ADDRESSING ENCODER-ONLY TRANSFORMER LIMITATIONS WITH GRAPH NEURAL NETWORKS FOR TEXT CLASSIFICATION

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

By

Arda Can Aras

January 2025

Addressing Encoder-Only Transformer Limitations with Graph Neural Networks for Text Classification
By Arda Can Aras
January 2025

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Aykut Koç (Advisor)

Tolga Çukur

Çağrı Toraman

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

ADDRESSING ENCODER-ONLY TRANSFORMER LIMITATIONS WITH GRAPH NEURAL NETWORKS FOR TEXT CLASSIFICATION

Arda Can Aras
M.S. in Electrical and Electronics Engineering
Advisor: Aykut Koç
January 2025

Recent advancements in natural language processing (NLP) have been primarily driven by transformer-based models, which capture contextual information within sequences, revolutionizing tasks such as text classification and natural language understanding. In parallel, graph neural networks (GNNs) have emerged as powerful tools for modeling structured data, leveraging graph representations to capture global relationships across entities. However, significant challenges persist at the intersection of these fields, limiting the efficacy and scalability of existing models. These challenges include the inability to seamlessly integrate contextual and structural information, computational inefficiencies associated with static graph construction and transductive learning, and the underperformance of models in low-labeled data scenarios. This thesis explores and addresses these challenges by developing novel methodologies that unify transformers and GNNs, leveraging their complementary strengths. The first contribution, Graph Receptive Transformer Encoder (GRTE), introduces an architecture that combines pre-trained transformer models with heterogeneous and homogeneous graph representations to enhance text classification in both inductive and transductive settings. Compared to state-of-the-art models, GRTE achieves significant computational efficiency, reducing training overhead by up to 100×. The second contribution, Relational Modeling for Heterogeneous Text Graphs (Text-RGNNs), proposes a relational modeling framework for heterogeneous text graphs, enabling the nuanced representation of diverse interactions between nodes and demonstrating substantial accuracy improvements of up to 10.61% over existing models, particularly in low-labeled data settings. Finally, the third contribution, Enhancing Transformer Encoders with Vector Visibility Graph Neural Networks (VISPool), introduces a scalable architecture that dynamically constructs vector visibility graphs from transformer outputs, enabling seamless integration of graph-based

reasoning into transformer pipelines while improving performance on NLP benchmarks such as General Language Understanding Evaluation (GLUE), with performance improvements of up to 13% in specific tasks. Through comprehensive experimentation and benchmarking against state-of-the-art models, this thesis establishes the efficacy of these proposed methodologies. The results demonstrate the potential for improved performance, scalability, and the ability to address long-standing challenges in NLP and GNN integration. These contributions lay a robust foundation for future research and applications at the intersection of graph-based and transformer-based approaches, advancing the state of the art in text representation and classification.

Keywords: natural language processing (NLP), transformer, graph neural networks (GNNs), text classification.

ÖZET

YALNIZCA KODLAYICI KULLANAN DÖNÜŞTÜRÜCÜLERİN METİN SINIFLANDIRMASINDAKİ SINIRLAMALARININ ÇİZGE SINIR AĞLARI İLE AŞILMASI

Arda Can Aras
Elektrik ve Elektronik Mühendisliği, Yüksek Lisans
Tez Danışmanı: Aykut Koç
Ocak 2025

Son yıllarda Doğal Dil İşleme (DDİ) alanındaki gelişmeler, sıralı verilerde bağlamsal bilgiyi yakalayabilen dönüştürücü tabanlı modeller tarafından yönlendirilmiş ve metin sınıflandırma ile doğal dil anlama gibi görevlerde devrim yaratmıştır. Paralel olarak, Çizge Sinir Ağları (GNNs), çizge temsillerini kullanarak varlıklar arasındaki küresel ilişkileri modelleyebilen güçlü araçlar olarak ortaya çıkmıştır. Ancak, bu alanların kesişiminde, mevcut modellerin etkinliğini ve ölçeklenebilirliğini sınırlayan önemli zorluklar devam etmektedir. Bu zorluklar arasında bağlamsal ve yapısal bilgiyi sorunsuz bir şekilde entegre edememe, statik grafik oluşturma ve transdüktif öğrenmeyle ilişkili hesaplama verimsizlikleri ve düşük etiketli veri senaryolarında modellerin yetersiz performansı yer almaktadır. Bu tez, dönüştürücü ve GNN alogirtmalarının birbirini tamamlayan güçlü yönlerinden yararlanarak bu zorlukları ele alan yenilikçi metodolojiler geliştirmektedir. İlk katkı, Graph Receptive Transformer Encoder (GRTE), önceden eğitilmiş dönüştürücü modelleri ile heterojen ve homojen grafik temsillerini birleştirerek metin sınıflandırmasını hem indüktif hem de transdüktif ortamlarda iyileştiren bir mimari sunmaktadır. GRTE, mevcut en gelişmiş modellerle karşılaştırıldığında, eğitim yükünü 100× oranında azaltarak önemli ölçüde hesaplama verimliliği sağlamaktadır. İkinci katkı, Relational Modeling for Heterogeneous Text Graphs (Text-RGNNs), heterojen metin çizgeler için ilişkisel bir modelleme çerçevesi önermekte ve düğümler arasındaki çeşitli etkileşimlerin incelikli bir şekilde temsil edilmesini sağlayarak, özellikle düşük etiketli veri senaryolarında mevcut modellere göre %10.61 oranına kadar doğruluk artışı göstermektedir. Son olarak, üçüncü katkı olan Enhancing Transformer Encoders with Vector Visibility Graph Neural Networks (VISPool),

dönüştürücü çıktılarından dinamik olarak vektör görünürlük çizgeleri oluşturan ölçeklenebilir bir mimari sunmakta ve çizge tabanlı akıl yürütmeyi dönüştürücü iş akışlarına sorunsuz bir şekilde entegre ederken General Language Understanding Evaluation (GLUE) gibi DDİ veri kümelerinde belirli görevlerde %13 oranına kadar performans iyileştirmeleri sağlamaktadır. Kapsamlı deneyler ve mevcut en gelişmiş modellerle yapılan karşılaştırmalı değerlendirmeler yoluyla, bu tez önerilen metodolojilerin etkinliğini ortaya koymaktadır. Sonuçlar, performans iyileştirme, ölçeklenebilirlik ve DDİ ile GNN entegrasyonundaki uzun süredir devam eden zorlukları ele alma potansiyelini göstermektedir. Bu tez, çizge tabanlı ve dönüştürücü tabanlı yaklaşımların kesişiminde gelecekteki araştırmalar ve uygulamalar için sağlam bir temel oluşturarak metin temsili ve sınıflandırmasında alanın en ileri düzeyine önemli bir katkı sağlamaktadır.

Anahtar sözcükler: Doğal Dil İşleme (DDİ), dönüştürücü, Çizge Sinir Ağları (GNNs), metin sınıflandırması.

Acknowledgement

This work is supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK), Directorate of Science Fellowships and Grant Programmes (BİDEB), through the 2210-A National MSc Scholarship Program and partially by the Directorate of Research Support Programs (ARDEB) through projects 1001-120E346 and 124E179.

This work is also supported by Türk Telekomünikasyon A.Ş. through the 5G and Beyond Joint Graduate Support Program coordinated by the Information and Communication Technologies Authority.

I want to express my deepest gratitude to my family, especially my parents, Nalan Aras and Coşkun Aras, for their unwavering love, guidance, and support. Their belief in me has been the cornerstone of my achievements. I thank all my family members for their constant encouragement throughout this journey. To my friends, I thank every single one of them for being my constant source of joy, laughter, and perspective, even during the most challenging times.

I am immensely grateful to my advisor, Dr. Aykut Koç, for his invaluable mentorship and insightful advice. Working under his guidance has been an enriching experience. I would also like to extend my heartfelt thanks to the members of our research group, especially to my co-authors, Tuna Alikaşiofğlu and Emirhan Koç, for the collaborative spirit and inspiring discussions that enriched this work.

Lastly, I owe my most profound appreciation to my wife, Nehir Aras. I thank her for always being by my side through every high and low and for her patience and understanding. Her tolerance of the long hours I dedicated to my work, her support during frustration, and her encouragement when I needed it most have been invaluable. She has given me the strength and motivation to keep pushing forward. I am forever grateful for all the love, kindness, and unwavering support.

Contents

List of Figures

List of Tables

Disclaimer

This thesis is a cumulative work based on a series of papers previously published by the author. While the individual papers stand independently, this thesis provides a cohesive narrative that integrates the findings and discussions from each work to present a comprehensive overview of the conducted research. Any overlapping content has been appropriately referenced. The contents presented herein include original research and findings that have been presented in the following publications:

- **A. C. Aras**, T. Alikaşifoğlu and A. Koç, "Graph Receptive Transformer Encoder for Text Classification," in *IEEE Transactions on Signal and Information Processing over Networks*, vol. 10, pp. 347-359, 2024, DOI: 10.1109/TSIPN.2024.3380362. [1]

- **A. C. Aras**, T. Alikaşifoğlu and A. Koç, "Text-RGNNs: Relational Modeling for Heterogeneous Text Graphs," in *IEEE Signal Processing Letters*, vol. 31, pp. 1955-1959, 2024, DOI: 10.1109/LSP.2024.3433568. [2]

- T. Alikaşifoğlu, **A. C. Aras**, and A. Koç, "VISPool: Enhancing transformer encoders with vector visibility graph neural networks," In Findings of the Association for Computational Linguistics: ACL 2024, pages 2547–2556, Bangkok, Thailand. Association for Computational Linguistics. DOI: 10.18653/v1/2024.findings-acl.149. [3]

Chapter 1

Introduction

This chapter briefly introduces the strengths and weaknesses of encoder-only transformers for text classification and how these weaknesses have been previously addressed by integrating graph neural networks (GNNs). On the other hand, existing works that rely on GNNs to overcome these weaknesses come with their challenges. We further discuss existing challenges in these approaches, summarize our motivation and contributions, and outline the remaining thesis organization.

1.1 Text Classification

Natural language processing (NLP) is a multidisciplinary field at the intersection of computer science, linguistics, and artificial intelligence, focused on enabling machines to understand, interpret, and generate human language. It encompasses a wide range of tasks, including language translation, speech recognition, information retrieval, semantic communications, text simplification, bias detection, named-entity recognition (NER) and word embeddings each contributing to various real-world applications [4–41]. Text classification is a critical branch of NLP in these diverse areas due to its significance in automating information organization and extraction.

Text classification is a fundamental task in NLP, with broad applications ranging from sentiment analysis and question answering to text summarization and segmentation [26, 42–45]. The goal is to assign predefined labels to textual units such as sentences, paragraphs, or entire documents [14, 15, 27, 36, 46, 47]. Given its importance, text classification has been extensively studied, leading to the development of highly advanced models.

Among these models, encoder-only transformers like BERT [4, 5, 48, 49] have emerged as dominant solutions, achieving state-of-the-art performance on a wide range of text classification benchmarks. These models excel at capturing local contextual semantics within a document by employing powerful self-attention mechanisms. However, despite their success, there remains significant room for improvement in two key aspects: capturing global contextual representations across documents and leveraging the vast amount of available unlabeled data.

While encoder-only transformers such as BERT are highly effective at understanding individual documents' internal structure and semantics, they are inherently limited to single-sequence contexts. During fine-tuning, they operate on individual documents in isolation, meaning they cannot directly capture inter-document relationships, shared patterns, or broader contextual dependencies across multiple documents. Modeling such global relationships is crucial in many real-world applications—such as analyzing a corpus of scientific articles, news reports, or social media posts. Without mechanisms to incorporate these inter-document dependencies, encoder-only transformers may fail to fully exploit the rich structural information present in large text corpora.

Although models like BERT are pre-trained on massive corpora of unlabeled text using unsupervised objectives such as masked language modeling, they require labeled data during their fine-tuning stages for specific tasks like text classification. Fine-tuning involves adapting the pre-trained model to a target task using labeled datasets often limited in size and scope. In contrast, real-world applications frequently involve vast amounts of unlabeled data. This limitation becomes particularly problematic in low-resource settings, where acquiring labeled data is expensive and time-consuming.

A final weakness of existing text classification approaches with models like BERT is their reliance on simple strategies for generating fixed-length representations, such as using the `[CLS]` token embedding or applying mean, max, and average pooling over token embeddings. While these methods are straightforward and efficient, they often fail to capture the detailed sequential structure of the data. By aggregating token embeddings without explicitly considering their order or localized interactions, these strategies risk omitting valuable contextual information, which can limit classification performance.

