# Graph Neural Networks on Discriminative Graphs of Words

**Yassine Abbahaddou**
LIX, École Polytechnique
Institute Polytechnique de Paris, France
yassine.abbahaddou@polytechnique.edu

**Johannes F. Lutzeyer**
LIX, École Polytechnique
Institute Polytechnique de Paris, France
johannes.lutzeyer@polytechnique.edu

**Michalis Vazirgiannis**
LIX, École Polytechnique
Institute Polytechnique de Paris, France
mvazirg@lix.polytechnique.fr

## Abstract

In light of the recent success of Graph Neural Networks (GNNs) and their ability to perform inference on complex data structures, many studies apply GNNs to the task of text classification. In most previous methods, a heterogeneous graph, containing both word and document nodes, is constructed using the entire corpus and a GNN is used to classify document nodes. In this work, we explore a new Discriminative Graph of Words Graph Neural Network (DGoW-GNN) approach encapsulating both a novel discriminative graph construction and model to classify text. In our graph construction, containing only word nodes and no document nodes, we split the training corpus into disconnected subgraphs according to their labels and weight edges by the pointwise mutual information of the represented words. Our graph construction, for which we provide theoretical motivation, allows us to reformulate the task of text classification as the task of walk classification. We also propose a new model for the graph-based classification of text, which combines a GNN and a sequence model. We evaluate our approach on seven benchmark datasets and find that it is outperformed by several state-of-the-art baseline models. We analyse reasons for this performance difference and hypothesise under which conditions it is likely to change. Our code is publicly available at: https://github.com/abbahaddou/DGOW.

## 1 Introduction

Text classification is an important task in natural language processing. It has attracted an increasing amount of attention following the success of Deep Learning. There are many application of text classification including sentiment analysis, intent detection and spam filtering [22]. A key component of text classification is the representation of text. Recently, an increasing body of work surrounding text classification suggests to model text corpora as graphs [9, 11, 33]. The major benefit of graph representations is the ability to capture global information about the vocabulary, unlike sequential representations that are limited to local contextual information in sentences. Such approaches have two main components, the graph construction and the classification model. We will now introduce each of these in turn.

Various graph constructions have been proposed to model text. In most cases, words and documents are represented by nodes in a graph and edges are drawn based on different relationship metrics, which we discuss now. Word co-occurrence is among the most popular relationship metrics; all the

NeurIPS 2023 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2023).

terms that co-occur within a fixed-size sliding window are linked by edges [36, 40, 49]. Other works propose different weight computation for edges such as semantic [41], syntactic [1] and sequential relations [43, 24]. In this work, we will also use a fixed-size sliding window to draw edges between word nodes in our Discriminative Graph of Words (DGoW) and weight edges by the pointwise mutual information of the represented words. The main distinguishing criterion of our construction is that we also take training labels into consideration in our graph construction, by constructing one disconnected subgraph per class. This construction allows us to forgo the use of document nodes and to better separate the information of the different classes as we demonstrate both theoretically and empirically.

Once a graph representation of the corpus is obtained, there are several approaches to the text classification task. There exist, non-deep-learning approaches [40, 11] that first extract word and document embeddings from the graph structure, and then use machine learning algorithms, e. g., Support Vector Machines and Naive Bayes, to classify these embeddings. These methods outperform existing frequency-based criteria [25, 27], but suffer from some limitations, such as high-dimensionality, data sparsity, and lack of flexibility. These two-step approaches, separating the graph embedding and classification model, can be simplified by the use of Graph Neural Networks (GNNs), which simultaneously perform the two steps, as has been done in many recent studies [47, 50, 24]. Some of these methods [49, 47] operate in the *transductive* learning setting, where both training and test sentences are used to construct the training graph. While others [46, 42] work in the *inductive* learning setting, where only the training sentences are used in the graph construction. In this paper, we contribute the Discriminative Graph of Words Graph Neural Networks (DGow-GNN), in which we compose a GNN and a sequence model to simultaneously benefit from structural information of words in our DGoW construction and the order in which they arise.

Our contributions can be summarised as follows.

**1)** We propose a *new graph construction* by splitting the training corpus into disconnected subgraphs according to their labels and give theoretical motivation for our construction. This allows us to *reformulate the problem of text classification as a walk classification task*. Formally, we predict the probability that a sentence is represented as walk in the subgraph of each class.

**2)** We propose a *new model, the DGoW-GNN*, for graph-based representation of text which is a combination of a GNN and a sequence model.

**3)** We perform *extensive experimental validation* of our proposed graph construction and model on seven real-world benchmark datasets and observe that our DGoW-GNNs are outperformed by several state-of-the-art baseline models. We furthermore, analyse and explain these performance differences and hypothesise under which conditions they are likely to change.

## 2 Related Work

We now introduce GNNs and give an overview of graph-based approaches to text classification.

### 2.1 Graph Neural Networks

Graph Neural Networks (GNNs) are neural networks that operate on graph-structured data, which is defined to be the combination of a graph structure, denoted by $G = (V, E)$, where $V$ and $E$ denote the vertex and edge sets, respectively, and a node feature matrix $X \in \mathbb{R}^{|V| \times d}$, containing the node feature vector of node $v_i$ in its $i^{\text{th}}$ row. Like most deep learning approaches, GNNs are formed by stacking several computational layers, each of which produce a hidden representation for each node in the graph, denoted by $H^{(\ell)} = [h_v^{(\ell)}]_{v \in V}$. A GNN layer $\ell$ updates node representations relying mainly (or only) on the structure of the graph and the output of the previous layer $H^{(\ell-1)}$. Conventionally, the node features are used as input to the first layer $H^{(0)} = X$. The most popular framework of GNNs is that of Message Passing Neural Networks [10], where the computations are split into two main steps:

**Message-Passing**: Given a node $v$, this step applies a permutation-invariant function to its neighbours, denoted by $\mathcal{N}(v)$, to generate the aggregated representation,

$$m_v^{(\ell)} = \text{AGGREGATE}^{(\ell)}(\{h_u^{(\ell-1)}, u \in \mathcal{N}(v)\}).$$

**Update**: In this step, we combine the aggregated hidden states with the previous hidden representation of the central node $v$, usually by making use of a learnable function,

$$h_v^{(\ell)} = \text{UPDATE}^{(\ell)}(h_v^{(\ell-1)}, m_v^{(\ell)}).$$

Depending on the task, an additional readout or pooling function can be added after the last layer to aggregate the representation of nodes,

$$h_G = \text{READOUT}(H^{(L)}).$$

Graph Convolutional Networks [21] are among the most famous and commonly used GNN architectures. In GCNs, the graph convolutions are approximated by an order-one truncation of the expansion in terms of Chebyshev polynomials, which gives rise to a message-passing step in which weighted averages are taken over neighbourhoods in the graph. The weights in these weighted averages are fixed and depend on the square-rooted node degrees. A more general aggregation scheme is proposed in the more recent Graph Attention Networks [44], in which again a weighted average is used to combine information over graph neighbourhoods. The weights in these weighted averages are learned via a one hidden layer multi-layer perceptron taking both the central node's and neighbouring node's hidden states as input. Since the parameters of the attention mechanism in the GAT network were non-identifiable the Graph Attention Network V2 [4] was proposed to yield a better-functioning attention mechanism, in which the weight matrices are separated by a non-linearity. Alternative standard GNNs include the Graph Isomorphism Network [48], which sums hidden states over neighbourhoods, and the GraphSage model [13], which uses a learned aggregator, in which the hidden states of the central node and neighbouring nodes are concatenated, processed by a learnable weight matrix and then aggregated using one of several proposed aggregation schemes. While our proposed DGoW-GNN can be defined on the basis of any GNN, without loss of generality, we make use of the GCN, the most commonly used GNN, in our proposed architecture here.

### 2.2 GNNs For Text Classification

*TextGCN* [49] was the first work to apply a GCN to text classification. The authors construct a heterogeneous graph with both word and document nodes, draw weighted word-document edges using TF-IDF weights and weighted word-word edges using the point-wise mutual information (PMI). The input graph in *TextGCN* is constructed using both training and test documents, and a GCN is used to classify the document nodes.

