6664	4997+333	1334	20,276	28,73	Liao et al. (2021)	
Other	ArangoHate	2019	Twitter posts	2	7006	—	—	—	13,3	Lu et al. (2020)	
	FountaHate	2018	Twitter posts	4	99996	—	—	—	15,7	Lu et al. (2020)	

- **R8 / R52** are two subsets of the Reuters 21587 dataset<sup>2</sup> which contain 8 and 52 news topics from Reuters financial news services, respectively.
- **20NG** is another widely used news categorization dataset that contains 20 newsgroups. originally collected it citeLang95, but the procedures are not explicitly described.
- **AG News** (Zhang et al. 2015) is a large-scale news categorization dataset compared with other commonly used datasets, which are constructed by selecting the top-4 largest categories from the AG corpus. Each news topic contains 30,000 samples for training and 1900 samples for testing.
- **Database systems and logic programming (DBLP)** is a topic classification dataset to classify the computer science paper titles into six various topics (Mei et al. 2021). Different from paragraph or document based topic classification dataset, DBLP aims to categorise scientific paper titles into corresponding categories, the average input sentence length is much lower than others.
- **Dbpedia** (Lehmann et al. 2015) is a large-scale multilingual knowledge base that contains 14 non-overlapping categories. Each category contains 40000 samples for training and 5000 samples for testing.
- **WebKB** (Craven et al. 1998) is a long corpus web page topic classification dataset.
- **TREC** (Li and Roth 2002) is a question topic classification dataset to categorise one question sentence into 6 question categories.

### 5.1.2 Sentiment analysis

The purpose of sentiment analysis is to analyse and mine the opinion of the textual content which could be treated as a binary or multi-class classification problem. The sources of existing sentiment analysis tasks come from movie reviews, product reviews or user comments, social media posts, etc. Most sentiment analysis datasets aim to predict people's opinions of one or two input sentences, of which the average length of each input text body is around 25 tokens.

- **Movie review (MR)** (Pang and Lee 2005) is a binary sentiment classification dataset for movie reviews, which contains positive and negative data equally distributed. Each review only contains one sentence.
- **Stanford sentiment treebank (SST)** (Socher et al. 2013) is an upgraded version of MR which contains two subsets SST-1 and SST-2. SST-1 provides five fine-grained labels, while SST-2 is a binary sentiment classification dataset.
- **Internet movie database (IMDB)** (Maas et al. 2011) is also an equally distributed binary classification dataset for sentiment analysis. Different from other short text classification datasets, the average number of words in each review is around 221.
- **Yelp 2014** (Tang et al. 2015) is a large-scale binary category-based sentiment analysis dataset for longer user reviews collected from Yelp.com.

GNN-based text classifiers also use certain binary sentiment classification benchmark datasets. Most of them are gathered from shorter user reviews or comments (normally one or two sentences) from different websites including Amazon Alexa Reviews (**AAR**), Twitter

<sup>2</sup> For the original Reuters 21587 dataset, please refer to this link <http://www.daviddlewis.com/resources/testcollections/reuters21578>.

US Airline (**TUA**), Youtube comments (**SenTube-A** and **SenTube-T**) (Uryupina et al. 2014).

### 5.1.3 Other datasets

There are some datasets targeting other tasks, including hate detection, grammaticality checking, etc. For example, **ArangoHate** (Arango et al. 2019) is a hate detection dataset, a sub-task of intent detection, which contains 2920 hateful documents and 4086 normal documents by resampling the merged datasets from Davidson et al. (2017) and Waseem (2016). In addition, Founta et al. (2018) proposes another large-scale hate language detection dataset, namely **FountaHate** to classify the tweets into four categories, including 53,851, 14,030, 27,150, and 4,965 samples of normal, spam, hateful and abusive, respectively. Since there is no officially provided training and testing splitting ratio for the above datasets, the numbers represented in Table 4 follow the ratios (train/development/test is 85:5:10) defined by Lu et al. (2020).

### 5.1.4 Dataset summary

Since an obvious limitation of corpus-level GNN models has high memory consumption limitation (Zhang and Zhang 2020; Huang et al. 2019; Ding et al. 2020), the datasets with a smaller number of documents and vocabulary sizes such as Ohsmed, R8/R52, 20NG or MR are widely used to ensure feasibly build and evaluate corpus-level graphs. For the document-level GNN-based models, some larger-size datasets like AG-News can be adopted without considering the memory consumption problem. From Table 4, we could find most of the related works mainly focus on the GNN applied in topic classification and sentiment analysis, which means the role of GNNs in other text classification tasks such as spam detection, intent detection, abstractive question answering need to be further exploited.

## 5.2 Evaluation methods

### 5.2.1 Performance metrics

In evaluating and comparing the performance of the proposed models with other baselines, accuracy and F1 are the most commonly used metrics to conduct overall performance analysis, ablation studies, and breakdown analysis. We use *TP*, *FP*, *TN* and *FN* to represent the number of true positive, false positive, true negative and false negative samples. *N* is the total number of samples.