Given these limitations, there is a pressing need to develop models that address both local and global contexts in text classification. Models should capture fine-grained semantics within individual documents and incorporate global relationships across a corpus. Additionally, enabling models to effectively leverage unlabeled data during fine-tuning through semi-supervised approaches would significantly enhance their performance in real-world, low-resource scenarios. Moreover, a final weakness of existing transformer-based approaches is their reliance on simple strategies like `[CLS]` token embeddings or pooling techniques, which may fail to capture the sequential structure of the data fully. Therefore, the goal is to create robust models that integrate transformer encoders' contextual strength and leverage global structural information, utilize unlabeled data, and preserve the sequential order of textual inputs for more effective classification.

1.2 Graph Neural Networks in Text Classification

With the rapid advancement of technology, the volume of data being stored and processed globally has grown significantly. While traditional signal processing methods are applied to data with regular structures [50–67], graph signal processing (GSP) and graph neural network (GNN) techniques have emerged as popular solutions for handling irregular and non-Euclidean data [68–72].

In response to the limitations of encoder-only transformers, numerous works have attempted to integrate transductive GNNs with encoder-only transformers, such as BERT, to enhance their ability to capture global contextual information and leverage unlabeled data. These hybrid models combine the local contextual understanding of transformers with the global relational modeling capabilities of transductive GNNs. By representing documents and their relationships as graphs, GNNs can propagate information across nodes, thereby addressing the issue of inter-document dependencies. Additionally, GNNs that are inherently transductive can use labeled and unlabeled data to contribute to the training process through message passing. Moreover, existing approaches that utilize inductive GNNs for graph-based text classification in conjunction with encoders aim to reduce reliance on simple strategies like the `[CLS]` token and better capture fine-grained sequential and structural information. Despite their promising potential, these approaches come with their own set of challenges.

In the text classification literature, GNNs can operate in a transductive or inductive setting, depending on the specific method and application. The model requires the full graph during training and inference in a transductive setting, meaning test data must be present at training time. Conversely, the model can generalize to unseen nodes in an inductive setting by learning transferable patterns from the graph structure. One approach is not inherently superior to the other; the choice between transductive and inductive methods depends on the context of the problem. For instance, transductive methods are well-suited for scenarios where the entire graph is known beforehand, and there is no newly streaming data to perform classification. In contrast, inductive methods are preferable when new, unseen data needs to be incorporated dynamically during inference.

1.2.1 Transductive Approaches

Transductive GNNs requires the entire graph to be available during training and inference. The model learns node representations in this setting by propagating

information across the entire graph, including labeled and unlabeled nodes. This approach can achieve high performance when the complete graph is static and known in advance, as the model directly learns from the structure and relationships of all nodes.

The standard approach for text classification with GNNs in the transductive setting is to represent the whole corpus as a single graph, where each document is connected. This approach can utilize both labeled and unlabeled documents during training. Labeled documents compute the loss and update model parameters, while unlabeled documents contribute by enabling better representation learning through the message-passing process. Since each node aggregates information from its neighbors, unlabeled nodes indirectly influence the representations of labeled nodes, helping the model generalize better. This allows GNNs to exploit the structure of the entire graph to retrieve global structural information, improving performance even when the labeled data is sparse.

The first attempt to retrieve global structural information while preserving the contextual information from the large-scale pre-trained models, which is lacking in the encoder-only transformers, is made by BertGCN [73]. Unlike traditional approaches, they manage to represent the corpus as a single heterogeneous graph where documents and words are connected with semantic relations. They used homogeneous graph convolutional network (GCN) [74] as their GNN layer. Several other GNN-based methods for text classification are also proposed [75–84]. Moreover, graph-based approaches are also successfully developed for other NLP tasks such as multi-parallel word alignment, multilingual label propagation, modeling ideological salience & framing in polarized groups, text generation from knowledge graphs, peer review, and factuality evaluation in summarization [85–91].

However, transductive GNNs has notable limitations. Since they require the full graph during inference, they are unsuitable for scenarios where new, unseen nodes or documents may need to be dynamically classified. Furthermore, the computational cost of training transductive models can be high for large graphs, as the entire graph needs to be processed. These limitations make transductive GNNs less practical in dynamic environments where the graph structure

changes frequently, or new data is continuously added. Therefore, these limitations raise our first motivation to develop the Graph Receptive Transformer Encoder (GRTE) [1], a **transductive model capable of inductive inference, uses contextual information from the pre-trained models, and is scaleable for large graphs**.

As previously mentioned, it is common to see a pattern of using homogenous GNNs for heterogeneous graphs as in [73, 92]. These approaches utilize a single-weight matrix to handle all the connections between different types of nodes like word and document nodes. This simplification raises an issue because it needs to properly account for the various relationships between different types of nodes in the heterogeneous graph. Since corpora used in text classification are generally treated as heterogeneous graphs, there is a need for relational modeling of diverse data entities and their complex interactions. Therefore, this limitation raises our second motivation to come up with an approach that can accomplish **relational treatment of heterogenous text graphs** to obtain better performance, namely Relational Modeling for Heterogeneous Text Graphs (Text-RGNNs) [2].

1.2.2 Inductive Approaches

In contrast, inductive GNNs can generalize to previously unseen nodes during inference. These models can classify new nodes without accessing the entire graph at inference time by learning transferable patterns from the graph structure and node features. This inductive capability is advantageous in real-world scenarios involving dynamic data, such as social networks, citation graphs, or streaming environments. However, they struggle to utilize unlabeled data during the training process.

A common strategy for text classification using inductive GNNs involves representing each document as an individual graph and classifying it. Despite the effectiveness of existing inductive GNNs, current methods typically rely on static graph construction techniques that utilize statistical metrics, including term frequency-inverse document frequency (TF-IDF), point-wise mutual information

(PMI), and co-occurrence statistics [93, 94]. These techniques demand significant text preprocessing and often encounter scalability challenges.

In a different vein, visibility graphs (VGs) have been introduced as a novel approach for mapping time series data into a graph [95–98]. Then, [99] extends the notion to vector visibility graphs (VVGs) to map multivariate time series. The constructed graphs enable the extraction of sequential relations, facilitating improved analysis and insights.

In a broader perspective, transformers generate document embeddings as multivariate time series with inherent sequential relations, where VGs are designed to capture sequential relations. In contrast to statistical methods, VG preserves sequential topology and can utilize transformer output without text processing.

These inherent limitations of existing inductive models inspired our third motivation: to develop a solution that **dynamically generates graphs for each document and seamlessly integrates with transformer-based pipelines**, namely Enhancing Transformer Encoders with Vector Visibility Graph Neural Networks (VISPool) [3].

1.3 Contributions

Although the corresponding literature on text classification with transformer-based GNNs is vast and well established, with numerous previous works of the theory of practice, it is not the case that the problems mentioned earlier are tackled. Therefore, it still needs to be investigated.

This work categorizes our contributions into three main parts corresponding to the thesis's three main chapters. We start with providing (1) a transductive approach that is capable of inductive inference, using contextual information from pre-trained models, and scalable for large graphs, followed by (2) a model that treats heterogeneous text graphs in the relational matter and finally, (3)

an inductive approach that dynamically generates graphs per document that can seamlessly integrate with transformer pipelines. The overview of these main points is as follows:

- Existing transductive GNNs require the entire graph to be available during training and inference, making them impractical for dynamic environments where new data is continuously introduced. To overcome this limitation, we propose a model that can accomplish inductive inference on top of the features learned through transductive learning. Our approach leverages the contextual information captured by pre-trained transformer models while maintaining scalability for large graphs. This solution directly addresses our first motivation to design a **transductive model capable of inductive inference, using contextual information from pre-trained models, and scalable for large graphs**.

- Many existing transductive GNN-based methods oversimplify the relationships in heterogeneous graphs by using homogeneous GNNs, which employ a single-weight matrix for all types of nodes and edges. This simplification neglects the diverse interactions among different types of nodes, such as word and document nodes. We introduce a relational modeling approach that provides a fine-grained treatment of heterogeneous text graphs to address this issue. Our method enhances performance by explicitly capturing the complex relationships between different node types, fulfilling our second motivation to develop a solution for **relational treatment of heterogeneous text graphs**.

- Existing inductive GNNs typically rely on static graph construction techniques based on statistical metrics, such as TF-IDF, PMI, or co-occurrence statistics. These methods require extensive preprocessing and are prone to scalability issues in large-scale applications. We propose a novel dynamic graph generation mechanism that constructs graphs on the fly for each document, eliminating the need for static preprocessing steps. Furthermore, our solution seamlessly integrates with transformer-based pipelines, making it more adaptable and efficient. This contribution aligns with our third

motivation to design a method that **dynamically generates graphs per document and seamlessly integrates with transformer pipelines**.

Through these contributions, we provide a comprehensive solution that addresses key challenges in text classification using transformer and GNN-based methods, ensuring better generalization, scalability, and adaptability in real-world applications.

1.4 Organization

The remainder of this thesis is structured as follows: Chapter 2 introduces the mathematical notation and foundational concepts related to classification with GNNs that are essential for understanding the proposed methods. In Chapter 3, we present the GRTE model, which addresses the challenge of incorporating graph-based relational information into transformer fine-tuning for improved classification in low-resource settings. Chapter 4 introduces the Text-RGNNs framework, designed to handle heterogeneous text graphs and capture fine-grained inter-document relationships. Chapter 5 describes the VISPool approach, which solves the problem of leveraging sequential structure in document embeddings by dynamically constructing vector visibility graphs (VVGs) and applying GCNs. Finally, Chapter 6 summarizes the key findings and contributions of this thesis and outlines potential directions for future work.

Chapter 2

Foundations of Classification with Graph Neural Networks

This chapter lays the groundwork for understanding text classification with GNNs. We introduce the notational conventions used throughout the thesis in Section 2.1, including the representation of matrices, vectors, and sets. In Section 2.2, we define graphs and graph signals, establishing the fundamental concepts required to model textual data as graphs. Finally, in Section 2.3, we present a general framework for node classification using GNNs, illustrating how graph-based methods can be applied to text classification tasks by representing textual units as nodes and their relationships as edges.

2.1 Notation

Bold capital letters denote matrices $\mathbf{A}$, while bold lowercase letters denote vectors $\mathbf{x}$. For a set $\mathcal{V}$, $|\mathcal{V}|$ denotes its cardinality. $(\cdot)^T$ denotes transpose of their argument. $\mathbf{I}_n$ denotes an identity matrix of size n, and we omit the subscript if the context is clear. $\mathbf{0}_{m\times n}$ and $\mathbf{1}_{m\times n}$ denote matrices of zeros and ones with m rows and n columns.

2.2 Graphs and Graphs Signals

Following the notation provided in [100], let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$ be a graph where $\mathcal{V} = \{v_0, v_1, \ldots, v_{N-1}\}$ is the set of nodes (vertices) with $|\mathcal{V}| = N \in \mathbb{Z}^+$, $\mathcal{E}$ is the set of edges and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the weighted adjacency matrix of $\mathcal{G}$. An edge $e = (i, j)$ is an element of $\mathcal{E}$, and if there is an edge from v_i to v_j then $\mathbf{A}(i, j) \neq 0$, where $\mathbf{A}(i, j)$ is the element at i^{th} row and j^{th} column of $\mathbf{A}$. If $\mathbf{A}(i, j) = 0$, then there is no edge connection from v_i to v_j. A graph is said to be *undirected* if the adjacency matrix $\mathbf{A}$ is symmetric. Let a graph signal $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$, be a mapping between the set of vertices $\mathcal{V}$ to $\mathbb{R}^N$ so that each vertex is mapped to a real number as $\mathbf{x} : \mathcal{V} \to \mathbb{R}$ such that $v_n \to x_n$ [101–103].

2.3 Classification with Graph Neural Networks

While it is challenging to provide a generalized framework encompassing all possible approaches for node classification using GNNs—mainly because different models are designed for either inductive or transductive settings and may employ varying notations—we present a simplified, general methodology. This formulation aims to capture the core idea of GNN-based node classification while remaining flexible enough to cover a broad range of applications. For clarity, we focus on the standard message-passing mechanism and provide node and matrix equations for node updates. The following notation is derived from one of the frontiers in the field, namely Graph Convolutional Network (GCN) [74].

In general, node classification with GNNs involves representing the data as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes, and $\mathcal{E}$ is the set of edges that define relationships between nodes. Each node $v_i \in \mathcal{V}$ is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^d$, and the feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ stacks these vectors for all nodes. The graph structure is encoded by the adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$.

The task of the GNN is to learn meaningful node representations by propagating and aggregating information from a node's neighbors. This is achieved

through iterative updates of the node representations over multiple layers. At each layer l, the representation of a node v_i is updated by aggregating information from its neighbors $\mathcal{N}(i)$ and combining it with its features. The node-wise update at layer l is given by:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \mathbf{W}^{(l)} \mathbf{h}_j^{(l)} + \mathbf{W}_0^{(l)} \mathbf{h}_i^{(l)}\right) \tag{2.1}$$

In this equation, $\mathbf{h}_i^{(l)}$ represents the feature vector of node v_i at layer l, $\mathbf{W}^{(l)}$ and $\mathbf{W}_0^{(l)}$ are learnable weight matrices, and $\sigma(\cdot)$ is a non-linear activation function, such as ReLU.