Several works propose extensions of *TextGCN*. Improvements are either made on the graph construction or the GNN architecture. For example, the authors of *TensorGCN* [24] combine three heterogeneous graphs which only differ in their word-word edge weights. One graph, relies on PMI weights, as is done in the *TextGCN* approach, for the two other graph, semantic and syntactic based weights are used. The three graphs are fed to a GCN to perform text classification. Other models such as the *HeteGCN* [35] and *SGCN* [31] change the GNN architecture instead of the graph construction.

The methods discussed thus far in this section share one common problem: they are all transductive methods, i.e., the constructed graphs require both the training and the test documents. Thus, it is difficult to directly predict the labels of unseen documents. To deal with this issue, several works propose to work in the inductive learning setting. This is achieved in the *TextING* [50] and *MPAD* [29] models by constructing graphs on the sentence level, i.e., each sentence is represented as an individual graph. While the *TextING* and *MPAD* approach successfully implement a model capable of inductive learning, they have the drawback that each sentence graph only captures local information of the current sentence and global information, present in sentences not currently under consideration, is forgone. Subsequent inductive methods aimed to make use of the whole corpus by constructing training graphs from the entire training dataset. The *InducT-GCN* [46] for example generalises the *TextGCN* by constructing the heterogeneous graph $G$ using only training sentences and training a GCN on it. To predict the label of an unseen small batch, the method creates a new graph $G'$ using both the training sentences and the batch. Due to the small size of the batch, the two graph $G$ and $G'$ are similar, so the GCN trained on $G$ can reasonably be expected to generalise to $G'$. In another setting, *PTE* [42] train a GNN on a graph constructed with only training documents, word embeddings are then extracted using the hidden representation of nodes in the GNN. To classify a

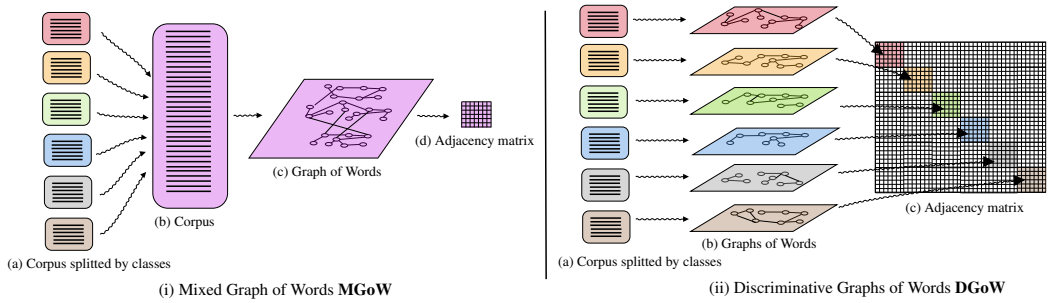


Figure 1: Illustration of the two configurations MGoW and DGoW for a toy example of 6 classes in the dataset. The coloured boxes represent different classes. In the *MGoW*, we merge all the sentence into one corpus and construct one graph of words. In *DGoW*, we keep the label based split, and create one disconnected subgraph per class.

new sentence, *PTE* [42] embed the sentence with the average of its words embeddings and then train small classifier (e.g, MLP) in an inductive setting.

Our novel graph construction and GNN-based model are applicable in the more realistic inductive learning settings, in which we do not have access to the test sentence structure during training.

## 3 Graph Construction

We begin by introducing our notation, formulating the problem of text classification and introducing necessary graph theoretical concepts. We then move on to present our proposed graph construction, the Discriminative Graph of Words (DGoW).

**Notation.** Each training or test sentence $s$ is a sequence of words $s = [w_1^{(s)}, w_2^{(s)}, \dots, w_{L(s)}^{(s)}]$, where $L(s)$ is the length of $s$. The number of words in a sentence depends on the data preprocessing used for the dataset. The label of a sentence belongs necessarily to a set of $P$ possible values.

**Problem Formulation.** First of all, let us introduce the task of text classification. Given a training corpus, i.e., a set of training sentences $\mathcal{S}^{train} = \{s_1^{train}, \dots, s_N^{train}\}$ and their corresponding labels $\mathcal{Y}^{train} = \{y_1^{train}, \dots, y_N^{train}\}$, the goal is to train a model to predict $\mathcal{Y}^{train}$ and generalise to the test corpus, i.e., the remaining unlabeled test sentences $\mathcal{S}^{test} = \{s_1^{test}, \dots, s_M^{test}\}$.

We now define the graph theoretic concepts of walks, connected components and disconnected subgraphs in graphs, which will be central to our proposed graph construction.

**Definition 3.1** (Walks, Connected Components and Disconnected Subgraphs)**.** A *walk* in a graph $G = (V, E)$ is a sequence of vertices, such that any two vertices adjacent in the sequence are connected by an edge in $G$. Then a subset $\mathcal{S} \in V$ is called a *connected component* in $G$ if there exists a walk between any two vertices $v_i, v_j \in \mathcal{S}$ and no walk exists from any $v_i \in \mathcal{S}$ to any $v_j \notin \mathcal{S}$. We further define *disconnected subgraphs* $\mathcal{C}_p$ for $p \in \{1, \dots, P\}$ in a graph $G$ to be graphs whose node and edge sets partition the node and edge set of $G$, respectively, such that there exists no walk from any $v_i \in \mathcal{C}_p$ to any $v_j \notin \mathcal{C}_p$ for $p \in \{1, \dots, P\}$.

We furthermore weight edges in our proposed graph construction by the PMI of words, which is calculated as follows,

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

where $p(i, j) = \frac{\#W(i,j)}{\#W}$, $p(i) = \frac{\#W(i)}{\#W}$, $\#W$ denotes the total number of fixed-sized windows (i.e., fixed-sized word spans in the text), $\#W(i)$ denotes the number of windows containing the word $i$ and $\#W(i, j)$ equals the number of windows containing both words $i$ and $j$ [28].

As discussed in Section 2, the standard approach in the literature [49, 24, 31] is to construct a graph on the basis of all sentences in the corpus, in which two words are linked based on different rules, which do not depend on the label of these sentences. These constructions typically contain both word and document nodes, where document nodes are linked to all words present in the represented

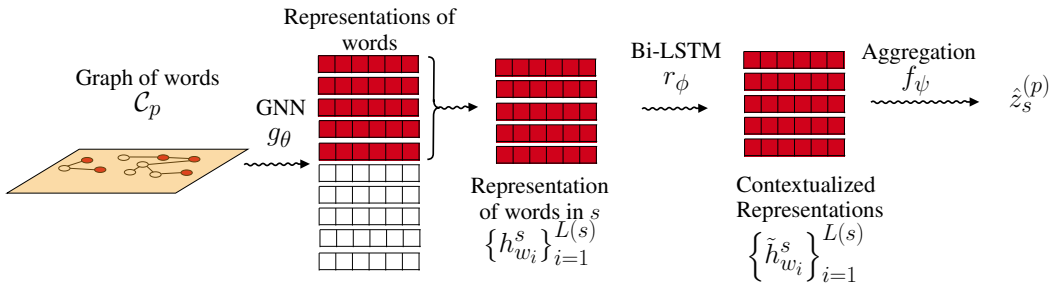


Figure 2: The architecture of *DGoW-GNN*. Our model takes as input the *Discriminative* graph of words $\mathcal{C}_p$, a class $p$ and a sentence $s$. The words occurring in $s$ are distinguished by the color red. As noticed, in the inductive setting, the sentence representation in $\mathcal{C}_p$ is not necessary a walk. We use the GNN $g_\theta$ to a vector representations of all the nodes in the graph. We select then only the vectors of the words occurring in $s$ and we fed it to a Bi-LSTM $r_\phi$ to contextualize the representations. At the end, we use a aggregation function $f_\psi$ to output a value in $[0, 1]$.

documents. In the following we will refer to such graph constructions as *Mixed Graphs of Words (MGoW)*. Since the text structure of a document is mainly encoded in the graph structure, representing the whole corpus in a single MGoW may reduce the capacity of GNNs to distinguish classes based on the structural characteristic of each class. We therefore propose an alternative graph construction.