- **Accuracy** and **error rate** are basic evaluation metrics adopted by many GNN-based text classifiers such as Li et al. (2021); Liu et al. (2016); Wang et al. (2020); Yao et al. (2019); Zhang and Zhang (2020). Most of the related papers run all baselines and their models ten times or five times to show the *mean ± standard deviation* of accuracy for reporting more convincing results. It can be defined as:

$$\text{Accuracy} = \frac{(TF + TN)}{N}; \quad (29)$$

$$\text{ErrorRate} = 1 - \text{Accuracy} = \frac{(FP + FN)}{N}. \quad (30)$$

- **Precision, recall** and **F1** are metrics for measuring the performance, especially for imbalanced datasets. Precision is used to measure the result's relevancy, while recall is utilized to measure how many truly relevant results are acquired. By calculating the harmonic average of Precision and Recall, we could get F1. Those three measurements can be defined as:

$$\text{Precision} = \frac{TP}{(TP + FP)}; \quad (31)$$

$$\text{Recall} = \frac{TP}{(TP + FN)}; \quad (32)$$

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{(\text{Precision} + \text{Recall})}. \quad (33)$$

Few papers only utilise recall or precision to evaluate the performance (Mei et al. 2021). However, precision and recall are more commonly used together with F1 or Accuracy to evaluate and analyse the performance from different perspectives, e.g. Li et al. (2019); Linmei et al. (2019); Lu et al. (2020); Xie et al. (2021). In addition, based on different application scenarios, different F1 averaging methods are adopted by those papers to measure the overall F1 score of multi-class (Number of Classes is  $C$ ) classification tasks, including:

- **Macro-F1** applies the same weights to all categories to get overall  $F1_{macro}$  by taking the arithmetic mean.

$$F1_{macro} = \frac{1}{C} \sum_{i=1}^C F1_i \quad (34)$$

- **Micro-F1** is calculated by considering the overall  $P_{micro}$  and  $R_{micro}$ . It can be defined as:

$$F1_{micro} = \frac{2 \times P_{micro} \times R_{micro}}{(P_{micro} + R_{micro})} \quad (35)$$

where:

$$P_{micro} = \frac{\sum_{i \in C} TP_i}{\sum_{i \in C} TP_i + FP_i}, R_{micro} = \frac{\sum_{i \in C} TP_i}{\sum_{i \in C} TP_i + FN_i} \quad (36)$$

- **Weighted-F1** is the weighted mean of F1 of each category where the weight  $W_i$  is related to the number of occurrences of the corresponding  $i$ th class, which can be defined as:

$$F1_{macro} = \sum_{i=1}^C F1_i \times W_i \quad (37)$$

### 5.2.2 Other evaluation aspects

Since two limitations of GNN-based models are time and memory consumption, except the commonly used qualitative performance comparison, representing and comparing the GPU or CPU memory consumption and the training time efficiency of proposed models are also adopted by many related studies to demonstrate the practicality in real-world applications. In addition, based on the novelties of various models, specific evaluation methods are conducted to demonstrate the proposed contributions.

- **Memory consumption** (Ding et al. 2020; Huang et al. 2019; Liu et al. 2021) lists the memory consumption of different models for comprehensively evaluating the proposed models in computational efficiency aspect.
- **Time measurement** (Ragesh et al. 2021; Pasa et al. 2021) performs performance training time comparison between their proposed models and baselines on different benchmarks. Due to the doubts about the efficiency of applying GNNs for text classification, it is an effective way to demonstrate they could balance performance and time efficiency.
- **Parameter sensitivity** is commonly conducted by GNNs studies to investigate the effect of different hyperparameters, e.g. varying sliding window sizes, embedding dimensions of proposed models to represent the model sensitivity via line chart such as Linmei et al. (2019); Ding et al. (2020); Liu et al. (2021).
- **Number of labelled documents** is a widely adopted evaluation method by GNN-based text classification models (Li et al. 2021; Wang et al. 2020; Linmei et al. 2019; Mei et al. 2021; Yao et al. 2019; Ragesh et al. 2021; Ding et al. 2020) which mainly analyse the performance trend by using different proportions of training data to test whether the proposed model can work well under the limited labelled training data.
- **Vocabulary size** is similar to the number of labelled documents, but it investigates the effects of using different sizes of vocabulary during the GNN training stage adopted by Wang et al. (2020).

### 5.2.3 Metrics summary

For general text classification tasks, accuracy, precision, recall, and varying F1 are commonly used evaluation metrics for comparison with other baselines. However, for GNN-based models, only representing the model performance cannot effectively represent the multi-aspects of the proposed models. In this case, there are many papers conducting external processes to evaluate and analyse the GNN-based classifier from multiple views, including time and memory consumption, model sensitivity and dataset quantity.