Alternatively, the same update can be expressed in matrix form for all nodes as:

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathbf{A}\mathbf{H}^{(l)}\mathbf{W}^{(l)} + \mathbf{H}^{(l)}\mathbf{W}_0^{(l)}\right) \tag{2.2}$$

Here, $\mathbf{H}^{(l)} \in \mathbb{R}^{|\mathcal{V}| \times d_l}$ denotes the matrix of node representations at layer l, and $\mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ and $\mathbf{W}_0^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ are learnable weight matrices. The term $\mathbf{A}\mathbf{H}^{(l)}\mathbf{W}^{(l)}$ aggregates information from neighboring nodes, while the term $\mathbf{H}^{(l)}\mathbf{W}_0^{(l)}$ updates the current node representations based on their own features.

After L layers of updates, the final node representations $\mathbf{H}^{(L)}$ are obtained. These representations can be used for node classification by passing them through a softmax classifier:

$$\hat{\mathbf{Y}} = \text{softmax}\left(\mathbf{H}^{(L)}\mathbf{W}_{\text{cls}} + \mathbf{b}_{\text{cls}}\right) \tag{2.3}$$

In this equation, $\hat{\mathbf{Y}} \in \mathbb{R}^{|\mathcal{V}| \times C}$ represents the predicted class probabilities for each node, C denotes the number of classes, and $\mathbf{W}_{\text{cls}} \in \mathbb{R}^{d_L \times C}$ and $\mathbf{b}_{\text{cls}} \in \mathbb{R}^C$ are the learnable parameters of the classifier.

This general framework for node classification can be directly applied to text classification by representing a text corpus as a graph. In such cases, nodes may represent documents, sentences, or words, and edges capture their relationships based on semantic similarity, co-occurrence, or other criteria. By learning meaningful node representations, GNNs can effectively classify textual units while leveraging the graph's local and global structural information.

While we have introduced a standard set of notations to describe our methodology, it is essential to note that GNN approaches can vary significantly depending on the specific problem and application. Different models may require additional or modified notations to accommodate unique graph structures, types of relationships, or domain-specific features. Therefore, in subsequent sections, readers may encounter extra notations introduced as needed to address the context of particular approaches or experiments.

Chapter 3

GRTE: Graph Receptive Transformer Encoder

In this chapter, we present our approach to addressing the limitations of existing transductive GNNs by enabling inductive inference while leveraging the strengths of pre-trained transformer models. Our method builds on the foundational idea of combining transductive learning with inductive capabilities, allowing the model to generalize to unseen data without requiring access to the entire graph during inference. We also ensure that the proposed solution maintains scalability for large graphs by efficiently integrating graph-based relational modeling with transformer-based contextual embeddings. The subsequent sections detail the design of our model, its components, and the underlying mechanisms that enable inductive inference on dynamically evolving data.

3.1 Problem Definition

In the context of text classification, notation is inconsistent between the existing works in the literature. GNN-based text classification methods, e.g., TextGCN, Hete-GCN, and BertGCN [73, 92, 104], use different notations across their works.

Therefore, we introduce our notation to overcome the possible conflicts.

Our problem definition assumes a corpus consists of n documents with a total of m unique words. We represent each document and word by their corresponding embedding vector. For the i^{th} document d_i and the j^{th} unique word w_j; $\mathbf{v}_i \in \mathbb{R}^p$ and $\mathbf{u}_j \in \mathbb{R}^q$ denote the p- and q-dimensional embedding vectors of d_i and w_j, respectively. The matrix $\mathbf{V} \in \mathbb{R}^{n \times p}$ and $\mathbf{U} \in \mathbb{R}^{m \times q}$ denote the document and word embedding matrices, respectively. Documents in a corpus may or may not have a corresponding class label, so we use $\mathcal{Y}_D$ to denote the set of labeled document indices. For total number of k classes in a corpus and $i \in \mathcal{Y}_D$, $\mathbf{y}_i \in \mathbb{R}^k$ denotes the 1-hot representation of d_i's target class, and $\mathbf{Y} \in \mathbb{R}^{|\mathcal{Y}_D| \times k}$ denotes the target class matrix. For the classification task, we denote the model output as $\mathbf{P} \in \mathbb{R}^{n \times k}$, which is the classification probability distribution of n documents over k classes.

On the other hand, we extract the graph matrices from the corpus by representing each document and unique word as a graph node. Since we have two types of nodes (documents and words), we have three types of graph matrices: *document-document* graph matrix $\mathbf{N} \in \mathbb{R}^{n \times n}$, *word-word* graph matrix $\mathbf{F} \in \mathbb{R}^{m \times m}$, and *document-word* graph matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$. The *document-word* graph is heterogeneous since it contains more than one node type, whereas the other two are homogeneous graphs. All these graphs are constructed by connecting the nodes with calculated weights based on the relation between the pairs according to several metrics, e.g., *Positive Point-wise Mutual Information (PPMI)*, *Term Frequency-Inverse Document Frequency (TF-IDF)*, and cosine similarity, but one can use any metric. Then, any weighted adjacency matrix $\mathbf{A}$ can be constructed based on $\mathbf{N}, \mathbf{F}, \mathbf{X}, \mathbf{I}_n$ and $\mathbf{I}_m$ sub-graph matrices, regarding the dimensions, to represent whole corpus as a heterogeneous graph, where $\mathbf{I}_n$ is the size n identity matrix. These matrices are crucial to generating several versions of graph convolutional layers with trainable weight matrices $\mathbf{W}$, as we address them as *paths* in the following sections. In this sense, we denote $\mathbf{S} \in \{\mathbf{N}, \mathbf{F}, \mathbf{X}, \mathbf{X}^\top, \mathbf{I}_n, \mathbf{I}_m\}$ and $\mathbf{E} \in \{\mathbf{U}, \mathbf{V}\}$ as any generic sub-graph and embedding matrices, respectively.

Our approach can be summarized as follows. With a given corpus of documents $\{d_i\}_{i=1}^n$ with unique words $\{w_j\}_{j=1}^m$, we first generate the pre-trained model

(BERT in this case) document embeddings $\mathbf{V}_{\text{BERT}}$. Then, we construct $\mathbf{N}$, $\mathbf{F}$, and $\mathbf{X}$ matrices for several versions of GCN layers. Ultimately, we aim to learn improved word embeddings $\mathbf{U}$ using large-scale pre-trained models and GCNs together in a transductive manner. In parallel, we aim to train an inductive classifier that uses the word embedding matrix $\mathbf{U}$ to compute improved document embeddings $\mathbf{V}$ to predict the target class of the unlabeled documents. In that manner, our model can be generalized to inductive settings. We search for the optimal embeddings through pre-trained models and GCNs and aggregate them to improve classification performance. Our primary motivation is to simultaneously exploit the *contextual* information captured by large-scale pre-trained and *global* information captured by GCNs. We summarize our notation in Table 3.1, note that we also use subscripts to denote generation methods, e.g., $\mathbf{V}_{\text{BERT}}, \mathbf{P}_{\text{GCN}}$ and superscripts for the layer indices, e.g., $\mathbf{U}^{(0)}, \mathbf{W}^{(1)}$.

Table 3.1: Summary of notation.

Symbol	**Meaning**
n	number of documents in a corpus
m	number of unique words in a corpus
k	total number of classes
d_i	i^{th} document, $i \in \{1, \ldots, n\}$
w_j	j^{th} unique word, $j \in \{1, \ldots, m\}$
$\mathcal{Y}_D$	set of labeled document indices
$\mathbf{y}_i \in \mathbb{R}^k$	1-hot coding of d_i's target class for total k classes
$\mathbf{v}_i \in \mathbb{R}^p$	p-dimensional embedding of d_i
$\mathbf{u}_j \in \mathbb{R}^q$	q-dimensional embedding of w_j
$\mathbf{Y} \in \mathbb{R}^{\lvert\mathcal{Y}_D\rvert \times k}$	target class matrix with rows $\mathbf{y}_i, \forall i \in \mathcal{Y}_D$
$\mathbf{V} \in \mathbb{R}^{n \times p}$	document embedding matrix with i^{th} row $\mathbf{v}_i$
$\mathbf{U} \in \mathbb{R}^{m \times q}$	word embedding matrix with j^{th} row $\mathbf{u}_j$
$\mathbf{A} \in \mathbb{R}^{r \times r}$	weighted adjacency matrix of a graph with r nodes
$\mathbf{I}_n \in \mathbb{R}^{n \times n}$	identity matrix of size n
$\mathbf{N} \in \mathbb{R}^{n \times n}$	*document-document* homogeneous graph matrix
$\mathbf{F} \in \mathbb{R}^{m \times m}$	*word-word* homogeneous graph matrix
$\mathbf{X} \in \mathbb{R}^{n \times m}$	*document-word* heterogeneous graph matrix
$\mathbf{P} \in \mathbb{R}^{n \times k}$	classification probability distributions
$\mathbf{W} \in \mathbb{R}^{n_{\text{in}} \times n_{\text{out}}}$	any trainable weight matrix
$\mathbf{S} \in \{\mathbf{N}, \mathbf{F}, \mathbf{X}, \mathbf{X}^\top, \mathbf{I}_n, \mathbf{I}_m\}$	any sub-graph matrix
$\mathbf{E} \in \{\mathbf{U}, \mathbf{V}\}$	any embedding matrix

3.2 Related Work

In this section, we review the key related works that form the foundation for our proposed model. Specifically, we focus on three prominent models in the field—TextGCN, BertGCN, and Hete-GCN—and highlight how they construct adjacency matrices, as these methods directly influence our model improvements. We begin by discussing transductive classifiers, including TextGCN and BertGCN, in Section 3.2.1, and then move on to inductive classifiers, specifically Hete-GCN, in Section 3.2.2. This overview provides the necessary context for understanding the design choices and enhancements made in our approach.

3.2.1 Transductive Classifiers

TextGCN [92] proposed a method by using the equations of GCN with an adjacency matrix as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{F}_{m\times m} & \mathbf{X}^{\top}_{m\times n} \\ \mathbf{X}_{n\times m} & \mathbf{I}_n \end{bmatrix}, \tag{3.1}$$

where n and m are the total numbers of documents and unique words in the vocabulary, respectively. Note that TextGCN uses *document-word* and *word-word* graph matrices but excludes *document-document* relation graph matrix $\mathbf{N}$ mentioned in Section 3.1. They use a two-layer network and obtain document embeddings as a by-product of the layers. Then, TextGCN's adjacency matrix given in Eq. (3.1) is also defined element-wise as follows:

$$\mathbf{A}(i,j) = \begin{cases} \text{PPMI}(i,j), & \text{if } i,j \text{ are words} \\ \text{TF-IDF}(i,j), & \text{if } i \text{ is document, } j \text{ is word} \\ 1, & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

The summation $n+m$ is huge for large-scale applications most of the time, which implies that TextGCN is unsuitable for such cases. Also, when the number of

model parameters is large, learning them with a small number of labeled examples is impossible. Next, TextGCN learns document embeddings in a transductive manner only. There is no inductive inference on top of transductive learning. The learned document embeddings cannot be generalized to classify unseen documents.

To leverage the performance of TextGCN, BertGCN [73] uses large-scale pretrained models on top of GCN layers in TextGCN. They construct a heterogeneous graph over the dataset and represent documents as nodes. Node embedding of the documents is BERT representation. The constructed graph is the same as the one in TextGCN. The document embeddings of the nodes are the `[CLS]` tokens retrieved from fine-tuned BERT. Document embeddings are updated simultaneously during training with the GCN layer. In BertGCN, `[CLS]` token embeddings of the documents that are retrieved from BERT are represented as $\mathbf{V}_{\text{BERT}}$ and words are represented as $\mathbf{U}_{\text{BERT}}$ where each document and word has dimension d. Since BertGCN only passed document embeddings as input due to their non-sparse input matrix implementation, $\mathbf{U}_{\text{BERT}} = \mathbf{0}$. Therefore, the input embedding matrix to the model can be given as follows:

$$\begin{bmatrix} \mathbf{U}^{(0)} \\ \mathbf{V}^{(0)} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{V}_{\text{BERT}} \end{bmatrix}_{(n+m)\times d}. \tag{3.3}$$

All the limitations mentioned earlier for TextGCN are still valid since BertGCN did not change the structure of TextGCN. BertGCN elevates TextGCN's performance by using BERT document embeddings. However, BertGCN can only learn in a transductive manner and needs to learn many parameters. Also, they did not propose inductive inference on top of it. They interpolate the BERT and GCN prediction results at the final stage with the parameter λ.