**Definition 3.2** (Discriminiative Graph of Words)**.** In our Discriminiative Graphs of Words (DGoW) we construct one disconnected subgraph for each class in our training corpus. Edges in these disconnected subgraphs are drawn if words co-occur within a sliding window of size $\omega$ in the training corpus of the class corresponding to the given disconnected subgraph. Edges in our graph are weighted by the point-wise mutual information of the words represented by the connected word nodes.

Both the MGoW and DGoW constructions are illustrated in Figure 1. Contrary to most baselines, only word nodes are considered in DGoWs. Our proposed model for document classification, to be introduced in Section 4, will not rely on document nodes to classify documents.

In this work, we aim to stress the advantages of the DGoW construction over the MGoWs. Formally, we want to show that, relying on graph structure, the DGoW construction brings sentences from the same class closer and separate sentences from different classes. Conceptually, separating the classes in the corpus into disconnected subgraphs, with no edges between nodes in different classes, separates the classes better and therefore should aid in the task of discriminating these classes. This intuition is formalised in the following theoretical result on the spectral node embeddings of graphs containing several connected components.

**Theorem 3.3.** *[6] For a graph with $Q$ connected components the spectral node embeddings, produced by the normalised Laplacian eigenvectors corresponding to the smallest normalised Laplacian eigenvalue, are indicator vectors establishing the connected component membership of vertices.*

We refer the reader to [45, Proposition 4] for the formal statement and proof of Theorem 3.3. Note that the disconnected subgraphs that correspond to the different classes may contain several connected components as there may exist training sentences that share no single word with the remaining sentences in their class. However, the sum of all eigenvectors indicating the connected components in a given disconnected subgraph is itself an eigenvector and hence, the eigenspace of the smallest normalised Laplacian eigenvalue can not only inidicate the connected components of a DGoW, but also the disconnected subgraphs it contains. Therefore, the spectral embeddings of nodes in different connected components will indicate the class membership of these nodes. In Section 5.1 we will experimentally validate this theoretical insight for two different node embedding methods and clearly demonstrate that the classes are better separated in a DGoW than they are in a MGoW.

## 4 Proposed Method

We now introduce our text classification model that takes DGoWs, construced as outlined in Section 3 as input. To take advantage of the structural differentiation in DGoW, we perform the classification of a sentence $s$ by evaluating the probability of it arising as a *walk* of word nodes in each of the different disconnected subgraphs of our DGoW. This allows us to explicitly take the potentially different text structures of classes into account and thereby reformulate the task of text classification to correspond to the task of *walk classification* in our DGoW. Thus, we train a neural network model $\mathcal{M}_\Theta$ to predict the probability that a sentence belongs to each disconnected subgraph. The predicted label of a sentence $s$ then corresponds to the class with the highest probability.

$$\hat{y}_s = \underset{q \in \{1,\ldots,P\}}{\arg\max}\ \mathcal{M}_\Theta\left(q|s, \{\mathcal{C}_p\}_{p=1}^{P}, \mathcal{S}^{train}\right),$$

where $\mathcal{C}_p$ denotes the disconnected subgraph containing the sentence structure of class $p$, $s$ is the input sentence and $\mathcal{S}^{train}$ is the set of sentences in the training set used in the graph construction, which does not necessarily include $s$ since we are working in the inductive learning setting.

Our model $\mathcal{M}_\Theta$ is illustrated in Figure 2. It consists of three parts: the first part consist of a GNN $g_\theta$ to encode the nodes in graphs, the second part is a sequence model $r_\phi$ to capture the local context of words and the third part is an aggregation function $f_\psi$ to map the output into the desired format. This combination allows us to simultaneously capture the global contextual information of a word beyond the sentence currently under consideration with the GNN, as well as, the local information of the ordered words in their sentences. Below, we give further detail about each part of our model.

**Part 1 : Graph Neural Network.** Given a sentence $s = [w_1^{(s)}, w_2^{(s)}, \ldots, w_{L(s)}^{(s)}]$ belonging to a class $p$, we first encode the corresponding disconnected subgraph $\mathcal{C}_p$ with a GNN $g_\theta$. The goal of this GNN is to refine the embedding of each word in the class $p$. Formally, we start by selecting the disconnected subgraph $\mathcal{C}_p$ corresponding to the class $p$, we then feed it to $g_\theta$ and deduce the class dependent embedding of the words $\{w_i^{(s)}\}_{i=1}^{L(s)}$. To obtain the embedding of a word in all classes simultaneously, we can feed the entire DGoW to the GNN, where the adjacency matrix is block diagonal as illustrated in Figure 1. We use $h_w^{(p)}$ to denote the GNN output for word $w$ in a chosen class $p$.

$$\left[h_{w_1^{(s)}}^{(p)}, h_{w_2^{(s)}}^{(p)}, \ldots, h_{w_{L(s)}^{(s)}}^{(p)}\right] = g_\theta\left(s|\{\mathcal{C}_p\}_{p=1}^{P}\right).$$

Without loss of generality, we use the GCN architecture as the GNN $g_\theta$ in our proposed model.

**Part 2 : Sequence Model.** We select the GNN embeddings of only the words occurring in the sentences $\{h_{w_i^{(s)}}^{(p)}\}_{i=1}^{L(s)}$. We feed the sequence of words into a sequence model $r_\phi$ , e. g., Bi-LSTM [15], to have contextualised embeddings and explicity benefit from the information contained in the ordering of the words in a sentence.

$$\left[\tilde{h}_{w_1^{(s)}}^{(p)}, \ldots, \tilde{h}_{w_{L(s)}^{(s)}}^{(p)}\right] = r_\phi\left(\left[h_{w_1^{(s)}}^{(p)}, \ldots, h_{w_{L(s)}^{(s)}}^{(p)}\right]\right).$$

**Part 3 : Aggregator.** Now that the embedding of each word depends on both the class structure and the context of the sentence, we aggregate the embedding using a function $f_\psi$. In our case, we simply average each output of the Bi-LSTM and feed the new representation to a Multi-Layer Perceptron (MLP) to produce a predicted probability value $\hat{z}_s^{(p)} \in [0, 1]$ for each class $p = 1, \ldots, P$.

$$\hat{z}_s^{(p)} = f_\psi\left(\left[\tilde{h}_{w_1^{(s)}}^{(p)}, \ldots, \tilde{h}_{w_{L(s)}^{(s)}}^{(p)}\right]\right).$$

We next predict sentences to belong to the class with the highest predicted probability, i.e.,

$$\hat{y}_s = \underset{q \in \{1,\ldots,P\}}{\arg\max}\ \hat{z}_s^{(p)}.$$

To train our model to perform these individual binary predictions of how likely a given sentence is to arise in a given class $p$ we have to make use of negative samples during training. We use the term

Table 1: Structural similarity, measured via spectral node embeddings, between different pairs of labels for the R8 dataset.

| | **Labels** | $\omega = 2$ | $\omega = 5$ | $\omega = 10$ | $\omega = 15$ | $\omega = 20$ |
|---|---|---|---|---|---|---|
| MGoW | *earn/earn* | 84.68 | 70.18 | 67.23 | 65.29 | 66.96 |
| | *acq/acq* | 87.93 | 68.57 | 62.47 | 58.90 | 57.97 |
| | *earn/acq* | 80.67 | 36.78 | 26.74 | 25.75 | 26.18 |
| DGoW | *earn/earn* | 90.26 | 80.90 | 71.36 | 72.46 | 72.06 |
| | *acq/acq* | 99.58 | 89.03 | 87.75 | 84.21 | 83.19 |
| | *earn/acq* | 0 | 0 | 0 | 0 | 0 |

negative samples for a class $p$ to describe sentences, which are sampled from the corpus excluding $p$. It will be the task of our model to predict that these negative samples do not belong to the currently considered class $p$. Specifically, in our training procedure each sentence is considered twice. Firstly, we look at the sentence in its corresponding class, i. e., $y_i = p$, in this case, we train $\mathcal{M}_\Theta$ to predict the value of 1 for $\hat{z}_s^{(p)}$. Secondly to obtain negative samples, we randomly select a class $q$ different form the class label, i. e., $y_i \neq q$, and we train $\mathcal{M}_\Theta$ to predict the value of 0 for $\hat{z}_s^{(q)}$. Since, we perform the task of binary classification, we use the Binary Cross-Entropy loss.