## 6 Performance

While different GNN text classification models may be evaluated on different datasets, there are some datasets that are commonly used across many of these models, including

**Table 5** Performance Table

Type	Method	External Resource	20NG	R8	R52	Ohsuned	MR
Corpus-level	TextGCN (Yao et al. 2019)	N/A	86.34 ± 0.09	97.07 ± 0.10	93.56 ± 0.18	68.36 ± 0.56	76.74 ± 0.20
	SGC (Wu et al. 2019)	N/A	88.5 ± 0.1	97.2 ± 0.1	94.0 ± 0.2	68.5 ± 0.3	75.9 ± 0.3
	S2GC (Zhu and Koniusz 2020)	N/A	88.6 ± 0.1	97.4 ± 0.1	94.5 ± 0.2	68.5 ± 0.1	76.7 ± 0.0
	TG-transformer (Zhang and Zhang 2020)	GloVe	—	98.1 ± 0.1	95.2 ± 0.2	70.4 ± 0.4	—
	DHTG (Wang et al. 2020)	N/A	87.13 ± 0.07	97.33 ± 0.06	93.93 ± 0.10	68.80 ± 0.33	77.21 ± 0.11
	TensorGCN (Liu et al. 2020)	GloVe, CoreNLP	87.74 ± 0.05	98.04 ± 0.08	95.05 ± 0.11	70.11 ± 0.24	77.91 ± 0.07
	STGCN (Ye et al. 2020)	BERT	—	98.5	—	—	82.5
	NMGC (Lei et al. 2021)	N/A	86.61 ± 0.06	97.31 ± 0.09	94.35 ± 0.06	69.21 ± 0.17	76.21 ± 0.25
	BertGCN (Lin et al. 2021)	BERT	89.3	98.1	96.6	72.8	86
	RobertaGCN (Lin et al. 2021)	RoBERTa	89.5	98.2	96.1	72.8	89.7
Doc-level	T-VGAE (Xie et al. 2021)	N/A	88.08 ± 0.06	97.68 ± 0.14	95.05 ± 0.10	70.02 ± 0.14	78.03 ± 0.11
	RegGNN (Li et al. 2019)	GloVe	—	97.93 ± 0.31	95.17 ± 0.17	67.93 ± 0.33	78.71 ± 0.56
	Text-Level-GNN (Huang et al. 2019)	GloVe	84.16 ± 0.25*	97.8 ± 0.2	94.6 ± 0.3	69.4 ± 0.6	75.47 ± 0.06*
	TextING (Zhang et al. 2020)	GloVe	—	98.13 ± 0.12	95.68 ± 0.35	70.84 ± 0.52	80.19 ± 0.31
	HyperGAT (Ding et al. 2020)	N/A	86.62 ± 0.16	97.97 ± 0.23	94.98 ± 0.27	69.90 ± 0.34	78.32 ± 0.27
	TextSSL (Piao et al. 2021)	GloVe	85.26 ± 0.28	97.81 ± 0.14	95.48 ± 0.26	70.59 ± 0.38	79.74 ± 0.19

\* : Indicates unavailability. \*\*: Refers to replication from HyperGAT (Ding et al. 2020)

**20NG, R8, R52, Ohsumed and MR.** The accuracy of various models on these five datasets is presented in Table 5. Some of the results are reported with ten times average accuracy and standard derivation while some only report the average accuracy. Several conclusions can be drawn:

- Models that use external resources usually achieve better performance than those that do not, especially models with BERT and RoBERTa (Lin et al. 2021; Ye et al. 2020).
- Under the same setting, such as using GloVe as the external resource, Corpus-level GNN models (e.g. TG-Transformer (Zhang and Zhang 2020), TensorGCN (Liu et al. 2020)) typically outperform Document-level GNN models (e.g. TextING (Zhang et al. 2020), TextSSL (Piao et al. 2021)). This is because Corpus-level GNN models can work in a transductive way and make use of the test input, whereas Document-level GNN models can only use the training data.
- The advantage of Corpus-level GNN models over Document-level GNN models only applies to topic classification datasets and not to sentiment analysis datasets such as **MR**. This is because sentiment analysis involves analyzing the order of words in a text, which is something that most Corpus-level GNN models cannot do.

## 7 Challenges and future work

### 7.1 Model performance

With the development of pre-trained models (Devlin et al. 2018; Liu et al. 2019), and prompt learning methods (Gao et al. 2021; Liu et al. 2021) achieve great performance on text classification. Applying GNNs in text classification without this pre-training style will not be able to achieve such good performance. For both corpus-level and document-level GNN text classification models, researching how to combine GNN models with these pretrained models to improve the pretrained model performance can be the future work. Meanwhile, more advanced graph models can be explored, e.g. more heterogeneous graph models on word and document graphs to improve the model performance.

### 7.2 Graph construction

Most GNN text classification methods use a single, static-value edge to construct graphs based on document statistics. This approach applies to both corpus-level GNN and document-level GNN. However, to better explore the complex relationship between words and documents, more dynamic hyperedges can be utilized. Dynamic edges in GNNs can be learned from various sources, such as the graph structure, document semantic information, or other models. And hyperedges can be built for a more expressive representation of the complex relationships between nodes in the graph.

### 7.3 Application

While corpus-level GNN text classification models have demonstrated good performance without using external resources, these models are mostly transductive. To apply them in real-world settings, an inductive learning approach should be explored. Although some inductive corpus-level GNNs have been introduced, the large amount of space required to

construct the graph and the inconvenience of incremental training still present barriers to deployment. Improving the scalability of online training and testing for inductive corpus-level GNNs represents a promising area for future work.