3.2.2 Inductive Classifiers

One of the models that can make inductive inferences on top of transductive learning is Hete-GCN [104]. They proposed a model that uses heterogeneous graphs

to decrease the duration of training and testing for the TextGCN model. The adjacency matrix given in Eq. (3.2) slows down the layer computations in both TextGCN and BertGCN. Hete-GCN does not use a large-scale pre-trained model that can leverage the generalization performance of the learned word embeddings.

3.3 Methodology

In this section, we propose the architecture of our model in detail. First we provide graph construction procedure in Section 3.3.1. Then, we provide the GCN layer equations in Section 3.3.2. After that, we introduce possible model types which differentiate depending on the embedding they use and training regime in Section 3.3.3. Next, we provide complexity analysis and their implications in Section 3.3.4 to analyze and compare our model with existing works. Then, we introduce how we generalize transductive learning to inductive inference in Section 3.3.5. Finally, in Section 3.3.6, we show that TextGCN is a particular case of our model by deriving TextGCN equations from our GCN layer equations given in Section 3.3.2.

3.3.1 Graph Construction

Following the setup described in TextGCN [92], we prepare the graphs for datasets. Each raw text is cleaned and tokenized. Test and train IDs for documents are given within the dataset. Therefore, we only make a 90 to 10 split in the train IDs to obtain the train and validation set. Previously in Section 3.1, we defined the essential graph relation matrices, introduced how we obtain those matrices, and the metrics we used to denote edges between nodes. To construct $\mathbf{F}$, we use the PPMI scores as weights. $\mathbf{X}$ is a *document-word relation* matrix consisting of TF-IDF scores. $\mathbf{X}$ and $\mathbf{F}$ defined as in Eq. (3.1) and Eq. (3.2). $\mathbf{N}$ is *document-document relation* matrix, which is obtained from $\mathbf{X}$ with top 25 neighbours obtained using cosine similarity score. We use all documents to compute PPMI and TF-IDF in a transductive setting, but only training documents

are used during an inductive setting. Unseen tokens are removed from validation and test documents.

3.3.2 GCN Layers and Model Paths

We propose GCN layers that can be constructed from any chosen sub-graph matrix $\mathbf{S} \in \{\mathbf{N}, \mathbf{F}, \mathbf{X}, \mathbf{X}^\top, \mathbf{I}_n, \mathbf{I}_m\}$, a suitable initial embedding matrix $\mathbf{E} \in \{\mathbf{U}, \mathbf{V}\}$, and a desired activation function $\phi \in \{\textsc{Softmax}, \text{ReLU}, \dots\}$. Therefore, in the generic sense, we construct a GCN layer as follows:

$$\mathbf{E}^{(i+1)} = \phi_i(\mathbf{S}^{(i)}\mathbf{E}^{(i)}\mathbf{W}^{(i)}), \tag{3.4}$$

where $\phi_i, \mathbf{S}^{(i)}, \mathbf{E}^{(i)}, \mathbf{W}^{(i)}$ are the i^{th} layer activation function, sub-graph matrix, embedding matrix, and trainable weight matrix, respectively. For a selected $\mathbf{S}^{(i)}$, dimensions need to match the $\mathbf{E}^{(i)}$ dimensions. For example, if we choose $\mathbf{S}^{(0)} = \mathbf{N} \in \mathbb{R}^{n \times n}$, then we need an embedding $\mathbf{E}^{(0)}$, with first dimension n, where $\mathbf{V} \in \mathbb{R}^{n \times p}$ is a suitable candidate. Then, we can choose $\mathbf{E}^{(0)} = \mathbf{V}^{(0)} \in \{\mathbf{I}_n, \mathbf{V}_{\text{BERT}}\}$, so we construct a GCN layer that takes document embeddings as input and produces document embeddings as output. Similarly, we can construct a GCN layer with word embedding input and output by choosing $\mathbf{S}^{(0)} = \mathbf{F}$ and $\mathbf{E}^{(0)} = \mathbf{U}^{(0)} \in \{\mathbf{I}_m, \mathbf{U}_{\text{BERT}}\}$. We can even construct a GCN layer that takes word embeddings as input and produces document embeddings as output, and vice versa, by choosing $\mathbf{S}^{(0)} = \mathbf{X}$, $\mathbf{E}^{(0)} = \mathbf{U}^{(0)}$ and $\mathbf{S}^{(0)} = \mathbf{X}^\top$, $\mathbf{E}^{(0)} = \mathbf{V}^{(0)}$. Given an $\mathbf{S}^{(i)}$ or $\mathbf{E}^{(i)}$, the remaining selections are constrained based on matrix dimensions. We provide examples of the GCN layers with ReLU activation as follows:

3.3.2.1 $\mathbf{F}$ − ReLU GCN Layer

Accepts word embeddings $\mathbf{U}^{(i)}$ as input then it produces word embeddings $\mathbf{U}^{(i+1)}$:

$$\mathbf{U}^{(i+1)} = \text{ReLU}(\mathbf{F}\mathbf{U}^{(i)}\mathbf{W}^{(i)}). \tag{3.5}$$

3.3.2.2 **TX** – ReLU GCN Layer

Accepts document embeddings $\mathbf{V}^{(i)}$ as input then it produces word embeddings $\mathbf{U}^{(i+1)}$:

$$\mathbf{U}^{(i+1)} = \text{ReLU}(\mathbf{X}^\top \mathbf{V}^{(i)} \mathbf{W}^{(i)}). \tag{3.6}$$

3.3.2.3 **X** – ReLU GCN Layer

Accepts word embeddings $\mathbf{U}^{(i)}$ as input then it produces document embeddings $\mathbf{V}^{(i+1)}$:

$$\mathbf{V}^{(i+1)} = \text{ReLU}(\mathbf{X}\mathbf{U}^{(i)} \mathbf{W}^{(i)}). \tag{3.7}$$

3.3.2.4 **N** – ReLU GCN Layer

Accepts document embeddings $\mathbf{V}^{(i)}$ as input then it produces document embeddings $\mathbf{V}^{(i+1)}$:

$$\mathbf{V}^{(i+1)} = \text{ReLU}(\mathbf{N}\mathbf{V}^{(i)} \mathbf{W}^{(i)}). \tag{3.8}$$

Another important layer in our model is the interpolation layer, where we aggregate the predictions of the BERT and GCN outputs, which we further discuss. Interpolation of fine-tuned BERT with GCN block can be viewed as follows, $\lambda \in [0, 1]$:

$$\mathbf{P}_{\text{GCN}} = \textsc{Softmax}(\mathbf{S}^{(\text{final})} \mathbf{E}^{(\text{final})} \mathbf{W}_{\text{GCN}}^{(\text{final})}), \tag{3.9}$$

$$\mathbf{P}_{\text{BERT}} = \textsc{Softmax}(\mathbf{V}_{\text{BERT}} \mathbf{W}_{\text{BERT}}), \tag{3.10}$$

$$\mathbf{P} = \lambda \mathbf{P}_{GCN} + (1 - \lambda) \mathbf{P}_{\text{BERT}}. \tag{3.11}$$

GCN layers that use homogeneous graphs for message passing have the same input and output entity type (e.g., words or documents). On the other hand, GCN layers that use heterogeneous graphs for message passing consume one entity type as input and produce another. Therefore, Eq. (3.5) is a homogeneous layer, and Eq. (3.7) is a heterogeneous layer. Also, neither TextGCN nor BertGCN uses *document-document* relation matrix $\mathbf{N}$. Still, we offer the flexibility to use

it as a GCN layer. The GCN block consists of a GCN layer with LayerNorm and Dropout. GCN block of our architecture with Linear layer is shown in Fig. 3.1.

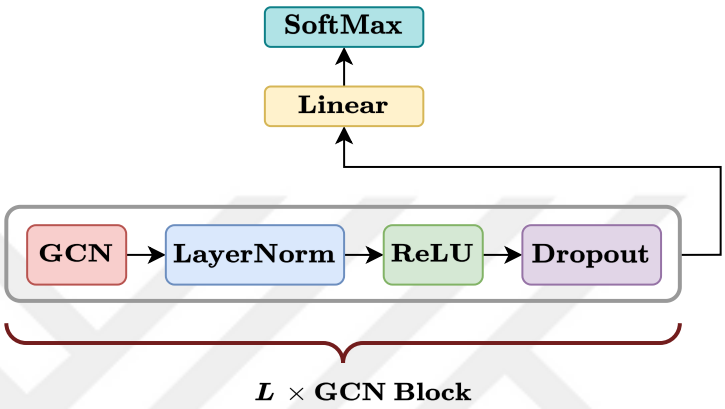


Figure 3.1: The GCN Block consists of several GCN layers combined with LayerNorm and Dropout at the left side. The output of this GCN Block is fed into the linear layer.

We also define "model paths" by concatenating multiple GCN blocks. This means we can combine as many GCN blocks as we want to construct a model path, given that the embedding dimensions match. We can combine an $\mathbf{S}^{(0)}$ – ReLU and $\mathbf{S}^{(1)}$ – Softmax GCN layers to form $\mathbf{S}^{(0)} - \mathbf{S}^{(1)}$ path. In this paper, for a direct and fair comparison to TextGCN, HeteGCN, and BertGCN, we focus on two-layer GCN structures, so we employ $\mathbf{F} - \mathbf{X}$, $\mathbf{TX} - \mathbf{X}$, $\mathbf{N} - \mathbf{N}$, and $\mathbf{X} - \mathbf{N}$ paths, where $\mathbf{TX}$ corresponds $\mathbf{X}^{\top}$.

Additionally, the number of GCN layers in our model is chosen as two to mitigate the over-smoothing problem commonly encountered in deep GNNs. As the number of layers in GNNs increase, node representations become overly similar, leading to a loss of discriminative power. This phenomenon, known as over-smoothing [105], causes the model to struggle to distinguish between nodes with different labels, ultimately degrading its performance. By limiting the number of GCN layers to two, we aim to balance the ability to capture meaningful neighborhood information while preserving the distinctiveness of node features. This design choice helps maintain the model's effectiveness, particularly when preserving fine-grained differences between nodes, which is crucial for accurate classification.

Note that any combination of any length path can be generated, given that consecutive embedding dimensions match the selected sub-graph matrices. For example, with $\mathbf{E}^{(0)} = \mathbf{U}_{\text{BERT}}$, $\mathbf{F} - \mathbf{X} - \mathbf{N}$ is a valid path, but $\mathbf{X} - \mathbf{F}$ is not since document embedding is not a valid input for $\mathbf{F}$.

We utilize BERT and its variants for the transformer encoder component of the GRTE since the BERT variants reach the state-of-art results in the text classification tasks [4, 48, 106–108] compared to existing transformer encoders in literature. However, it is essential to indicate that our model can work with any transformer encoder block since any embedding representation retrieved from the encoder is sufficient to be used as an input to the GCN block. At the final stage, the model can be classified by interpolating the transformer encoder block and different GCN blocks obtained through the combinations of paths presented in Section 3.3.2. In Fig. 3.2, possible four different paths namely as $\mathbf{F} - \mathbf{X}$, $\mathbf{X} - \mathbf{N}$, $\mathbf{TX} - \mathbf{X}$, and $\mathbf{N} - \mathbf{N}$ are given as examples for 2 layer GRTE.

3.3.3 Model Types

In parallel to model "paths", we also propose four gradual model "types" for GRTE, where they differ in the context of their training scheme. With this approach, we can also analyze which part of our model contributes to the overall performance.

3.3.3.1 Type I

The first model type does not have any connection with BERT. We used an identity matrix as GCN input for all the GCN layers. This approach is used to see whether our proposed GCN block in Fig. 3.1 is better than Hete-GCN and TextGCN alone. In short, we set $\mathbf{V}^{(0)} = \mathbf{I}_n$ and $\mathbf{U}^{(0)} = \mathbf{I}_m$.