# 5 Experiments and Results

In this section, we present the benchmark datasets used for our experiments, the selected baselines and the process used for the training and evaluation. We furthermore present experiments in which we observe the DGoW construction to lead to better class separation than a MGoW in Section 5.1. Finally we present the results of our DGoW-GNN and our baseline models on seven real-world benchmark datasets in Section 5.2, as well as an analysis explaining the performance of our model together with several ablation studies highlighting the impact of different model choices in our DGoW-GNN architecture in Section 5.3.

We provide information about implementation details of our experiments, including optimal hyper-parameters, in Appendix A. All source code is publicly available on GitHub [1].

**Datasets.** For a fair comparison, we use five datasets used throughout the graph-based text classification literature [49, 24, 50, 34, 7]. In particular we run experiments on the *Reuters 8 (R8)* and *Reuters 52 (R52)* [2], *Ohsumed (OH)* [14], *Movie Review (MR)* [30] and *20 Newsgroups (20NG)* [23] datasets. In addition, we include the *BBC News (BBC)* [12] and *Internet Movie Database (IMDb)* [26] datasets to test our model on long documents as well as very large datasets. In Appendix B we provide further details on these datasets and their summary statistics, as well as further information on our data pre-processing.

**Baselines.** Since our DGoW-GNNs apply in the inductive learning setting, we benchmark the performance of our model against the state-of-the art graph-based inductive methods. In particular, we consider the *InductTGCN* [46], *TextING* [50] and *HyperGAT* [7] for experimental comparison.

## 5.1 Structural Embedding Experiments

We now present a set of experiments which aims to measure the structural separateness of sentences in different classes in both the MGoW and DGoW constructions. To do so, we obtain node, i.e., word, embeddings in the two graph constructions using either the *spectral embeddings* obtained from the symmetrically normalised graph Laplacian [6, 3] or the *FastGAE* model [37]. We then represent sentences as the sum of their respective word embeddings. To measure the structural similiarity of classes in the graph we make use of the cosine similarity of sentence embeddings,

$$\delta(p, q, h) = \frac{1}{|\mathcal{C}_p||\mathcal{C}_q|} \sum_{i \in \mathcal{C}_p} \sum_{j \in \mathcal{C}_q;\ i \neq j} \frac{h_i^\top h_j}{\|h_i\| \|h_j\|},$$

[1] Code available at https://github.com/abbahaddou/DGOW

Table 2: Results of different models on the benchmark datasets; ① Inductive approaches ② Sequence models.

| | Model | Reference | R8 | R52 | OH | MR | 20NG | BBC | IMDB |
|---|---|---|---|---|---|---|---|---|---|
| ① | InductTGCN | [46] | 96.60 (0.17) | 93.18 (0.23) | 66.23 (0.48) | 75.75 (0.50) | 90.64 (0.21) | 96.36 (0.18) | 86.36 (0.29) |
| | TextING | [50] | 97.19 (0.30) | 94.24 (0.30) | **69.55 (0.45)** | 79.40 (0.44) | - | 97.54 (0.14) | - |
| | HyperGAT | [7] | 96.53 (0.25) | 92.75 (0.28) | 62.64 (0.61) | 76.76 (0.31) | **91.34 (0.17)** | 96.56 (0.33) | 86.25 (0/12) |
| | DGoW-GNN w/o Bi-LSTM | Ours | 94.11 (1.12) | 82.98 (1.19) | 30.01 (2.30) | 67.89 (0.41) | 71.00 (1.75) | 89.64 (0.53) | 68.66 (0.56) |
| | DGoW-GNN | Ours | 95.17 (0.22) | 86.26 (1.54) | 44.59 (1.01) | 71.65 (0.39) | 79.21 (1.29) | 92.14 (6.32) | 76.76 (1.90) |
| ② | Bi-LSTM | | 94.41 (0.60) | 86.09 (2.15) | 36.90 (1.11) | 72.36 (0.86) | 71.24 (1.25) | 86.96 (3.05) | 86.38 (0.37) |
| | BERT | [17] | 97.78 (0.20) | 93.21 (0.21) | 61.86 (0.72) | 85.30 (0.31) | 86.22 (0.31) | 97.77 (0.33) | 70.53 (0.72) |
| | RGCN-BERT | Ours | **98.01 (0.61)** | **93.58 (0.57)** | 62.41 (0.81) | **86.13 (0.54)** | 87.43 (0.67) | **98.01 (0.41)** | **87.02 (0.64)** |

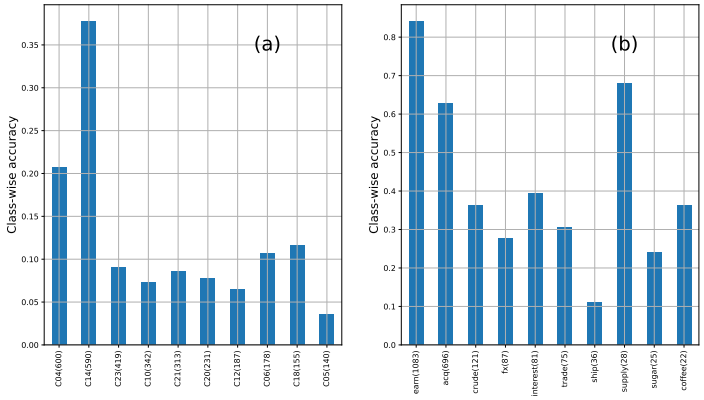


Figure 3: Class-wise accuracy of the DGoW-GNN on the OH (a) and R8 (b) datasets.

where $h_i$ and $h_j$ denote sentence embeddings and $\|\cdot\|$ is the $L_2$ norm of vectors. When $p \neq q$, $\delta$ measures the intra-similarity between the two classes, i. e., to which degree the subgraphs, representing two different sentences of labels $p$ and $q$, are structurally similar. On the other hand, when $p = q$, $\delta$ measures the inter-similarity between sentence embeddings within the same class. In this experiment we only consider classes with more than 400 sentences to get reasonably stable estimates. After constructing the graph on the basis of the whole corpus, we furthermore, randomly sample 400 sentences in each class to compute the two similarity metrics on the basis of these.

We report the result for the R8 dataset in Table 1. We notice the structural similarity of sentences in the MGoW configuration to be very high even when their two classes are different. On the other hand, the similarity between different classes in the DGoW configuration is almost null for different classes. In other words, while the structural inter-similarity is high in both configurations, the correlation between the graph representations of two sentences from two different classes is almost null in DGoW. We can benefit from the structural differences in DGoW to easily distinguish the classes. In Appendix C we show results obtained on the OH dataset and also for the *FastGAE* [37] embedding method run on both the R8 and OH datasets. These additional experiments all support the conclusions drawn on the basis of Table 1.

Another conclusion we can draw concerns the effect of the window size in Table 1. As the window size increases, the similarities decrease. So, increasing the window size helps differentiate different classes, but has the counter-effect on sentences belonging to the same class. Since, the intra-similarity is almost constant in the DGoW configuration, the best window size is $\omega = 2$.

### 5.2 Results of DGoW-GNN

We now analyse the results of *DGoW-GNN* in the inductive learning setting. We compare to graph-based approaches as well as the Bi-LSTM, a finetuned BERT and a combination of the BERT and RGCN model [38]. We report all the results in Table 2.