## 8 Conclusion

This survey article introduces how Graph Neural Networks have been applied to text classification in two different ways: corpus-level GNN and document-level GNN, with a detailed structural figure. Details of these models have been introduced and discussed, along with the datasets commonly used by these methods. Compared with traditional machine learning and sequential deep learning models, graph neural networks can explore the relationship between words and documents in the global structure (corpus-level GNN) or the local document (document-level GNN), performing well. A detailed performance comparison is applied to investigate the influence of external resources, model learning methods, and types of different datasets. Furthermore, we propose the challenges for GNN text classification models and potential future work.

**Author contributions** Kunze Wang: Conceived and designed the analysis and Review, Collected the data, Contributed data or analysis, and Wrote the paper Yihao Ding: Collected the data, Contributed data or analysis, and Wrote the paper Soyeon Caren Han: Conceived and designed the analysis and review, and Research Supervision, and Manuscript Drafting

**Funding** Open Access funding enabled and organized by CAUL and its Member Institutions

**Data availability** No datasets were generated or analysed during the current study.

## Declarations

**Competing interests** The authors declare no competing interests.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Abreu J, Fred L, Macêdo D, Zanchettin C (2019) Hierarchical attentional hybrid neural networks for document classification. In: International Conference on Artificial Neural Networks, Springer, pp. 396–402
- Aggarwal CC, Zhai C (2012) A survey of text classification algorithms. Mining text data. Springer, Boston, pp 163–222
- Alsaedi A (2020) A survey of term weighting schemes for text classification. Int J Data Mining Model Manag 12 (2):237–254

- Arango A, Pérez J, Poblete B (2019) Hate speech detection is not as easy as you may think: A closer look at model validation. In: Proceedings of the 42nd International Acm Sigir Conference on Research and Development in Information Retrieval, pp. 45–54
- Bach FR, Jordan MI (2002) Kernel independent component analysis. *J Mach Learn Res* 3:1–48
- Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint [arXiv:1409.0473](https://arxiv.org/abs/1409.0473)
- Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings
- Battaglia PW, Hamrick JB, Bapst V, Sanchez-Gonzalez A, Zambaldi V, Malinowski M, Tacchetti A, Raposo D, Santoro A, Faulkner R, et al (2018) Relational inductive biases, deep learning, and graph networks. arXiv preprint [arXiv:1806.01261](https://arxiv.org/abs/1806.01261)
- Benamira A, Devillers B, Lesot E, Ray AK, Saadi M, Malliaros FD (2019) Semi-supervised learning and graph neural networks for fake news detection. In: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 568–569
- Bhavani A, Kumar BS (2021) A review of state art of text classification algorithms. In: 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), pp. 1484–1490. IEEE
- Bird S, Klein E, Loper E (2009) Natural language processing with python: analyzing text with the natural language toolkit. O'Reilly Media Inc., Sebastopol
- Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet allocation. *J Mach Learn Res* 3:993–1022
- Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. *Trans Assoc Comput Linguist* 5:135–146
- Bronstein MM, Bruna J, LeCun Y, Szlam A, Vandergheynst P (2017) Geometric deep learning: going beyond Euclidean data. *IEEE Signal Proc Mag* 34 (4):18–42
- Cavnar WB, Trenkle JM, et al (1994) N-gram-based text categorization. In: Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, vol. 161175 . Las Vegas, NV
- Chen Y, Wu L, Zaki M (2020) Iterative deep graph learning for graph neural networks: better and robust node embeddings. *Adva Neural Inf Proc Syst* 33:19314–19326
- Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint [arXiv:1406.1078](https://arxiv.org/abs/1406.1078)
- Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint [arXiv:1412.3555](https://arxiv.org/abs/1412.3555)
- Craven M, McCallum A, PiPasquo D, Mitchell T, Freitag D (1998) Learning to extract symbolic knowledge from the world wide web. Technical report, Carnegie-mellon univ pittsburgh pa school of computer Science
- Dai Y, Shou L, Gong M, Xia X, Kang Z, Xu Z, Jiang D (2022) Graph fusion network for text classification. *Knowl-Based Syst* 236:107659
- Davidson T, Warmsley D, Macy M, Weber I (2017) Automated hate speech detection and the problem of offensive language. In: Proceedings of the International AAAI Conference on Web and Social Media, 11, 512–515
- Deng X, Li Y, Weng J, Zhang J (2019) Feature selection for text classification: a review. *Multimedia Tools Appl* 78 (3):3797–3816
- Devlin J, Chang M-W, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint [arXiv:1810.04805](https://arxiv.org/abs/1810.04805)
- Dieng AB, Wang C, Gao J, Paisley J (2016) Topicrnn: A recurrent neural network with long-range semantic dependency. arXiv preprint [arXiv:1611.01702](https://arxiv.org/abs/1611.01702)
- Ding K, Wang J, Li J, Li D, Liu H (2020) Be more with less: Hypergraph attention networks for inductive text classification. arXiv preprint [arXiv:2011.00387](https://arxiv.org/abs/2011.00387)
- Founta AM, Djouvas C, Chatzakou D, Leontiadis I, Blackburn J, Stringhini G, Vakali A, Sirivianos M, Kourtellis N (2018) Large scale crowdsourcing and characterization of twitter abusive behavior. In: Twelfth International AAAI Conference on Web and Social Media
- Gao T, Fisch A, Chen D (2021) Making pre-trained language models better few-shot learners. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 3816–3830
- Gao C, Wang X, He X, Li Y (2022) Graph neural networks for recommender system. In: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, pp. 1623–1625
- Genkin A, Lewis DD, Madigan D (2007) Large-scale Bayesian logistic regression for text categorization. *Technometrics* 49 (3):291–304

- Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017) Neural message passing for quantum chemistry. In: International Conference on Machine Learning, pp. 1263–1272. PMLR
- Graves A (2012) Long short-term memory. *Superv Seq Label Recurr Neural Netw*. <https://doi.org/10.1007/978-3-642-24797-2>
- Hakami S (1962) On the realizability of a set of integers as degrees of the vertices of a graph. *SIAM J Appl Math* 10:496–506
- Hakim AA, Erwin A, Eng KI, Galinium M, Muliady W (2014) Automated document classification for news article in bahasa indonesia based on term frequency inverse document frequency (tf-idf) approach. In: 2014 6th International Conference on Information Technology and Electrical Engineering (ICITEE), pp. 1–4. IEEE
- Hamilton WL, Ying R, Leskovec J (2017) Representation learning on graphs: Methods and applications. arXiv preprint [arXiv:1709.05584](https://arxiv.org/abs/1709.05584)
- Harish BS, Guru DS, Manjunath S (2010) Representation and classification of text documents: a brief review. *IJCA* 110:119
- Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9 (8):1735–1780
- Huang L, Ma D, Li S, Zhang X, Wang H (2019) Text level graph neural network for text classification. arXiv preprint [arXiv:1910.02356](https://arxiv.org/abs/1910.02356)
- Iyyer M, Manjunatha V, Boyd-Graber J, Daumé III H (2015) Deep unordered composition rivals syntactic methods for text classification. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (volume 1: Long Papers), pp. 1681–1691
- Jindal R, Malhotra R, Jain A (2015) Techniques for text classification: literature review and current trends. *Webology* 12 (2):1–28
- Joachims T (1998) Text categorization with support vector machines: Learning with many relevant features. In: European Conference on Machine Learning, pp. 137–142. Springer
- Joachims T (2005) Text categorization with support vector machines: Learning with many relevant features. In: Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings, pp. 137–142. Springer
- Kadhim AI (2019) Survey on supervised machine learning techniques for automatic text classification. *Artif Intell Rev* 52 (1):273–292
- Kalchbrenner N, Grefenstette E, Blunsom P (2014) A convolutional neural network for modelling sentences. arXiv preprint [arXiv:1404.2188](https://arxiv.org/abs/1404.2188)
- Kaur R, Kumar M (2018) Domain ontology graph approach using markov clustering algorithm for text classification. In: International Conference on Intelligent Computing and Applications, pp. 515–531. Springer
- Khan A, Baharudin B, Lee LH, Khan K (2010) A review of machine learning algorithms for text-documents classification. *J Adv Inf Technol* 1 (1):4–20
- Kim Y (2014) Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, pp. 1746–1751
- Kipf TN, Welling M (2016) Variational graph auto-encoders. NIPS Workshop on Bayesian Deep Learning
- Kowsari K, Jafari Meimandi K, Heidarysafa M, Mendu S, Barnes L, Brown D (2019) Text classification algorithms: a survey. *Information* 10 (4):150
- Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes PN, Hellmann S, Morsey M, Van Kleef P, Auer S et al (2015) Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia. *semantic web* 6 2:167–195. Crossref, ISI (2015)
- Lei F, Liu X, Li Z, Dai Q, Wang S (2021) Multihop neighbor information fusion graph convolutional network for text classification. *Math Probl Eng*. <https://doi.org/10.1155/2021/6665588>
- Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: International Conference on Machine Learning, pp. 1188–1196. PMLR
- Li Q, Peng H, Li J, Xia C, Yang R, Sun L, Yu PS, He L (2022) A survey on text classification: from traditional to deep learning. *ACM Trans Intell Syst Technol* 13 (2):1–41
- Liang Z, Ding H, Fu W (2021) A survey on graph neural networks for recommendation. In: 2021 International Conference on Culture-oriented Science & Technology (ICCST), pp. 383–386. IEEE
- Liao W, Bak-Jensen B, Pillai JR, Wang Y, Wang Y (2021) A review of graph neural networks and their applications in power systems. *J Modern Power Syst Clean Energy* 10 (2):345–360
- Liao W, Zeng B, Liu J, Wei P, Cheng X, Zhang W (2021) Multi-level graph neural network for text sentiment analysis. *Comput Electr Eng* 92:107096
- Li W, Li S, Ma S, He Y, Chen D, Sun X (2019) Recursive graphical neural networks for text classification. arXiv preprint [arXiv:1909.08166](https://arxiv.org/abs/1909.08166)