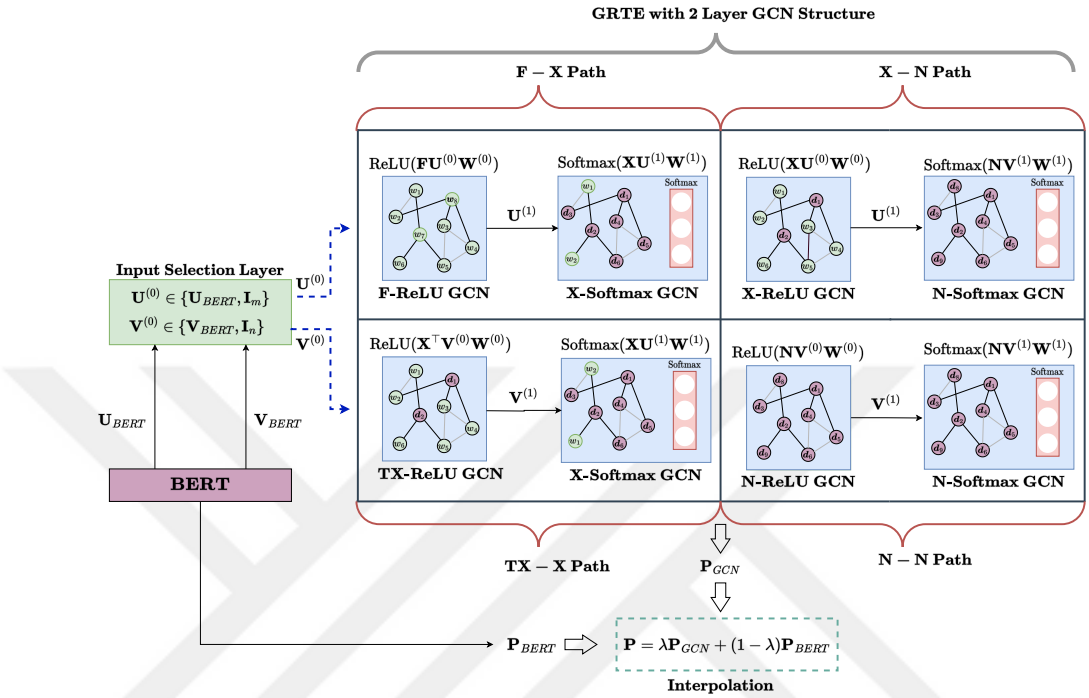


Figure 3.2: General architecture of the proposed GRTE for two layers. The model accepts document and word embeddings in the input stage. These embeddings are further used in the compatible GCN layer as input and then passed to the second GCN layer. At the final stage, there is an optional interpolation layer depending on the model variant.

3.3.3.2 Type II

For this type, we first fine-tune BERT on the dataset, then use the `[CLS]` token as document representation. These document representations are used as input matrix to possible GCN paths. Notice that we only use BERT embeddings as input to GCN layers that accept document embeddings. In short, we set $\mathbf{V}^{(0)} = \mathbf{V}_{\text{BERT}}$ and $\mathbf{U}^{(0)} = \mathbf{I}_m$.

3.3.3.3 Type III

We follow the same approach in Type II but also include interpolation at the end of the prediction layer of GCN. So, we obtain the predictions by aggregating

BERT and GCN predictions as $\mathbf{P} = \lambda \mathbf{P}_{GCN} + (1 - \lambda)\mathbf{P}_{\text{BERT}}$ for $\lambda \in [0, 1]$.

3.3.3.4 Type IV

We initiate each epoch for the Type IV variant by computing document embeddings using the current encoder weights. These embeddings are further used as representations for document nodes. A batch is sampled from labeled training documents throughout each iteration to obtain predictions from the encoder component. Simultaneously, predictions are obtained solely from the GCN component for the training instances in the current batch. The final predictions interpolate outputs from the encoder and GCN components, with the interpolation governed by the hyperparameter λ. In short, the $\mathbf{V}^{(0)} = \mathbf{V}_{\text{BERT}}$ was a fixed embedding matrix for previous types, but in this type the BERT embeddings also get updated with training.

Table 3.2: Training regimes for model types.

	Fine-tuned BERT	**Interpolation**	**Joint Training**
Type I	✗	✗	✗
Type II	✓	✗	✗
Type III	✓	✓	✗
Type IV	✓	✓	✓

3.3.4 Complexity Analysis

To comprehensively assess the performance characteristics of our proposed GRTE, we present a detailed analysis encompassing theoretical computation time and the overall size of learnable parameters. The tabulated results encapsulate the total computation of a single forward pass employing a 2-layer GCN model for TextGCN, BertGCN, and GRTE while accounting for the total parameter size, irrespective of the dataset. The calculations are for a corpus with n documents, m unique words, p-dimensional encoder embeddings, h hidden dimension and k

target classes (see Table 3.1). We also denote forward computational cost and trainable parameter count of the BERT model as $\mathcal{C}_{\text{BERT}}$ and n_{BERT}, respectively. The results are given in Table 3.3.

The *Forward Computation* column in Table 3.3 represents the computational complexity for the models during the forward pass. For all datasets, there is an assumption that $n, m > p, h > k$. From Table 3.3, one can observe that for both TextGCN and BertGCN, total computation time involves n^2, m^2 and mn terms because of the adjacency matrix structure. Unlike these approaches, the paths introduced in our model can omit at least one of these terms. This is one of the main reasons that we can do faster forward-pass calculations.

For example, when we compare the TextGCN with the $\mathbf{F}-\mathbf{X}$ path. We denote the forward computational cost of TextGCN and GRTE ($\mathbf{F}-\mathbf{X}$) as $\mathcal{C}_{\text{TextGCN}}$ and $\mathcal{C}_{\text{GRTE}(\mathbf{F}-\mathbf{X})}$, respectively. As can be verified from Table 3.3:

$$\begin{aligned}\mathcal{C}_{\text{TextGCN}} &= \mathcal{O}((h+k)(m+n)^2),\\ \mathcal{C}_{\text{GRTE}(\mathbf{F}-\mathbf{X})} &= \mathcal{O}(hm^2 + km(n+h)),\end{aligned}$$

or equivalently,

$$\begin{aligned}\mathcal{C}_{\text{TextGCN}} &= \mathcal{O}(hm^2 + kmn + hmn + km^2 + (h+k)n^2),\\ \mathcal{C}_{\text{GRTE}(\mathbf{F}-\mathbf{X})} &= \mathcal{O}(hm^2 + kmn + hmk).\end{aligned}$$

The difference in total computation cost is in the following order:

$$\begin{aligned}\mathcal{C}_{\text{diff}} &\triangleq \mathcal{C}_{\text{TextGCN}} - \mathcal{C}_{\text{GRTE}(\mathbf{F}-\mathbf{X})}\\ &= \mathcal{O}(hm(n-k) + km^2 + (h+k)n^2).\end{aligned}$$

On the other hand, the *# of Trainable Parameters* column represents the total number of parameters that need to be trained for the model. Since we omit joint training for Type I, Type II, and Type III, we significantly reduce the number of trainable parameters from the BERT variant. Also, for these types, since

we use bipartite graphs as GCN layers, the trainable parameter count reduces compared to TextGCN. In conclusion, incorporating bipartite graphs in GCN layers diminishes the number of trainable parameters and decreases the overall computational load. This, in turn, directly accelerates the training duration for a single epoch.

Table 3.3: Dataset independent matrix size dimension, total computation, and trainable parameter count.

Method	**First Layer**			**Second Layer**			**Forward Computation**	**# of Trainable Parameters**
	$\mathbf{S}^{(0)}$	$\mathbf{E}^{(0)}$	$\mathbf{W}^{(0)}$	$\mathbf{S}^{(1)}$	$\mathbf{E}^{(1)}$	$\mathbf{W}^{(1)}$		
TextGCN	$(n+m)\times(n+m)$	$(n+m)\times(n+m)$	$(n+m)\times h$	$(n+m)\times(n+m)$	$(n+m)\times h$	$h\times k$	$\mathcal{O}((h+k)(n+m)^2)$	$(n+m+k)h$
BertGCN	$(n+m)\times(n+m)$	$(n+m)\times p$	$p\times h$	$(n+m)\times(n+m)$	$(n+m)\times h$	$h\times k$	$\mathcal{O}((h+k)(n+m)^2)+\mathcal{C}_{\text{BERT}}$	$(p+k)h+n_{\text{BERT}}$
GRTE (F-X)								
Type I	$m\times m$	$m\times m$	$m\times h$	$n\times m$	$m\times h$	$h\times k$	$\mathcal{O}(m^2h+mk(n+h))$	$(m+k)h$
Type II	$m\times m$	$m\times m$	$m\times h$	$n\times m$	$m\times h$	$h\times k$	$\mathcal{O}(m^2h+mk(n+h))$	$(m+k)h$
Type III	$m\times m$	$m\times m$	$m\times h$	$n\times m$	$m\times h$	$h\times k$	$\mathcal{O}(m^2h+mk(n+h))$	$(m+k)h$
Type IV	$m\times m$	$m\times m$	$m\times h$	$n\times m$	$m\times h$	$h\times k$	$\mathcal{O}(m^2h+mk(n+h))+\mathcal{C}_{\text{BERT}}$	$(m+k)h+n_{\text{BERT}}$
GRTE (TX-X)								
Type I	$m\times n$	$n\times n$	$n\times h$	$n\times m$	$m\times h$	$h\times k$	$\mathcal{O}(nh(m+n)+mk(n+h))$	$(n+k)h$
Type II	$m\times n$	$n\times p$	$p\times h$	$n\times m$	$m\times h$	$h\times k$	$\mathcal{O}(nh(m+p)+mk(n+h))$	$(p+k)h$
Type III	$m\times n$	$n\times p$	$p\times h$	$n\times m$	$m\times h$	$h\times k$	$\mathcal{O}(nh(m+p)+mk(n+h))$	$(p+k)h$
Type IV	$m\times n$	$n\times p$	$p\times h$	$n\times m$	$m\times h$	$h\times k$	$\mathcal{O}(nh(m+p)+mk(n+h))+\mathcal{C}_{\text{BERT}}$	$(p+k)h+n_{\text{BERT}}$
GRTE (N-N)								
Type I	$n\times n$	$n\times n$	$n\times h$	$n\times n$	$n\times h$	$h\times k$	$\mathcal{O}(n^2(2h+k)+nhk)$	$(n+k)h$
Type II	$n\times n$	$n\times p$	$p\times h$	$n\times n$	$n\times h$	$h\times k$	$\mathcal{O}(n^2(h+k)+nh(k+p))$	$(p+k)h$
Type III	$n\times n$	$n\times p$	$p\times h$	$n\times n$	$n\times h$	$h\times k$	$\mathcal{O}(n^2(h+k)+nh(k+p))$	$(p+k)h$
Type IV	$n\times n$	$n\times p$	$p\times h$	$n\times n$	$n\times h$	$h\times k$	$\mathcal{O}(n^2(h+k)+nh(k+p))+\mathcal{C}_{\text{BERT}}$	$(p+k)h+n_{\text{BERT}}$
GRTE (X-N)								
Type I	$n\times m$	$m\times m$	$m\times h$	$n\times n$	$n\times h$	$h\times k$	$\mathcal{O}(m^2h+n^2k+nh(m+k))$	$(m+k)h$
Type II	$n\times m$	$m\times m$	$m\times h$	$n\times n$	$n\times h$	$h\times k$	$\mathcal{O}(m^2h+n^2k+nh(m+k))$	$(m+k)h$
Type III	$n\times m$	$m\times m$	$m\times h$	$n\times n$	$n\times h$	$h\times k$	$\mathcal{O}(m^2h+n^2k+nh(m+k))$	$(m+k)h$
Type IV	$n\times m$	$m\times m$	$m\times h$	$n\times n$	$n\times h$	$h\times k$	$\mathcal{O}(m^2h+n^2k+nh(m+k))+\mathcal{C}_{\text{BERT}}$	$(m+k)h+n_{\text{BERT}}$

3.3.5 Inductive Inference

Like in Hete-GCN [104], the flexibility of our model enables us to use different word embeddings available at the appropriate intermediate layers to make inductive inferences on unseen documents. For instance, we can store word embedding at the $\mathbf{F}$ layer for the $\mathbf{F}-\mathbf{X}$ path. To make an inductive inference, document embeddings for unseen documents can be obtained by simply averaging word embeddings or using more sophisticated algorithms that produce document embeddings from the word embeddings within them [5, 109].