We observe that our *DGoW-GNN* is outperformed by the graph-based baseline models. We believe this performance difference to arise as a result of insufficient context being accessible within

our graph construction in the perfectly separated disconnected subgraphs. Indeed, we verify this hypothesis for the R8 and OH datasets in Figure 3, where we are clearly able to observe that the performance of our *DGoW-GNN* on sentences in the different classes correlates with the number of training sentences in these classes. We therefore hypothesise that our DGoW graph construction and *DGoW-GNN* has the potential to outperform the state-of-the-art baselines on larger datasets, in which we have more training sentences per class to add sufficient context to each word node.

We furthermore notice that our *DGoW-GNN* consistently outperforms the *Bi-LSTM*, which highlights the positive contribution of the graph construction to the model performance. We observe the graph-based models, including the *DGoW-GNN*, to outperform BERT on the IMDB and 20NG datasets, since the graph-based approaches are better adapted to long documents than BERT which, due to the high complexity, uses truncation and thus loses crucial information from the documents. The superiority of BERT on the remaining datasets can be explained by the power of pretrained models on short sentences [16].

We investigate the RGCN-BERT model, in which we work with a MGoW, where edges are typed according to the training class from which they originate. This allows us to make use of the RGCN [38], in which we aggregate and update over the different types of edges separately and then aggregate the type-wise representation. The node embeddings from the RGCN are then concatenated with the corresponding word embeddings from the BERT model to be fed to final classifier. We clearly observe that this heterogeneous graph construction falling in between the MGoW and DGoW construction in conjunction with the BERT embeddings outperforms all other baselines.

### 5.3 Ablation Studies

For the graph construction, we obtain the best results with a window sizes $\omega = 2$ as observed in Appendix E.1. As in previous work [19, 18], smaller window sizes help produce syntactic representations of words and also capture relations other than co-occurrence, e. g., dependency relations.

We further study the effect of different word embeddings as node features in our DGoW by training our *DGoW-GNN* also on DGoWs with the pre-trained GLoVe embeddings as node features [32]. We provide the experimental results in Appendix E.2. In most datasets, we obtain better results using the one-hot encodings. As noticed by [8], *pretrained embeddings, e. g., GloVe, can have a detrimental effect on model performance.* Therefore, we use one-hot encodings of the represented words as node features in our DGoW for our *DGoW-GNN* and all baseline models.

To study the importance of combining a GCN and a Bi-LSTM in *DGoW-GNN*, we train only the GCN and the aggregation function, and we omit the Bi-LSTM. We refer to this model as *DGoW w/o Bi-LSTM*. We used the same graph construction and the same training setup. We also trained separately a Bi-LSTM on the multi-label classification task. We report the comparison results in Table 2. We notice an improvement in the classification accuracy when mixing the GCN and the Bi-LSTM in our DGoW-GNN model.

We also test two different aggregation functions (*MLP* and PROD), in addition to the average aggregation function AVG. These aggregation functions are described in Appendix E.4 together with the result from this ablation study. We clearly observe the best results using the AVG aggregator.

The extensive ablation studies in this section show that we present a well-optimised and sufficiently explored approach in our DGoW-GNNs.

## 6 Conclusion

In this work, we present a new graph construction, the DGoW, for the task of text classification. We show both theoretically and in practice that our DGoW better separates the classes to be recovered in text classification. We also propose a new graph-based model *DGoW-GNN*, which is a combination of a GNN, a sequence model, and an aggregation function. Our experiments demonstrate that *DGoW-GNN* is outperformed by state-of-the-art graph-based approaches for text classification in the inductive learning setting. While our *DGoW-GNN* does not outperform the existing state-of-the art baselines, we believe it to be a well-motivated idea, which furthers our understanding of the graph-based approach to text classification and has the potential to lead to performance improvements at large scale.

## Acknowledgements

This work was partially supported by ANR via the AML-HELAS (ANR-19- CHIA-0020) project. The computation (on GPUs) was performed using HPC resources from GENCI-IDRIS (Grant 2023-AD010613410R1).

## References

[1] Ragheb Al-Ghezi and Mikko Kurimo. Graph-based syntactic word embeddings. *Proceedings of the Graph-based Methods for Natural Language Processing (TextGraphs)*, 2020.

[2] Chidanand Apté, Fred Damerau, and Sholom M. Weiss. Automated learning of decision rules for text categorization. *ACM Trans. Inf. Syst.*, 12(3):233–251, jul 1994.

[3] Thomas Bonald, Alexandre Hollocou, and Marc Lelarge. Weighted spectral embedding of graphs. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 494–501. IEEE, 2018.

[4] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

[5] Yahui Chen. Convolutional neural network for sentence classification. Master's thesis, University of Waterloo, 2015.

[6] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[7] Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. Be more with less: Hypergraph attention networks for inductive text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4927–4936, Online, November 2020. Association for Computational Linguistics.

[8] Lukas Galke and Ansgar Scherp. Bag-of-words vs. graph vs. sequence in text classification: Questioning the necessity of text-graphs and the surprising strength of a wide mlp. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4038–4051, 2022.

[9] Michael Gamon. Graph-based text representation for novelty detection. In *Proceedings of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing*, pages 17–24, New York City, June 2006. Association for Computational Linguistics.

[10] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning (ICML)*, pages 1263–1272. PMLR, 2017.

[11] Tuba Gokhan, Phillip Smith, and Mark Lee. GUSUM: Graph-based unsupervised summarization using sentence features scoring and sentence-BERT. In *Proceedings of TextGraphs-16: Graph-based Methods for Natural Language Processing*, pages 44–53, Gyeongju, Republic of Korea, October 2022. Association for Computational Linguistics.

[12] Derek Greene and Pádraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press, 2006.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[14] William Hersh, Chris Buckley, TJ Leone, and David Hickam. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *SIGIR'94*, pages 192–201. Springer, 1994.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[16] Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. BERT for coreference resolution: Baselines and analysis. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5803–5808, Hong Kong, China, November 2019. Association for Computational Linguistics.

[17] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2, 2019.

[18] Vlado Keselj. Book review: Speech and language processing by Daniel Jurafsky and James H. Martin. *Computational Linguistics*, 35(3), 2009.

[19] Douwe Kiela and Stephen Clark. A systematic study of semantic vector space model parameters. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 21–30, 2014.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[21] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[22] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

[23] Ken Lang. Newsweeder: Learning to filter netnews. In Armand Prieditis and Stuart Russell, editors, *Machine Learning Proceedings 1995*, pages 331–339. Morgan Kaufmann, San Francisco (CA), 1995.

[24] Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. Tensor graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8409–8416, 2020.

[25] H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317, 1957.

[26] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[27] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI Conference on Artificial Intelligence*, 1998.

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[29] Giannis Nikolentzos, Antoine Tixier, and Michalis Vazirgiannis. Message passing attention networks for document understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8544–8551, 2020.

[30] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[31] Luca Pasa, Nicoló Navarin, Wolfgang Erb, and Alessandro Sperduti. Simple graph convolutional networks. *arXiv: abs/2106.05809*, 2021.

[32] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[33] Duy Phung, Tuan Ngo Nguyen, and Thien Huu Nguyen. Hierarchical graph convolutional networks for jointly resolving cross-document coreference of entity and event mentions. In *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*, pages 32–41, Mexico City, Mexico, June 2021. Association for Computational Linguistics.

[34] Yinhua Piao, Sangseon Lee, Dohoon Lee, and Sun Kim. Sparse structure learning via graph neural networks for inductive document classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11165–11173, 2022.

[35] Rahul Ragesh, Sundararajan Sellamanickam, Arun Iyer, Ramakrishna Bairi, and Vijay Lingam. Hetegcn: heterogeneous graph convolutional networks for text classification. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 860–868, 2021.

[36] François Rousseau and Michalis Vazirgiannis. Graph-of-word and tw-idf: new approach to ad hoc ir. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 59–68, 2013.

[37] Guillaume Salha, Romain Hennequin, Jean-Baptiste Remy, Manuel Moussallam, and Michalis Vazirgiannis. Fastgae: Scalable graph autoencoders with stochastic subgraph decoding. *Neural Netw.*, 142(C):1–19, oct 2021.

[38] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer, 2018.