- Lilleberg J, Zhu Y, Zhang Y (2015) Support vector machines and word2vec for text classification with semantic features. In: 2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI\* CC), pp. 136–140. IEEE
- Lin Y, Meng Y, Sun X, Han Q, Kuang K, Li J, Wu F (2021) Bertgn: Transductive text classification by combining gnn and bert. *Findings Assoc Comput Linguist* 2021:1456–1462
- Linmei H, Yang T, Shi C, Ji H, Li X (2019) Heterogeneous graph attention networks for semi-supervised short text classification. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 4821–4830
- Li C, Peng X, Peng H, Li J, Wang L (2021) Textglt: Graph-based transductive learning for semi-supervised text classification via structure-sensitive interpolation. *IJCAI. ijcai.org*
- Li X, Roth D (2002) Learning question classifiers. In: COLING 2002: The 19th International Conference on Computational Linguistics
- Li Y, Tarlow D, Brockschmidt M, Zemel RS (2016) Gated graph sequence neural networks. In: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings
- Liu Z, Tan H (2021) Traffic prediction with graph neural network: a survey. *CICTP* 2021:467–474
- Liu X, You X, Zhang X, Wu J, Lv P (2020) Tensor graph convolutional networks for text classification. *Proc AAAI Conf Arti Intel* 34:8409–8416
- Liu Y, Guan R, Giunchiglia F, Liang Y, Feng X (2021) Deep attention diffusion graph neural networks for text classification. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 8142–8152
- Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V (2019) Roberta: A robustly optimized bert pretraining approach. arXiv preprint [arXiv:1907.11692](https://arxiv.org/abs/1907.11692)
- Liu P, Qiu X, Chen X, Wu S, Huang X-J (2015) Multi-timescale long short-term memory neural network for modelling sentences and documents. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 2326–2335
- Liu P, Qiu X, Huang X (2016) Recurrent neural network for text classification with multi-task learning. In: IJCAI
- Liu C, Zhan Y, Li C, Du B, Wu J, Hu W, Liu T, Tao D (2022) Graph pooling for graph neural networks: Progress, challenges, and opportunities. arXiv preprint [arXiv:2204.07321](https://arxiv.org/abs/2204.07321)
- Liu X, Zheng Y, Du Z, Ding M, Qian Y, Yang Z, Tang J (2021) Gpt understands, too. [arXiv:2103.10385](https://arxiv.org/abs/2103.10385)
- Lu Z, Du P, Nie J-Y (2020) Vgcn-bert: augmenting bert with graph embedding for text classification. In: European Conference on Information Retrieval, pp. 369–382. Springer
- Maas A, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 142–150
- Malekzadeh M, Hajibabaei P, Heidari M, Zad S, Uzuner O, Jones JH (2021) Review of graph neural network in text classification. In: 2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), pp. 0084–0091. IEEE
- Manning CD, Surdeanu M, Bauer J, Finkel JR, Bethard S, McClosky D (2014) The stanford corenlp natural language processing toolkit. In: ACL (System Demonstrations), pp. 55–60. <http://dblp.uni-trier.de/db/conf/acl/acl2014-d.html#ManningSBFBM14>
- Marin A, Holenstein R, Sarikaya R, Ostendorf M (2014) Learning phrase patterns for text classification using a knowledge graph and unlabeled data. In: Fifteenth Annual Conference of the International Speech Communication Association
- Mariyam A, Basha SAH, Raju SV (2021) A literature survey on recurrent attention learning for text classification. In: IOP Conference Series: Materials Science and Engineering, 1042, 012030. IOP Publishing
- Matsuo Y, Sakaki T, Uchiyama K, Ishizuka M (2006) Graph-based word clustering using a web search engine. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, pp. 542–550
- Mei X, Cai X, Yang L, Wang N (2021) Graph transformer networks based text representation. *Neurocomputing* 463:91–100
- Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv preprint [arXiv:1301.3781](https://arxiv.org/abs/1301.3781)
- Minaee S, Kalchbrenner N, Cambria E, Nikzad N, Chenaghlu M, Gao J (2021) Deep learning-based text classification: a comprehensive review. *ACM Comput Surv* 54 (3):1–40
- Mironczuk MM, Protasiewicz J (2018) A recent overview of the state-of-the-art elements of text classification. *Exp Syst Appl* 106:36–54