3.3.6 Derivation of TextGCN from GRTE

TextGCN equations [92] can be obtained using the proposed GCN layers in Section 3.3.2 as a special case of our model, like in Hete-GCN. The final equation of the TextGCN is given as follows (in our notation):

$$\mathbf{H} = \text{ReLU}(\mathbf{A}\mathbf{I}_{(n+m)}\mathbf{W}^{(0)}), \tag{3.12}$$

$$\mathbf{Z} = \textsc{Softmax}(\mathbf{A}\mathbf{H}\mathbf{W}^{(1)}), \tag{3.13}$$

and the cross-entropy error over all labeled documents:

$$\mathcal{L} = -\sum_{i \in \mathcal{Y}_D} \sum_{j=1}^{k} \mathbf{Y}(i,j) \cdot \ln \mathbf{Z}(i,j). \tag{3.14}$$

Let us define $\mathbf{W}^{(0)}$ in Eq. (3.12) as follows;

$$\mathbf{W}^{(0)} = \begin{bmatrix} \mathbf{W}_u^{(0)} \\ \mathbf{W}_v^{(0)} \end{bmatrix}, \mathbf{W}^{(0)} \in \mathbb{R}^{(n+m)\times d_0}, \tag{3.15}$$

where d_0 is a selected dimension of the input features. Now if we simplify Eq. (3.12) this yields;

$$\begin{aligned} \begin{bmatrix} \mathbf{U}^{(1)} \\ \mathbf{V}^{(1)} \end{bmatrix} &= \text{ReLU}\left(\mathbf{A}\mathbf{I}_{(n+m)}\mathbf{W}^{(0)}\right), \\ &= \text{ReLU}\left(\begin{bmatrix} \mathbf{F}_{m\times m} & \mathbf{X}^{\top}_{m\times n} \\ \mathbf{X}_{n\times m} & \mathbf{I}_n \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_n \end{bmatrix} \begin{bmatrix} \mathbf{W}_u^{(0)} \\ \mathbf{W}_v^{(0)} \end{bmatrix}\right), \\ &= \text{ReLU}\left(\begin{bmatrix} \mathbf{F}\mathbf{I}_m\mathbf{W}_u^{(0)} + \mathbf{X}^{\top}\mathbf{I}_n\mathbf{W}_v^{(0)} \\ \mathbf{X}\mathbf{I}_m\mathbf{W}_u^{(0)} + \mathbf{I}_n\mathbf{W}_v^{(0)} \end{bmatrix}\right). \end{aligned} \tag{3.16}$$

Then, we can obtain the word and document embedding matrices as follows;

$$\mathbf{U}^{(1)} = \text{ReLU}\left(\mathbf{F}\mathbf{I}_m\mathbf{W}_u^{(0)} + \mathbf{X}^{\top}\mathbf{I}_n\mathbf{W}_v^{(0)}\right), \tag{3.17}$$

$$\mathbf{V}^{(1)} = \text{ReLU}\left(\mathbf{X}\mathbf{I}_m\mathbf{W}_u^{(0)} + \mathbf{I}_n\mathbf{W}_v^{(0)}\right). \tag{3.18}$$

Now to obtain Eq. (3.17), choose $\mathbf{N} = \mathbf{I}_n, \mathbf{U}^{(0)} = \mathbf{I}_m, \mathbf{V}^{(0)} = \mathbf{I}_n$. Now first pass $\mathbf{U}^{(0)}$ to both $\mathbf{F}-$ReLU GCN and $\mathbf{TX}-$ReLU GCN layers with the inputs above and sum them up, we obtain following equations:

$$\mathbf{U}_F^{(1)} = \text{ReLU}\left(\mathbf{F}\mathbf{I}_m\mathbf{W}_F^{(0)}\right), \tag{3.19}$$

$$\mathbf{U}_{TX}^{(1)} = \text{ReLU}\left(\mathbf{X}^{\top}\mathbf{I}_n\mathbf{W}_{TX}^{(0)}\right), \tag{3.20}$$

$$\mathbf{U}^{(1)} = \mathbf{U}_{F}^{(1)} + \mathbf{U}^{(1)}. \tag{3.21}$$

Now we can see that $\mathbf{U}^{(1)}$ in Eq. (3.17), is simply equal to summation of $\mathbf{U}_{F}^{(1)}$, $\mathbf{U}_{TX}^{(1)}$. Similarly we can also prove that for $\mathbf{V}^{(1)}$ in Eq. (3.18). We show that these embeddings can be retrieved using individual GRTE layers without using the whole adjacency matrix in Eq. (3.2). Therefore, our GCN block is sufficient to obtain the exact results of the TextGCN in a simplified manner.

3.4 Experiments

This section details the experimental setup and evaluation of our proposed GRTE model. We begin by describing the datasets used in our experiments, including their characteristics and preprocessing steps, in Section 3.4.1. Next, we outline the baseline models selected for comparison, providing a diverse range of state-of-the-art methods across different architectures, as explained in Section 3.4.2. Lastly, we discuss the hyperparameter tuning process, including the range and scale of the key parameters explored to ensure optimal performance, in Section 3.4.3.

3.4.1 Datasets

We ran our experiments on four benchmark corpora. Below, we provide brief descriptions of each dataset:

- **Ohsumed** [110]: This dataset contains 13,929 distinct abstracts on cardiovascular illnesses from the first 20,000 papers in 1991. Each document is linked to one or more of the 23 disease categories. For our experiments, we focus on single-label text classification, retaining only 7,400 documents with a single category. The training set consists of 3,357 documents, while the test set has 4,043.

- **Reuters-21578** [111]: This dataset is divided into two subsets: R8 and R52 (all-terms version).
  - **R8**: Contains $5,485$ training documents and $2,189$ test documents across 8 categories.
  - **R52**: Includes $6,532$ training documents and $2,568$ test documents, classified into 52 categories.
- **Movie Review (MR)** [112]: This dataset consists of single-sentence movie reviews. It contains $5,331$ positive reviews and $5,331$ negative reviews, making it a balanced binary classification dataset.
- **20NG** [113]: The 20 Newsgroups dataset contains $18,846$ documents categorized into 20 different classes. The training set consists of $11,314$ documents, while the test set has $7,532$.

Further details for these datasets are provided in Table 3.4.

Table 3.4: Statistics of words, documents, classes, average and max length of the sequences for the datasets used in our experiments.

	MR	**R8**	**R52**	**Ohsumed**	**20NG**
Words	18,764	7,688	8,892	14,157	42,757
Documents	10,662	7,674	9,100	7,400	18,846
Train Documents	7,108	5,485	6,532	3,357	11,314
Test Documents	3,554	2,189	2,568	4,043	7,532
Classes	2	8	52	23	20
Average Length	20.39	65.72	69.82	135.82	221.26
Max Length	39	520	612	476	819

3.4.2 Baseline Models

We consider a wide range of existing works and state-of-the-art models as our baselines. We compared the results of our model with the models given below.

- **CNN** [36]: Convolutional Neural Network for text classification. For the input layer, both randomly initialized word embedding version (CNN-rand)

and pre-trained GloVe [9] word embedding version (CNN-non-static) are used.

- **LSTM** [47]: Long short-term memory as given in [47], where the last hidden state vector is used to represent the documents. The randomly initialized version (LSTM) and the pre-trained GloVe word embeddings version (LSTM-pretrain) are considered.

- **fastText** [15]: Average of pre-trained GloVe word embeddings used as document embeddings, then model feeds these document embeddings to the linear classifier.

- **BERT** [48]: Transformer-based language model. We used the fine-tuned results given in [73].

- **RoBERTa** [106]: A variant of BERT. We used the fine-tuned results reported in [73].

- **TextGCN** [92]: Transductive graph-based model where the vanilla GCN is used as the GNN layer and the adjacency matrix is defined by Eq. (3.2).

- **TensorGCN** [84]: Semantic, syntactic, and sequential contextual information resides in the constructed text graph tensor. Two propagation learnings are performed on the text graph tensor: intra-graph and inter-graph propagation.

- **TG-Transformer** [79]: They propose a text graph sampling method that enables subgraph mini-batch training to scale to large-sized corpora. TG-Transformer utilizes a transformer to aggregate information in a subgraph batch with two proposed graph structural encodings.

- **BertGCN** [73]: Combination of TextGCN and BERT model.

- **RoBERTaGCN** [73]: Variant of a BertGCN model where BERT is replaced with RoBERTa.

- **Hete-GCN** [104]: Developed on top of TextGCN to remove redundant matrix multiplications.

3.4.3 Hyperparameter Tuning

We conduct an extensive hyperparameter search to ensure optimal performance for each configuration of paths and types in the proposed GRTE model. The hyperparameter tuning process involves varying key parameters across a defined range, using different steps or scales depending on the parameter type. The complete list of hyperparameters, along with their respective minimum and maximum values, as well as the tuning step or scale, is provided in Table 3.5.

The hyperparameters tuned during training include:

- **Batch size**: We experiment with batch sizes ranging from 32 to 128 in increments of 32. Larger batch sizes can stabilize gradient estimates but may require more memory, while smaller sizes allow more frequent updates.
- **Dropout**: Dropout is a critical regularization technique used to prevent overfitting. We vary the dropout rate between 0.0 and 1.0 in increments of 0.05 to explore different levels of regularization.
- **BERT learning rate**: Since BERT-based models require careful fine-tuning, we search for the optimal learning rate in the logarithmic scale, ranging from 1×10^{-4} to 1×10^{-2}. Fine-tuning BERT with an inappropriate learning rate can lead to either underfitting or catastrophic forgetting of pre-trained knowledge.
- **GCN learning rate**: The learning rate for GCN layers is crucial for ensuring effective graph-based representation learning. We tune it logarithmically over the range 1×10^{-6} to 1×10^{-4}.
- $\boldsymbol{\lambda}$ (*trade-off parameter*): This hyperparameter controls the balance between different loss terms or objectives in the model. We vary $\boldsymbol{\lambda}$ from 0.0 to 1.0 in increments of 0.05 to examine how different levels of trade-off affect performance.

To streamline the tuning process and ensure reproducibility, we use an automated hyperparameter sweep approach, employing random search and Bayesian optimization methods to explore the search space efficiently. This approach allows us to cover a broad range of hyperparameters without manually iterating over each possible combination. Additionally, we use early stopping criteria based on validation performance to avoid unnecessary training epochs and reduce computational cost.

Table 3.5: Hyperparameters tuned during training.

	Min	**Max**	**Step/Scale**
Batch_size	32	128	32
Dropout	0.0	1.0	0.05
BERT_lr	1e-4	1e-2	log
GCN_lr	1e-6	1e-4	log
$\boldsymbol{\lambda}$	0.0	1.0	0.05

3.5 Results & Discussions

In this section, we present the experimental results and discuss the performance of our proposed GRTE models. In Section 3.5.1, we begin by comparing the main results of our models with various state-of-the-art baselines across multiple datasets. The results, including both accuracy and weighted-F1 scores, are analyzed to demonstrate the generalization capability of our models. Additionally, in Section 3.5.2, we examine the impact of the interpolation parameter λ on the final predictions, offering insights into the contribution of GCN components in enhancing classification accuracy. Finally, we explore the relationship between training duration and model performance using time versus accuracy plots in Section 3.5.3. These analyses demonstrate that GRTE offers an optimal balance between accuracy and computational speed.

3.5.1 Main Results

The results of all our configurations and baselines are tabulated in Table 3.6. We run our proposed models with 10 different seeds. The means and standard deviations are reported in Table 3.6. The results of the Hete-GCN, TextGCN, and BertGCN are directly retrieved from the reported values of the corresponding works [73, 92, 104], and are denoted with *. We run a two-tailed t-test [114] to check for statistical significance. With 99% confidence, obtained results in Table 3.6 are statistically significant with p-values less than 0.01. Existing works did not report the WEIGHTED-F1 scores for their models. Observing whether the model can generalize to all classes is crucial for the multiclass classification task. From Table 3.6, our models obtain good WEIGHTED-F1 scores on top of ACCURACY, indicating that our models do not focus on a single class with the most support.

3.5.2 Importance of the Interpolation Parameter

To understand the importance of the GCN model on final layer predictions, we inspect the interpolation parameter λ. Type III $\mathbf{N}-\mathbf{N}$ path achieves 98.72 mean accuracy result with $\lambda = 0.70$ value and Type III $\mathbf{X}-\mathbf{N}$ path achieves 73.17 mean accuracy result with $\lambda = 0.75$. To deepen our understanding of the significance of each component in the interpolation process, we systematically explore the hyperparameter λ across a range from 0 to 1, incrementing by a step size of 0.05 on the 20NG dataset with Type III. The outcomes of this comprehensive exploration are elucidated in the visual representation provided in Fig. 3.3.

Since the prediction results from the GCN contribute to around 70% of the final prediction, the GCN component significantly affects the prediction results. Therefore, GCN has a significant role in increasing the performance of BERT and capturing the global information residing within documents.

Table 3.6: Results for the Mean and Standard Deviation for 10 seed runs for each model on datasets R8, R52, Ohsumed, and MR. w-F1 is the abbreviation for weighted F1. Results are evaluated with a t-test, and they satisfy 99% confidence interval ($p < 0.01$). The best accuracy results for each dataset are emboldened, and the runner-ups are underlined.