[39] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[40] Konstantinos Skianis, Fragkiskos Malliaros, and Michalis Vazirgiannis. Fusing document, collection and label graph-based representations with word embeddings for text classification. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, pages 49–58, New Orleans, Louisiana, USA, June 2018. Association for Computational Linguistics.

[41] Mark Steyvers and Joshua B. Tenenbaum. The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive science*, 29 1:41–78, 2005.

[42] Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1165–1174, 2015.

[43] Michalis Vazirgiannis. Graph of words: Boosting text mining tasks with graphs. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, page 1181, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[44] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[45] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[46] Kunze Wang, Soyeon Caren Han, and Josiah Poon. Induct-gcn: Inductive graph convolutional networks for text classification. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 1243–1249. IEEE, 2022.

[47] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[48] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[49] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019.

[50] Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. 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*, pages 334–339, Online, July 2020. Association for Computational Linguistics.

# Supplementary Material: Graph Neural Networks on Discriminative Graphs of Words

## A Implementation Details

To generate the DGoW, we used a fixed-sized sliding window of size 2. We fed the adjacency matrix of the DGoW into a Graph Convolution Network (GCN). We used the identity matrix as node features for the GCN. The number of GCN layers depends on the dataset, e. g. we use 2 layers for MR, 20NG and IMDB, 3 layers for R8, R52 and OH and 4 layers for BBC. For the aggregator, we use a Multi-Layer Perceptron with one hidden layer of dimension 128 and a scalar output. We use the ReLU activation function after the first layer and the sigmoid function after the second layer to output a value in the range 0 to 1.

To generate a random different label during training, we use *Frequency based negative sampling*, i. e., we sample labels according to their frequency in training. More precisely, given a training sentence from the class $\mathcal{C}_p$, we randomly sample a new label using the weights $\frac{|\mathcal{C}_q|}{\sum_s |\mathcal{C}_s|}$ for $q \neq p$.

We train our model on the seven datasets in the inductive settings using the Adam optimiser with a learning of $10^{-3}$. We repeated the training 10 times to test the stability of the model. To avoid overfitting, we randomly consider 10% of training sentences as validation set. In the inductive setting, we do not include the validation sentences in the graph construction. We stops the training as soon as the validation accuracy becomes worse than the four previous values.

To generate the structural embedding of *FastGAE*, we use 2-Layer *GCN* as the GAE autoencoder and we choose an embedding size of 256. We train it on Binary cross-entropy reconstruction loss using Adam optimiser [20] with a learning rate of $10^{-2}$.

For the spectral embedding, we chose 16 as the dimension of our embeddings. Since our embedding corresponds to the eigenvectors of the normalized Laplacian matrix and these eigenvectors are indicator vectors establishing which vertex is an element of which connected component [45], so the dimension should be greater than the number of connected components in our DGoW.

We also include sequence models in our baselines. We choose to train a BI-LSTM and we also finetune a pre-trained BERT model on the multi-label classification task. We report the result of all these baselines with the results of our model *DGNN* in Table 2.

To produce the baseline results, we used the code provided by the authors of *InductTGCN*[2], *TextING*[3] and *HyperGAT*[4]. We made sure that all code used the same data processing which will be described in Appendix C. We trained the baselines 10 times to compare the stability of each model. We were not able to produce the result of *TextING* for the 20NG dataset due to the high memory consumption of the model for high datasets, even by using RTX A6000 GPU. 20NG was also the only dataset in the original paper where the authors didn't report the results. For the model *TextING*, we didn't find any implementations, so we directly took the results from the original paper.

## B Further Information on The Considered Datasets

For a fair comparison, we used five datasets used in the graph-based text classification papers [49, 24, 50, 34, 7]. Below, we give a short description on the used datasets.

**Reuters 8 (R8)** and **Reuters 52 (R52)** [2]: two datasets collected from the Reuters financial newswire service. The classes represent topics such as business, sports, science, and technology.

[2]https://github.com/usydnlp/inducttgcn

[3]https://github.com/CRIPAC-DIG/TextING

[4]https://github.com/kaize0409/HyperGAT_TextClassification

Table 3: Basic statistics of the benchmark datasets.

| Dataset | # Docs | # Train | # Test | # Class | # Vocabulary | Avg. Length |
|---|---|---|---|---|---|---|
| R8 | 7,674 | 5,485 | 2,189 | 8 | 12,150 | 67 |
| R52 | 9,100 | 6,532 | 2,568 | 52 | 13,769 | 71 |
| OH | 7,400 | 3,357 | 4,043 | 23 | 12,258 | 133 |
| MR | 10,662 | 7,108 | 3,554 | 2 | 7,869 | 19 |
| 20NG | 18,846 | 11,314 | 7,532 | 20 | 40,852 | 220 |
| BBC | 2,225 | 1,225 | 1,000 | 5 | 14,380 | 217 |
| IMDB | 50,000 | 25,000 | 25,000 | 2 | 48,837 | 141 |

**Ohsumed (OH)** [14]: medical information database, consisting of titles and abstracts from medical journals.

**Movie Review (MR)** [30]: a binary sentiment classification dataset containing movie reviews.

**20 Newsgroups (20NG)** [23]: a dataset comprising around 18,000 newsgroup posts on 20 different newsgroups. The newsgroups cover a wide range of topics, including computers, politics, and sports.

In addition to these datasets, we include IMDB and BBC datasets to test our model on long documents as well as very large datasets.

**BBC News (BBC)** [12]: a dataset of news articles from the BBC news website, labeled with one of five categories: business, entertainment, politics, sport, and technology.

**Internet Movie Database (IMDB)** [26]: a dataset of 50,000 movie reviews labeled as either positive or negative. The dataset is balanced, meaning that there are an equal number of positive and negative reviews.

In Table 3, we present some basic statistics of the used datasets. More precisely, we present the total number of sentences, the number of training and test sentences, and number of classes. We also give the size of vocabulary and the average length of sentences in each dataset after the tokenisation. We notice the type of datasets is varied; we have datasets, such as IMDB, with a very large number of sentences, as well as very small datasets such as BBC. We also test datasets with varied vocabulary size and average length of documents.

We follow the same data processing used in baselines. Formally, except for the MR dataset, we remove non-alpha numerical characters, the leading and the trailing characters. All characters are converted to lowercase. And we finally split a sentence into words with a white-space character. In Table 3, we give supplementary information about the datasets. Additionally, we also remove all words that occur less than 2 times in the training data, and remove stop words (except for MR dataset as they improve the performance of models in the sentiment analysis task). There exist other tokenisation techniques in NLP [5, 39], but for consistency with previous work on graph-based approaches for text classification and, without loss of generality, we choose to adopt the same tokenisation as [46] for our method and for all used baselines.

## C Additional Structural Embedding Results

To support our DGoW configuration, we generated the structural embedding of words in both DGoW and MGoW configuration. Since we have multiple graphs in DGoW, we need to generate embeddings for each graph and it is important to generate all the embedding in the same vector space. To do so, we use the DGoW containing the several disconnected subgraphs illustrated in Figure 1 and do not consider disconnected subgraphs in isolation. As mentioned in Section 3, we used different node embedding methods : *Spectral Embeddings* [6, 3] and *FastGAE Embeddings* [37]. We generate the embeddings on two different dataset OH and R8. Since the number of nodes in the DGoW can be as large as $|\mathcal{V}| \times P$ where $P$ is the number of classes and $\mathcal{V}$ the vocabulary size, we keep only the 4 most frequent classes in OH when generating spectral embeddings as the methods is time-consuming with very high computational complexity. For a fair comparison for OH, we also keep the 4 most frequent classes in the MGoW configurations. For the embeddings of R8 and for the embeddings of OH when using *FastGAE*, we keep all the classes.