- Mou L, Men R, Li G, Xu Y, Zhang L, Yan R, Jin Z (2015) Natural language inference by tree-based convolution and heuristic matching. arXiv preprint [arXiv:1512.08422](https://arxiv.org/abs/1512.08422)
- Niepert M, Ahmed M, Kutzkov K (2016) Learning convolutional neural networks for graphs. In: International Conference on Machine Learning, pp. 2014–2023. PMLR
- Nikolentzos G, Tixier A, Vazirgiannis M (2020) Message passing attention networks for document understanding. In: Proceedings of the AAAI Conference on Artificial Intelligence, 34, 8544–8551
- Ostendorff M, Bourgonje P, Berger M, Moreno-Schneider J, Rehm G, Gipp B (2019) Enriching bert with knowledge graph embeddings for document classification. arXiv preprint [arXiv:1909.08402](https://arxiv.org/abs/1909.08402)
- Page L, Brin S, Motwani R, Winograd T (1999) The pagerank citation ranking: bringing order to the web. Technical report, Stanford InfoLab
- Pang B, Lee L (2005) Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. arXiv preprint [cs/0506075](https://arxiv.org/abs/cs/0506075)
- Pasa L, Navarin N, Erb W, Sperduti A (2021) Simple graph convolutional networks <https://doi.org/10.48550/ARXIV.2106.05809>
- Patra A, Singh D (2013) A survey report on text classification with different term weighing methods and comparison between classification algorithms. Int J Comput Appl 75 (7):14–18
- Peng H, Li J, He Y, Liu Y, Bao M, Wang L, Song Y, Yang Q (2018) Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In: Proceedings of the 2018 World Wide Web Conference, pp. 1063–1072
- Pennington J, Socher R, Manning CD (2014) Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543
- Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 2227–2237. Association for Computational Linguistics, New Orleans, Louisiana. <https://doi.org/10.18653/v1/N18-1202>. <https://aclanthology.org/N18-1202>
- Piao Y, Lee S, Lee D, Kim S (2021) Sparse structure learning via graph neural networks for inductive document classification. arXiv preprint [arXiv:2112.06386](https://arxiv.org/abs/2112.06386)
- Pintas JT, Fernandes LA, Garcia ACB (2021) Feature selection methods for text classification: a systematic literature review. Artif Intell Rev 54 (8):6149–6200
- Radford A, Narasimhan K, Salimans T, Sutskever I, et al (2018) Improving language understanding by generative pre-training
- Ragesh R, Sellamanickam S, Iyer A, Bairi R, Lingam V (2021) Hetegcn: Heterogeneous graph convolutional networks for text classification. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, pp. 860–868
- Ren Y, Wang R, Ji D (2016) A topic-enhanced word embedding for twitter sentiment classification. Inf Sci 369:188–198
- Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2008) The graph neural network model. IEEE Trans Neural Netw 20 (1):61–80
- Schenker A, Last M, Bunke H, Kandel A (2004) Classification of web documents using graph matching. Int J Pattern Recognit Artif Intel 18 (03):475–496
- Selva Birunda S, Kanniga Devi R (2021) A review on word embedding techniques for text classification. Innov Data Commun Technol Appl. [https://doi.org/10.1007/978-981-15-9651-3\\_23](https://doi.org/10.1007/978-981-15-9651-3_23)
- Shah FP, Patel V (2016) A review on feature selection and feature extraction for text classification. In: 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSP-NET), pp. 2264–2268. IEEE
- Silva FB, Tabbone S, Torres RdS (2014) Bog: A new approach for graph matching. In: 2014 22nd International Conference on Pattern Recognition, pp. 82–87. IEEE
- Skarding J, Gabrys B, Musial K (2021) Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. IEEE Access 9:79143–79168
- Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng AY, Potts C (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1631–1642
- Szummer M, Jaakkola T (2001) Partially labeled classification with Markov random walks. Adv Neural Inf Proc Syst 14:838
- Tai KS, Socher R, Manning CD (2015) Improved semantic representations from tree-structured long short-term memory networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 1556–1566