	R8		**R52**		**Ohsumed**		**MR**		**20NG**	
Method	w-F1	Accuracy	w-F1	Accuracy	w-F1	Accuracy	w-F1	Accuracy	w-F1	Accuracy
CNN-rand*	-	94.02 ± 5e-3	-	85.37 ± 4e-3	-	43.87 ± 1e-2	-	74.98 ± 7e-3	-	76.93 ± 6e-3
CNN-non-static*	-	95.71 ± 5e-3	-	87.59 ± 4e-3	-	58.44 ± 1e-2	-	77.75 ± 7e-2	-	82.15 ± 5e-3
LSTM*	-	93.68 ± 8e-3	-	85.54 ± 1e-2	-	41.13 ± 1e-2	-	75.06 ± 4e-3	-	65.71 ± 2e-3
LSTM (pre-train)*	-	96.06 ± 1e-3	-	90.48 ± 8e-3	-	51.10 ± 1e-2	-	77.33 ± 8e-3	-	75.43 ± 2e-3
fastText*	-	86.04 ± 2e-3	-	71.55 ± 4e-3	-	14.56 ± 1e-5	-	72.17 ± 1e-2	-	79.38 ± 3e-3
BERT*	-	97.80	-	96.40	-	70.50	-	85.70	-	85.30
RoBERTa	98.33 ± 2e-4	98.22 ± 1e-5	95.98 ± 3e-3	96.22 ± 1e-4	70.17 ± 1e-3	70.72 ± 1e-5	89.33 ± 1e-5	89.33 ± 2e-4	83.87 ± 2e-4	84.41 ± 4e-3
TextGCN*	-	97.07 ± 1e-3	-	93.56 ± 2e-3	-	68.36 ± 5e-3	-	76.74 ± 2e-3	-	86.34 ± 9e-4
TensorGCN*	-	98.04 ± 8e-4	-	95.05 ± 1e-3	-	70.11 ± 2e-3	-	77.91 ± 7e-4	-	87.74 ± 5e-4
TG-Transformer*	-	98.10 ± 1e-1	-	95.20 ± 2e-1	-	70.40 ± 4e-1	-	-	-	-
BertGCN*	-	98.10	-	**96.60**	-	72.80	-	86.00	-	89.30
RoBERTaGCN*	-	98.20	-	96.10	-	72.80	-	89.70	-	89.50
HeteGCN (F-X)	-	97.24 ± 5e-1	-	94.35 ± 3e-1	-	68.11 ± 2e-1	-	76.71 ± 3e-1	-	86.59 ± 1e-1
HeteGCN (TX-X)	-	96.96 ± 2e-1	-	92.87 ± 6e-1	-	66.14 ± 5e-1	-	75.21 ± 4e-1	-	83.71 ± 1e-2
HeteGCN (N-N)	-	94.39 ± 2e-1	-	91.93 ± 1e-3	-	62.46 ± 2e-1	-	74.23 ± 8e-1	-	87.15 ± 1e-4
HeteGCN (X-N)	-	94.39 ± 1e-1	-	91.93 ± 1e-1	-	62.46 ± 2e-1	-	74.23 ± 8e-1		75.99 ± 1e-1
GRTE (F-X)										
Type I	96.51 ± 2e-3	96.53 ± 3e-3	85.80 ± 1e-2	87.64 ± 1e-2	63.94 ± 7e-3	64.32 ± 6e-3	76.44 ± 4e-3	76.45 ± 4e-3	86.54 ± 2e-3	86.67 ± 4e-3
Type II	94.92 ± 1e-2	95.07 ± 1e-2	89.90 ± 1e-2	90.91 ± 6e-3	62.84 ± 2e-3	63.14 ± 2e-3	76.25 ± 3e-3	76.25 ± 4e-3	86.67 ± 2e-3	86.59 ± 4e-3
Type III	98.24 ± 2e-3	98.26 ± 3e-4	95.99 ± 2e-4	96.20 ± 2e-4	**73.25** ± 1e-3	**73.12** ± 1e-3	89.77 ± 4e-4	89.77 ± 4e-4	**89.76** ± 2e-3	**89.84** ± 4e-3
Type IV	98.22 ± 1e-4	98.45 ± 2e-4	95.27 ± 1e-4	95.91 ± 3e-3	72.55 ± 1e-4	72.89 ± 6e-3	89.59 ± 6e-3	89.59 3e-3	87.21 ± 2e-4	87.28 ± 2e-3
GRTE (TX-X)										
Type I	95.62 ± 7e-3	95.70 ± 3e-3	83.13 ± 1e-3	85.56 ± 1e-3	58.85 ± 2e-2	60.26 ± 2e-3	76.86 ± 4e-3	76.87 ± 4e-3	86.55 ± 4e-3	85.61 ± 2e-3
Type II	96.47 ± 4e-3	96.46 ± 4e-3	86.97 ± 6e-3	88.64 ± 7e-3	60.79 ± 1e-2	61.50 ± 2e-2	86.90 ± 1e-3	86.90 ± 1e-3	88.46 ± 1e-3	88.52 ± 2e-3
Type III	98.32 ± 3e-4	98.33 ± 3e-4	95.98 ± 2e-4	96.20 ± 2e-4	71.08 ± 5e-4	71.12 ± 5e-4	**89.92** ± 7e-4	**89.92** ± 7e-4	89.40 ± 4e-3	89.46 ± 2e-3
Type IV	98.22 ± 1e-4	98.45 ± 1e-4	95.27 ± 1e-4	95.91 ± 6e-3	71.22 ± 3e-3	71.9 ± 3e-3	89.90 ± 2e-2	89.90 ± 1e-3	86.40 ± 5e-3	86.46 ± 6e-3
GRTE (N-N)										
Type I	92.13 ± 5e-3	92.14 ± 5e-3	90.07 ± 1e-3	89.74 ± 2e-3	57.36 ± 4e-3	58.08 ± 3e-3	74.06 ± 3e-3	74.06 ± 3e-3	79.84 ± 4e-3	79.97 ± 4e-3
Type II	92.74 ± 6e-3	92.76 ± 5e-3	85.77 ± 2e-3	87.21 ± 1e-3	59.56 ± 3e-3	60.30 ± 2e-3	77.82 ± 6e-3	77.83 ± 6e-3	78.18 ± 4e-3	78.31 ± 1e-3
Type III	**98.71** ± 7e-3	**98.72** ± 3e-4	**96.02** ± 2e-3	96.25 ± 2e-3	71.59 ± 1e-3	71.70 ± 1e-3	89.69 ± 1e-4	89.69 ± 1e-4	87.47 ± 1e-3	87.52 ± 1e-3
Type IV	98.07 ± 1e-3	98.13 ± 1e-4	95.13 ± 3e-3	95.41 ± 6e-3	69.60 ± 2e-3	69.70 ± 1e-4	89.73 ± 1e-2	89.73 ± 6e-3	83.17 ± 6e-3	83.43 ± 3e-3
GRTE (X-N)										
Type I	91.85 ± 7e-3	91.96 ± 7e-3	89.07 ± 2e-3	89.70 ± 3e-3	56.11 ± 6e-3	56.32 ± 6e-3	73.65 ± 3e-3	73.65 ± 3e-3	79.49 ± 4e-3	79.62 ± 4e-3
Type II	91.61 ± 9e-3	92.76 ± 6e-3	89.77 ± 1e-3	89.78 ± 2e-3	56.07 ± 3e-3	56.27 ± 3e-3	74.34 ± 4e-3	74.35 ± 4e-3	79.71 ± 1e-3	79.83 ± 2e-3
Type III	98.26 ± 1e-3	98.26 ± 1e-4	95.98 ± 6e-4	96.20 ± 6e-4	71.45 ± 1e-3	71.55 ± 1e-3	89.72 ± 1e-3	89.72 ± 1e-3	88.29 ± 4e-3	88.34 ± 2e-3
Type IV	98.18 ± 2e-3	98.26 ± 3e-3	95.27 ± 3e-3	95.91 ± 6e-3	71.52 ± 6e-3	71.83 ± 3e-3	89.73 ± 4e-3	89.73 ± 6e-3	84.92 ± 2e-2	84.92 ± 3e-3

3.5.3 Traning Duration vs Performance

We propose different training regimes to see the relation between model performance and speed. The average per epoch time of each dataset for the $\mathbf{N-N}$ path is given in Table 3.7, where we also report the results of our runs of TextGCN, Hete-GCN, and RoBERTaGCN. Our proposed GRTE Type III model achieves the best accuracy result with the $\mathbf{N-N}$ path in the R8 dataset, which is also up to a hundred times faster than the RoBERTaGCN. Also, our Type I model outperforms Hete-GCN in terms of accuracy on the MR dataset for the $\mathbf{TX-X}$ path.

For better visual understanding, we also provide training duration analysis and model performances in Fig. 3.4, combining results from Table 3.7 for our

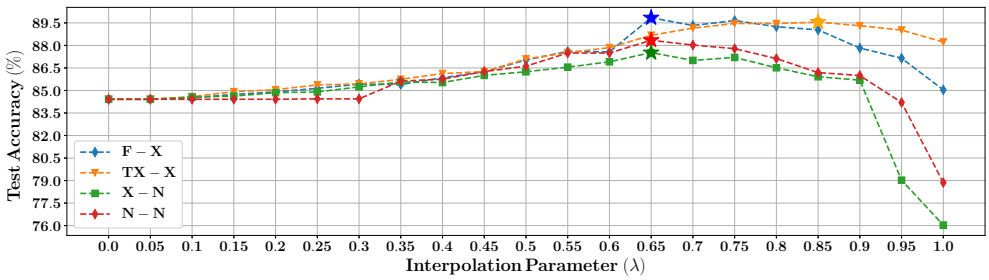


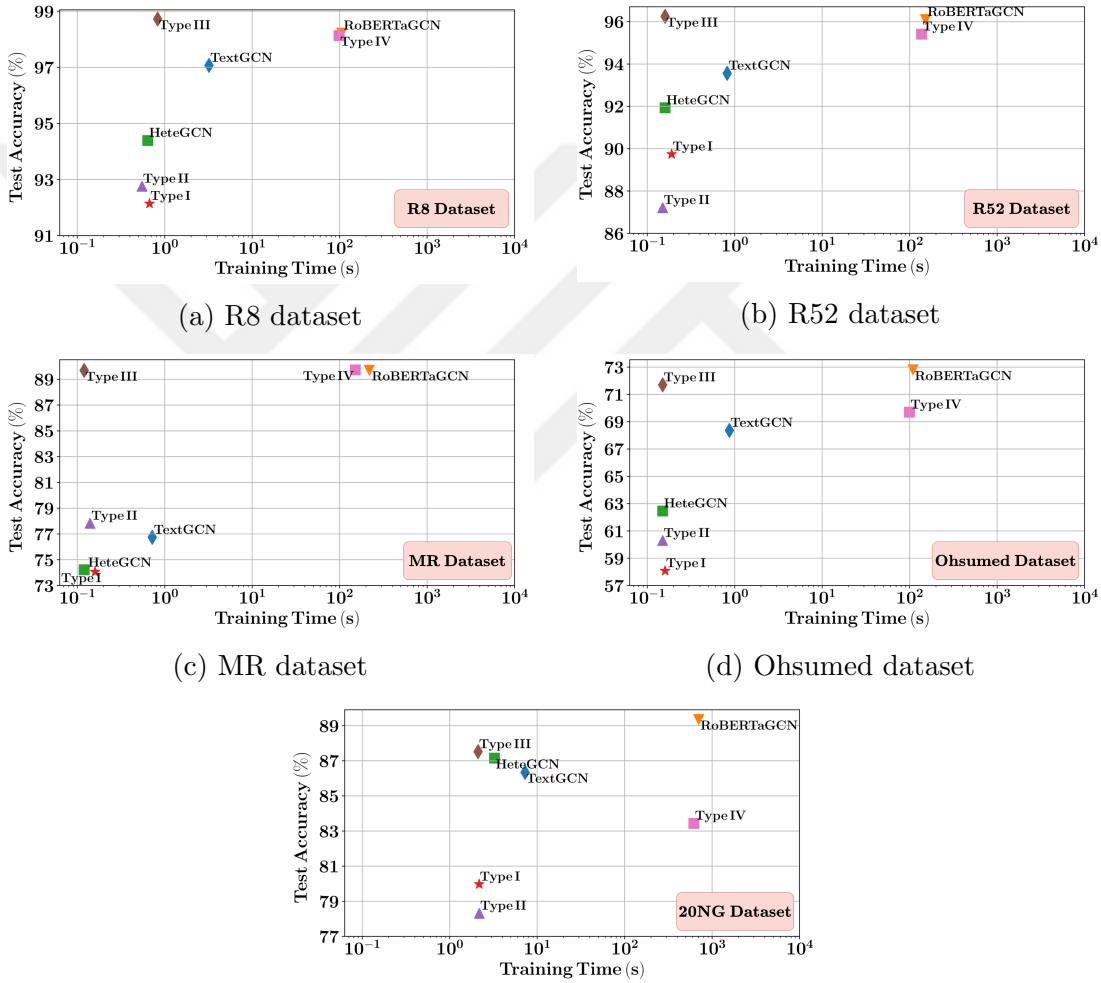
Figure 3.3: Accuracy of GRTE with varying λ on 20NG test set with four different paths for the Type III, where $\lambda = 0$ corresponds to the fine-tuned RoBERTa baseline. Markers with stars represent the best result.

Table 3.7: Accuracy and Average per epoch Duration (secs.) result for speed comparison. We ran each model for 100 epochs and obtained the average results. Best accuracy results and fastest models are emboldened individually.