Table 4: Structural similarity between different pairs of labels for OH dataset① MGoW Configuration ② DGoW Configuration

| | **Labels** | $\omega = 2$ | $\omega = 5$ | $\omega = 10$ | $\omega = 15$ | $\omega = 20$ |
|---|---|---|---|---|---|---|
| ① | *C04/C04* | 64.08 | 54.58 | 51.88 | 51.01 | 50.63 |
| | *C10/C10* | 65.46 | 52.31 | 49.32 | 48.01 | 46.17 |
| | *C14/C14* | 67.35 | 56.61 | 51.60 | 48.69 | 50.24 |
| | *C23/C23* | 60.31 | 48.10 | 44.39 | 42.42 | 41.70 |
| | *C04/C10* | 59.80 | 44.79 | 41.04 | 39.22 | 38.26 |
| | *C04/C14* | 56.76 | 42.12 | 36.76 | 34.58 | 34.95 |
| | *C04/C23* | 58.93 | 45.55 | 41.65 | 39.78 | 39.37 |
| | *C10/C14* | 62.06 | 46.66 | 41.70 | 39.67 | 39.20 |
| | *C10/C23* | 61.31 | 47.55 | 43.85 | 42.09 | 40.88 |
| | *C14/C23* | 61.72 | 48.17 | 43.33 | 41.02 | 40.86 |
| ② | *C04/C04* | 83.95 | 73.89 | 71.44 | 71.49 | 71.26 |
| | *C10/C10* | 88.07 | 80.25 | 77.65 | 77.96 | 78.11 |
| | *C14/C14* | 82.41 | 71.11 | 69.61 | 67.89 | 67.86 |
| | *C23/C23* | 84.27 | 73.80 | 70.53 | 70.15 | 70.16 |
| | *C04/C10* | 0 | 0 | 0 | 0 | 0 |
| | *C04/C14* | 0 | 0 | 0 | 0 | 0 |
| | *C04/C23* | 0 | 0 | 0 | 0 | 0 |
| | *C10/C14* | 0 | 0 | 0 | 0 | 0 |
| | *C10/C23* | 0 | 0 | 0 | 0 | 0 |
| | *C14/C23* | 0 | 0 | 0 | 0 | 0 |

### C.1 Results Structural Similarity Using Spectral Embedding

In Table 4, we report the result of the experiment comparing the *intra-similarity* and *inter-similarity* of the spectral embedding for the dataset OH. As discussed in Section 5, the intra-similariy is smaller and the inter-similarity is higher in the *DGoW* graph construction. Therefore, we can better separate between text structures in different classes using the *DGoW* configuration.

### C.2 Structural Similarity Using FastGAE Embedding

Table 5: Structural similarity between different pairs of labels for R8 dataset ① Configuration MGoW ② Configuration DGoW

| | **Labels** | $\omega = 2$ | $\omega = 5$ | $\omega = 10$ | $\omega = 15$ | $\omega = 20$ |
|---|---|---|---|---|---|---|
| ① | *earn/earn* | 61.80 | 63.99 | 61.57 | 55.52 | 61.10 |
| | *acq/acq* | 66.70 | 83.16 | 87.22 | 87.36 | 9.89 |
| | *earn/acq* | -3.02 | -4.60 | -8.21 | -2.36 | -15.20 |
| ② | *earn/earn* | 99.81 | 98.91 | 99.93 | 99.99 | 99.87 |
| | *acq/acq* | 99.86 | 99.98 | 99.99 | 99.99 | 99.99 |
| | *earn/acq* | -38.07 | -32.61 | -34.70 | -64.44 | -32.89 |

To further support the spectral embedding results, we use the FastGAE model to generate the structural embeddings. We report the result of the similarity comparison for OH and R8 datasets in Tables 5 and 6. We notice the same trends as spectral embeddings. The intra-similarity is always high and positive to the extent that some sentences in some classes exhibit greater structural similarity to a different class than the sentences in the same class. The inter-similarity is much higher and almost equal to 1 in the DGoW configuration. The intra-similarity is very small in DGoW compared to MGoW values. Some values are usually negative, which indicate that the graph representations of nodes and sentences in different classes are strongly opposite vectors.

Table 6: Structural similarity between different pairs of labels for OH dataset ① Configuration MGoW ② Configuration DGoW.

| | **Labels** | $\omega = 2$ | $\omega = 5$ | $\omega = 10$ | $\omega = 15$ | $\omega = 20$ |
|---|---|---|---|---|---|---|
| ① | *C23/C23* | 47.44 | 57.35 | 37.23 | 38.00 | 37.62 |
| | *C10/C10* | 56.38 | 79.58 | 63.42 | 49.23 | 47.63 |
| | *C04/C04* | 47.56 | 72.69 | 51.70 | 39.83 | 43.31 |
| | *C14/C14* | 48.54 | 38.79 | 10.91 | 51.74 | 55.45 |
| | *C20/C20* | 50.56 | 66.71 | 44.35 | 41.90 | 42.47 |
| | *C21/C21* | 58.43 | 78.70 | 58.09 | 55.21 | 54.51 |
| | *C10/C23* | 49.51 | 67.53 | 48.66 | 41.62 | 41.02 |
| | *C04/C23* | 45.56 | 64.63 | 44.02 | 32.57 | 33.77 |
| | *C14/C23* | 47.90 | 47.22 | 20.10 | 42.17 | 41.0 |
| | *C20/C23* | 46.56 | 61.91 | 40.76 | 28.66 | 29.23 |
| | *C21/C23* | 49.51 | 67.12 | 46.62 | 40.31 | 40.37 |
| | *C04/C10* | 47.69 | 76.09 | 57.32 | 39.09 | 40.21 |
| | *C10/C14* | 49.06 | 55.32 | 25.92 | 42.50 | 40.51 |
| | *C10/C20* | 48.31 | 72.91 | 53.15 | 32.83 | 31.63 |
| | *C10/C21* | 55.06 | 79.19 | 60.79 | 49.75 | 48.56 |
| | *C04/C14* | 44.71 | 53.03 | 23.62 | 28.92 | 25.51 |
| | *C04/C20* | 43.34 | 69.71 | 48.00 | 38.96 | 39.37 |
| | *C04/C21* | 47.28 | 75.65 | 54.90 | 38.99 | 41.99 |
| | *C14/C20* | 47.10 | 50.78 | 21.77 | 23.93 | 21.26 |
| | *C14/C21* | 49.52 | 54.96 | 24.90 | 41.12 | 39.72 |
| | *C20/C21* | 43.86 | 72.49 | 50.88 | 29.81 | 30.05 |
| ② | *C23/C23* | 100.0 | 100.0 | 99.98 | 100.0 | 99.99 |
| | *C10/C10* | 100.0 | 100.0 | 99.95 | 100.0 | 99.99 |
| | *C04/C04* | 100.0 | 100.0 | 99.98 | 100.0 | 99.99 |
| | *C14/C14* | 100.0 | 100.0 | 99.98 | 100.0 | 99.99 |
| | *C20/C20* | 100.0 | 100.0 | 99.99 | 100.0 | 99.99 |
| | *C21/C21* | 100.0 | 100.0 | 99.99 | 100.0 | 99.99 |
| | *C10/C23* | -9.52 | -10.08 | -6.09 | -9.96 | -2.98 |
| | *C04/C23* | -13.04 | -15.41 | -13.61 | -12.22 | -10.2 |
| | *C14/C23* | -13.96 | -13.63 | -13.19 | -12.89 | -7.23 |
| | *C20/C23* | -12.58 | -9.32 | -6.7 | -7.93 | -11.44 |
| | *C21/C23* | -9.63 | -10.31 | -9.61 | -9.02 | -8.03 |
| | *C04/C10* | -11.53 | -9.97 | -7.37 | -9.9 | -3.45 |
| | *C10/C14* | -10.84 | -10.69 | -7.53 | -9.44 | -5.05 |
| | *C10/C20* | -10.17 | -7.72 | -2.39 | -7.5 | -7.29 |
| | *C10/C21* | -7.32 | -7.51 | -3.36 | -6.61 | 1.28 |
| | *C04/C14* | -13.76 | -13.78 | -15.0 | -12.93 | -12.42 |
| | *C04/C20* | -13.19 | -10.03 | -6.88 | -9.81 | -12.27 |
| | *C04/C21* | -9.82 | -10.93 | -9.13 | -10.56 | -8.55 |
| | *C14/C20* | -14.3 | -9.53 | -10.82 | -8.57 | -13.27 |
| | *C14/C21* | -10.87 | -11.64 | -11.88 | -9.37 | -2.21 |
| | *C20/C21* | -15.64 | -8.61 | -2.39 | -8.11 | -7.15 |

# D Examples of Predictions

In Table 7, we give the predictions of *DGoW-GNN* for some randomly selected test sentences of the R8 dataset.