- Tang H, Mi Y, Xue F, Cao Y (2020) An integration model based on graph convolutional network for text classification. *IEEE Access* 8:148865–148876
- Tang D, Qin B, Liu T (2015) Document modeling with gated recurrent neural network for sentiment classification. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1422–1432
- Thekumparampil KK, Wang C, Oh S, Li L-J (2018) Attention-based graph neural network for semi-supervised learning. arXiv preprint [arXiv:1803.03735](https://arxiv.org/abs/1803.03735)
- Thomas JM, Moallemi-Oureh A, Beddar-Wiesing S, Holzüter C (2022) Graph neural networks designed for different graph types: a survey. arXiv preprint [arXiv:2204.03080](https://arxiv.org/abs/2204.03080)
- Uryupina O, Plank B, Severyn A, Rotondi A, Moschitti A (2014) Sentube: A corpus for sentiment analysis on youtube social media. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), pp. 4244–4249
- Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. *Adv Neural Inf Proc Syst*. <https://doi.org/10.48550/arXiv.1706.03762>
- Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph Attention Networks. International Conference on Learning Representations
- Vijayan VK, Bindu K, Parameswaran L (2017) A comprehensive study of text classification algorithms. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1109–1113. IEEE
- Wang K, Han SC, Poon J (2022) Induct-gcn: Inductive graph convolutional networks for text classification. In: 2022 26th International Conference on Pattern Recognition (ICPR), pp. 1243–1249. IEEE
- Wang K, Han C, Long S, Poon J (2022) Me-gcn: Multi-dimensional edge-embedded graph convolutional networks for semi-supervised text classification. In: ICLR 2022 Workshop on Deep Learning on Graphs for Natural Language Processing
- Wang Z, Wang C, Zhang H, Duan Z, Zhou M, Chen B (2020) Learning dynamic hierarchical topic graph with graph convolutional network for document classification. In: International Conference on Artificial Intelligence and Statistics, pp. 3959–3969. PMLR
- Wang J, Zhang S, Xiao Y, Song R (2021) A review on graph neural network methods in financial applications. arXiv preprint [arXiv:2111.15367](https://arxiv.org/abs/2111.15367)
- Waseem Z (2016) Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In: Proceedings of the First Workshop on NLP and Computational Social Science, pp. 138–142
- Welling M, Kipf TN (2016) Semi-supervised classification with graph convolutional networks. In: J. International Conference on Learning Representations (ICLR 2017)
- Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY (2020) A comprehensive survey on graph neural networks. *IEEE Trans Neural Netw Learn Syst* 32 (1):4–24
- Wu L, Chen Y, Shen K, Guo X, Gao H, Li S, Pei J, Long B (2021) Graph neural networks for natural language processing: A survey. arXiv preprint [arXiv:2106.06090](https://arxiv.org/abs/2106.06090)
- Wu M, Pan S, Zhu X, Zhou C, Pan L (2019) Domain-adversarial graph neural networks for text classification. In: 2019 IEEE International Conference on Data Mining (ICDM), pp. 648–657. IEEE
- Wu F, Souza A, Zhang T, Fifty C, Yu T, Weinberger K (2019) Simplifying graph convolutional networks. In: International Conference on Machine Learning, pp. 6861–6871. PMLR
- Xie Y, Xu Z, Zhang J, Wang Z, Ji S (2022) Self-supervised learning of graph neural networks: a unified review. *IEEE Trans Pattern Anal Mach Intell* 45 (2):2412–2429
- Xie Q, Huang J, Du P, Peng M, Nie J-Y (2021) Inductive topic variational graph auto-encoder for text classification. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4218–4227
- Xing Z, Pei J, Keogh E (2010) A brief survey on sequence classification. *ACM Sigkdd Exp Newslett* 12 (1):40–48
- Yan X, Guo J, Lan Y, Cheng X (2013) A biterm topic model for short texts. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 1445–1456
- Yang Y, Wei Y, Shen T (2021) A review of graph neural networks for recommender applications. In: 2021 IEEE International Conference on Unmanned Systems (ICUS), pp. 602–607. IEEE
- Yang Z, Yang D, Dyer C, He X, Smola A, Hovy E (2016) Hierarchical attention networks for document classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1480–1489
- Yao L, Mao C, Luo Y (2019) Graph convolutional networks for text classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, 33, 7370–7377
- Ye Z, Jiang G, Liu Y, Li Z, Yuan J (2020) Document and word representations generated by graph convolutional network and Bert for short text classification. *ECAI 2020:2275–2281*

- Ying Z, You J, Morris C, Ren X, Hamilton W, Leskovec J (2018) Hierarchical graph representation learning with differentiable pooling. *Adv Neural Inf Proc Syst*. <https://doi.org/10.48550/arXiv.1806.08804>
- Yin Y, Jin Z (2015) Document sentiment classification based on the word embedding. In: 2015 4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering, pp. 456–461. Atlantis Press
- Zhang W, Yoshida T, Tang X (2011) A comparative study of tf\* idf, lsi and multi-words for text classification. *Exp Syst Appl* 38 (3):2758–2765
- Zhang X, Zhao J, LeCun Y (2015) Character-level convolutional networks for text classification. *Adv Neural Inf Proc Syst*. <https://doi.org/10.48550/arXiv.1509.01626>
- Zhang S, Tong H, Xu J, Maciejewski R (2019) Graph convolutional networks: a comprehensive review. *Comput Soc Netw* 6 (1):1–23
- Zhang X-M, Liang L, Liu L, Tang M-J (2021) Graph neural networks and their current applications in bioinformatics. *Front Genet* 12:690049
- Zhang J, Meng L (2019) Gresnet: Graph residual network for reviving deep gnns from suspended animation. *arXiv preprint arXiv:1909.05729*
- Zhang Y, Yu X, Cui Z, Wu S, Wen Z, Wang L (2020) Every document owns its structure: Inductive text classification via graph neural networks. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 334–339
- Zhang H, Zhang J (2020) Text graph transformer for document classification. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 8322–8327
- Zhou Y (2020) A review of text classification based on deep learning. In: Proceedings of the 2020 3rd International Conference on GeoInformatics and Data Analysis, pp. 132–136
- Zhou X, Li C (2005) Text classification by Markov random walks with reward. DMIN. Citeseer, Chicago, pp 275–278
- Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, Li C, Sun M (2020) Graph neural networks: a review of methods and applications. *AI Open* 1:57–81
- Zhou M, Cong Y, Chen B (2015) The poisson gamma belief network. In: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7–12, 2015, Montreal, Quebec, Canada, pp. 3043–3051
- Zhu H, Koniusz P (2020) Simple spectral graph convolution. In: International Conference on Learning Representations
- Zulqarnain M, Ghazali R, Hassim YMM, Rehan M (2020) A comparative review on deep learning models for text classification. *Indones J Electr Eng Comput Sci* 19 (1):325–335

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.