# E Ablation Studies

In this appendix we provide extensive ablation study results.

Table 7: Five randomly sampled examples of *DGoW-GNN* predictions.

| Ground Truth | Test Sentence and Predictions |
|---|---|
| earn | **Sentence :** *entre computer centers inc etre nd qtr loss shr loss cts vs profit cts net loss vs profit revs mln vs mln st half shr loss cts vs profit cts net loss vs profit revs mln vs mln note current year net both periods includes dlr pretax provision for closing overseas operations and tax credits dlrs in quarter and dlrs in half reuter*<br><br>**Predictions :** acq: $9.68\ 10^{-9}$, crude: $4.76\ 10^{-9}$, **earn: $\mathbf{0.99}$**, grain: $1.97\ 10^{-8}$, interest: $7.40\ 10^{-9}$, money-fx: $1.45\ 10^{-8}$, ship: $1.16\ 10^{-08}$, trade: $8.33\ 10^{-9}$ |
| earn | **Sentence :** *weirton steel corp rd qtr net mln vs mln revs mln vs mln nine mths net mln vs mln revs mln vs mln note company does not report per share earnings as it is a privately owned concern net amounts reported are before taxes profit sharing and contribution to employee stock ownership trust reuter*<br><br>**Predictions :** acq: $6.56\ 10^{-8}$, crude: $2.44\ 10^{-8}$, **earn: $\mathbf{0.99}$**, grain: $1.18\ 10^{-8}$, interest: $1.08\ 10^{-8}$, money-fx: $1.19\ 10^{-8}$, ship: $1.64\ 10^{-8}$, trade: $6.27\ 10^{-9}$ |
| acq | **Sentence :** *lvi group lvi to make acquisition lvi group inc said it has agreed in principle to purchase all outstanding shares of spectrum holding corp for a proposed mln dlrs in cash lvi said an additional mln dlrs in common stock and seven mln dlrs in notes will become payable if spectrum has certain minimum future earnings lvi an interior construction firm said the acquisition is subject to execution of a definitive agreement and completion of due diligence lvi and spectrum an asbestos abatement concern expect to close the deal in june lvi said reuter*<br><br>**Predictions :** **acq: $\mathbf{5.55\ 10^{-2}}$**, crude: $3.53\ 10^{-8}$, earn: $1.26\ 10^{-4}$, grain: $2.76\ 10^{-8}$, interest: $1.28\ 10^{-8}$, money-fx: $9.17\ 10^{-9}$, ship: $1.24\ 10^{-7}$, trade: $8.46\ 10^{-8}$ |
| interest | **Sentence :** *average yen cd rates fall in latest week average interest rates on yen certificates of deposit cd fell to pct in the week ended april from pct the previous week the bank of japan said new rates previous in brackets average cd rates all banks pct money market certificate mmc ceiling rates for week starting from april pct average cd rates of city trust and long term banks less than days pct days pct average cd rates of city trust and long term banks days pct days pct days unquoted days pct over days pct unqtd average yen bankers acceptance rates of city trust and long term banks to less than days pct days pct days unquoted unqtd reuter*<br><br>**Predictions :** acq: $4.33\ 10^{-4}$, crude: $2.01\ 10^{-7}$, earn: $1.30\ 10^{-7}$, grain: $2.71\ 10^{-7}$, **interest: $\mathbf{0.99}$**, money-fx: $2.60\ 10^{-4}$, ship: $1.81\ 10^{-6}$, trade: $5.22\ 10^{-5}$ |
| trade | **Sentence :** *white house says japanese tarriffs likely the white house said high u s tariffs on japanese electronic goods would likely be imposed as scheduled on april despite an all out effort by japan to avoid them presidential spokesman marlin fitzwater made the remark one day before u s and japanese officials are to meet under the emergency provisions of a july semiconductor pact to discuss trade and the punitive tariffs fitzwater said i would say japan is applying the full court press they certainly are putting both feet forward in terms of explaining their position but he added that all indications are they the tariffs will take effect reuter*<br><br>**Predictions :** acq: $7.00\ 10^{-8}$, crude: $6.74\ 10^{-7}$, earn: $8.41\ 10^{-9}$, grain: $2.53\ 10^{-5}$, interest: $5.30\ 10^{-6}$, money-fx: $9.28\ 10^{-4}$, ship: $3.95\ 10^{-7}$, **trade: $\mathbf{0.99}$** |

### E.1 Window Size

In Table 8, we compare the performance of DGoW on the datasets R8, OH and MR when using different window sizes 2, 5, 10 and 15.

Table 8: Ablation study on the window size $\omega$.

| Model | R8 | OH | MR |
|---|---|---|---|
| DGoW w/ $\omega = 2$ | **95.17 (0.22)** | **44.59 (1.01)** | **71.65 (0.39)** |
| DGoW w/ $\omega = 5$ | 82.64 (0.19) | 43.33 (1.94) | 61.47 (0.40) |
| DGoW w/ $\omega = 10$ | 73.37 (0.17) | 41.21 (0.95) | 62.02 (0.66) |

We notice that the accuracy is decreasing when increasing the window size.

### E.2 Word Embedding

In Table 9, we compare the performance of DGoW on the datasets R8, OH and MR when using the one-hot embeddings and the pre-trained GLoVe embeddings of dimension 100.

Table 9: Ablation study on the word embedding

| **Model** | **R8** | **OH** | **MR** |
|---|---|---|---|
| DGoW w/ One-Hot | **95.17 (0.22)** | **44.59 (1.01)** | **71.65 (0.39)** |
| DGoW w/ GloVe | 80.49 (0.04) | 41.44 (0.84) | 62.32 (0.67) |

### E.3 GNN Model

In Table 10, we compare the performance of DGoW on the datasets R8, OH and MR when using different GNN architectures : GCN and GAT [44].

Table 10: Ablation study on the GNN architecture

| **Model** | **R8** | **OH** | **MR** |
|---|---|---|---|
| DGoW w/ GCN | **95.17 (0.22)** | **44.59 (1.01)** | **71.65 (0.39)** |
| DGoW w/ GAT | 79.84 (3.07) | 11.50 (1.20) | 62.43 (0.86) |

### E.4 Aggregation Function

Table 11: Ablation study on the aggregation function.

| **Aggregator** | **R8** | **R52** | **OH** | **MR** |
|---|---|---|---|---|
| DGoW w/ AVG | **95.17 (0.22)** | **86.26 (1.54)** | **44.59 (1.01)** | **71.65 (0.39)** |
| DGoW w/ MLP | 92.16 (1.38) | 82.08 (1.95) | 29.79 (5.13) | 70.78 (0.87) |
| DGoW w/ PROD | 90.22 (5.30) | 74.13 (1.50) | 25.01 (25.39) | 69.09 (0.58) |

We first tested the MLP aggregator that take the concatenation of the first and last element of the Bi-LSTM and feed it to an MLP. We also tested PROD function defined as follows

$$\prod_{(i,j)\in\mathcal{W}} \sigma\left(\tilde{h}_{w_i^{(s)}}^{(p)}{}^T \tilde{h}_{w_j^{(s)}}^{(p)}\right),$$

where is $\mathcal{W}$ is the set of all the windows of size $w$. By using the PROD function, we assume that a sentence belongs to a disconnected subgraph if only if every edge belongs to that disconnected subgraph. In Table 11, we observe the impact of the AVG, MLP and PROD aggregation function on the performance of our DGoW-GNN on all our considered datasets. As noticed, we obtain the best results using the AVG aggregator. It is also the aggregation with the smallest standard deviation.