Method	R8		R52		Ohsumed		MR		20NG	
	Duration (s)	Accuracy (%)	Duration (s)	Accuracy (%)	Duration (s)	Accuracy (%)	Duration (s)	Accuracy (%)	Duration (s)	Accuracy (%)
TextGCN*	3.21	97.07 ± 1e-3	0.82	93.56 ± 2e-3	0.87	68.36 ± 5e-3	0.72	76.74 ± 2e-3	7.26	86.34 ± 9e-4
RoBERTaGCN*	105.63	98.20	152.07	96.10	109.34	**72.80**	218.32	89.70	702.33	**89.50**
HeteGCN (N-N)	0.64	94.39 ± 2e-1	0.16	91.93 ± 1e-3	**0.15**	62.46 ± 2e-1	**0.12**	74.23 ± 8e-1	3.25	87.15 ± 1e-4
Type I (N-N)	0.67	92.14 ± 5e-3	0.19	89.74 ± 2e-3	0.16	58.08 ± 3e-3	0.16	74.06 ± 3e-3	2.17	79.97 ± 4e-3
Type II (N-N)	**0.55**	92.76 ± 5e-3	**0.15**	87.21 ± 1e-3	**0.15**	60.30 ± 2e-3	0.14	77.83 ± 6e-3	2.18	78.31 ± 1e-3
Type III (N-N)	0.83	**98.72** ± 3e-4	0.16	**96.25** ± 2e-3	**0.15**	71.70 ± 1e-3	**0.12**	89.69 ± 7e-4	**2.11**	87.52 ± 1e-3
Type IV (N-N)	98.43	98.13 ± 1e-4	136.72	95.41 ± 6e-4	99.37	69.70 ± 1e-4	150.96	**89.73** ± 1e-2	618.89	83.43 ± 3e-3

proposed models and main state-of-the-art methods.

Finally, upon examining Fig. 3.4, one can conclude that the proposed GRTE mostly populates the top left corners in time versus accuracy plots, which is the desired location since we look for better accuracy at higher speeds. GRTE mostly outperforms or at least performs on par BertGCN and RoBERTaGCN, which are state-of-the-art in terms of accuracy, while offering a remarkable increase ($\sim 100\times$) in terms of computational efficiency. GRTE also significantly outperforms HeteGCN regarding model performance while slightly improving computational efficiency. Thus, our proposed GRTE offers the ideal combination of model performance and computational efficiency.



(a) R8 dataset

(b) R52 dataset

(c) MR dataset

(d) Ohsumed dataset

(e) 20NG dataset

Figure 3.4: All four variants (Types I, II, III, and IV) of the proposed GRTE with $\mathbf{N}-\mathbf{N}$ path along with the baselines TextGCN, Hete-GCN and RoBERTaGCN. The top left corner is the desired location since we look for the best accuracy at higher speeds.

Chapter 4

Text-RGNNs: Relational Modeling For Heterogeneous Text Graphs

In this chapter, we present our approach to addressing the limitations of existing GNN-based methods that oversimplify the relational structure in heterogeneous graphs by treating them as homogeneous. While homogeneous GNNs employ a single-weight matrix for all node and edge types, this simplification overlooks the diverse and complex interactions between different types of nodes, such as word and document nodes in text graphs. To overcome this issue, we propose a relational modeling framework that explicitly accounts for these heterogeneous relationships, enabling a more fine-grained and accurate representation of the graph structure. The following sections detail our method, its underlying mechanisms, and its impact on improving text classification performance by providing relational treatment for heterogeneous text graphs.

4.1 Problem Definition

In the context of semi-supervised text classification, following the convention in [115], we have a list of documents as a corpus for each dataset, represented as $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$, with a cardinality of $|\mathcal{D}| = n$. Each document d_i has a corresponding one-hot encoded label $\mathbf{y}_i \in \mathbb{R}^K$, where K is the total number of classes. We split $\mathcal{D}$ into training, validation, and test sets, denoted as, $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, and $\mathcal{D}_{\text{test}}$. Further, to evaluate the semi-supervised nature of our method, we divide the training set into labeled and unlabeled, denoted as $\mathcal{D}^{p}_{\text{train}}$ and $\widehat{\mathcal{D}}^{100-p}_{\text{train}}$, respectively, where $p \in \{1, 5, 10, 20, 100\}$ is the labeled training percentage. Remark that, $\mathcal{D}^{p}_{\text{train}} \cup \widehat{\mathcal{D}}^{100-p}_{\text{train}} = \mathcal{D}_{\text{train}}$ and $\mathcal{D}^{p}_{\text{train}} \cap \widehat{\mathcal{D}}^{100-p}_{\text{train}} = \emptyset$.

4.2 Related Work

This section reviews the foundational concepts of GNNs and their application to homogeneous and heterogeneous graphs. Section 4.2.1 introduces the standard framework for homogeneous GNNs, where all nodes and edges are treated uniformly, and the node updates are performed using standard aggregation and combination functions. Building on this, Section 4.2.2 extends the discussion to heterogeneous GNNs, where multiple types of nodes and relations exist. We present a relational modeling approach incorporating relation-specific aggregations to capture the diverse interactions in complex graphs, which is critical for tasks involving heterogeneous data.

4.2.1 Node Updates for Homogeneous GNNs

For homogeneous GNNs that have single node and relation type, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph where $\mathcal{V}$ denotes set of vertices (nodes) and $\mathcal{E}$ denotes set of edges between those nodes. Let each node in the GNN, represented by a hidden vector $\mathbf{h}_v^{(l)} \in \mathbb{R}^{d^{(l)}}$ which has $d^{(l)}$ embedding dimension for the v node at l-th layer of the

GNN. Formally, the $(l+1)$-th layer of GNN uses the neighborhood of v which is denoted as $\mathcal{N}(v)$ to update its hidden representation at l-th layer [116]:

$$\mathbf{a}_v^{(l+1)} = \text{AGGREGATE}^{(l+1)}\left(\left\{\mathbf{h}_u^{(l)} : u \in \mathcal{N}(v)\right\}\right), \tag{4.1}$$

$$\mathbf{h}_v^{(l+1)} = \text{COMBINE}^{(l+1)}\left(\mathbf{h}_v^{(l)}, \mathbf{a}_v^{(k)}\right), \tag{4.2}$$

where $\mathbf{h}_v^{(l+1)}$ is the hidden vector and $\mathbf{a}_v^{(l+1)}$ is the aggregated neighborhood node information of node v at the $(l+1)$-th layer. The *aggregate* operator can be chosen from the *sum*, *max* or *mean* functions. Similarly, the *combine* operator can be replaced with the *sum*, *concatenation*, or any linear transform.

4.2.2 Node Updates for Heterogeneous GNNs

To further generalize to heterogeneous GNNs that have intrinsic relations, we follow the notation in R-GCNs [117] to denote the relational graph as $G = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ with nodes $v \in \mathcal{V}$, edges $(v, r, u) \in \mathcal{E}$, and relation types $r \in \mathcal{R}$. We define our proposed generalized aggregator depending on the relation types as follows:

$$\mathbf{a}_{v,r}^{(l+1)} = \text{AGGREGATE}^{(l+1)}\left(\left\{\mathbf{h}_u^{(l)} : u \in \mathcal{N}_v^r\right\}\right), \tag{4.3}$$

$$\mathbf{a}_v^{(l+1)} = \text{AGGREGATE}^{(l+1)}\left(\left\{\mathbf{a}_{v,r}^{(l+1)} : r \in \mathcal{R}\right\}\right), \tag{4.4}$$

where $\mathbf{a}_{v,r}^{(l+1)}$ is the relation-type-specific node neighborhood aggregation and $\mathbf{a}_v^{(l+1)}$ is the aggregation of all relations for node v at the $(l+1)$-th layer. We use $\mathcal{N}_v^r$ to denote the set of indices of the neighbor of node v under relation $r \in \mathcal{R}$. Our proposed aggregation functions in Eqs. (4.3) and (4.4) can be generalized to any existing homogeneous GNN in the literature to denote it in a heterogeneous format.

4.3 Methodology

In this section, we detail the methodology employed for constructing and training our proposed Text-RGNNs model. We begin with the embedding construction

process, where we generate node embeddings for both word and document nodes, as explained in Section 4.3.1. Next, we describe the heterogeneous graph construction method, including how different types of node relations are established using k-nearest neighbor graphs and TF-IDF matrices, in Section 4.3.2. Following that, we introduce the heterogeneous convolutional layers used in our model, derived from GraphSAGE and Graph Attention Networks (GAT), and explain their adaptation for relational modeling in Section 4.3.3. We then present a complexity analysis, outlining the computational cost associated with graph construction and model inference, in Section 4.3.4. Finally, the overall architecture of the proposed Text-RGNNs, including fine-tuning and graph construction stages, is illustrated in Fig. 4.1.

4.3.1 Embedding Construction

We generate nodes from the unique words and documents in $\mathcal{D}$ to obtain $\mathcal{V}$, which includes two types of nodes, namely, document and word nodes. We initialize node embeddings as follows: We first obtain a vocabulary from $\mathcal{D}^{100}_{\text{train}}$ and then train Word2Vec [10] to assign embeddings to word nodes. Since $\mathcal{D}_{\text{train}} \subseteq \mathcal{D}$, for out-of-vocabulary nodes, we assign a special embedding vector initialized randomly to ensure they are represented in the embedding space. Subsequently, for each $\mathcal{D}^{p}_{\text{train}}$, we fine-tune a RoBERTa model [106], then use a single forward-pass to assign the document node embeddings with the `[CLS]` tokens.

4.3.2 Heterogeneous Graph Construction

In the context of our problem, we have three relation types: word-word, document-document, and document-word relations. For word-word and document-document relation types, we construct k-nearest neighbors (k-NN) graphs based on the initial embeddings. Specifically, we derive an embedding matrix from initial embeddings for each node type. The number of neighbors, denoted as k, is chosen as a hyperparameter for our model. Using these embeddings,

we compute the pairwise distances between all nodes. For each node, we sort these distances to identify the k closest neighbors, establishing connections based on these proximities. The distances are computed using the Euclidean distance metric. For the document-word relation type, we calculate a term frequency-inverse document frequency (TF-IDF) matrix for each $\mathcal{D}$. The complexity analysis for graph construction is given in Section 4.3.4. The entry-wise definition of the heterogeneous adjacency matrix is given as follows:

$$A_{ij} = \begin{cases} k\text{-NN}_{ij}\text{of Word2Vec}, & \text{if } i, j \text{ are words} \\ k\text{-NN}_{ij}\text{of RoBERTa}, & \text{if } i, j \text{ are documents} \\ \text{tf-idf}(i, j), & \text{if } i \text{ is document, } j \text{ is word} \\ 1, & \text{if } i = j \end{cases} \tag{4.5}$$

4.3.3 Heterogeneous Convolutional Layers

In this work, we use the convolution layers in GraphSAGE [118] and Graph Attention Network (GAT) [119]. They are initially implemented as homogeneous GNN layers. Therefore, we propose the following derivation to update the representation of any type of node v at l-th layer. Our derivation for relational SAGEConv at l-th layer is as follows:

$$\mathbf{a}_{v,r}^{(l+1)} = \text{MAX}\left(\sigma\left(c_{u,r}\mathbf{W}_r^{(l)}\mathbf{h}_u^{(l)}\right) : u \in \mathcal{N}_v^r\right), \tag{4.6}$$

$$\mathbf{a}_v^{(l+1)} = \text{SUM}\left(\left\{\mathbf{a}_{v,r}^{(l+1)} : r \in \mathcal{R}\right\}\right), \tag{4.7}$$

$$\mathbf{h}_v^{(l+1)} = \mathbf{W}_0^{(l)}\left[\mathbf{a}_v^{(l+1)} \parallel \mathbf{h}_v^{(l)}\right], \tag{4.8}$$

where $\parallel$ is the concatenation operation. The normalization constants $c_{u,r}$ are defined as the entries of the heterogeneous adjacency matrix given in Eq. (4.5). We refer to a non-linear activation function as σ, which we use as a hyperparameter in our experiments to select between LeakyReLU and ReLU. Our derivation for relational GAT with a single layer is as follows:

$$\alpha_{(v,u),r} = \frac{\exp\left(\sigma\left(\mathbf{c}_r^\top[\mathbf{W}_r\mathbf{h}_v \parallel \mathbf{W}_r\mathbf{h}_u]\right)\right)}{\sum\limits_{k\in\mathcal{N}_v^r}\exp\left(\sigma\left(\mathbf{c}_r^\top[\mathbf{W}_r\mathbf{h}_v \parallel \mathbf{W}_r\mathbf{h}_k]\right)\right)}, \tag{4.9}$$

$$\mathbf{a}_{v,r} = \sigma\left(\text{SUM}\left(\{\alpha_{(v,u),r}\mathbf{h}_u : u \in \mathcal{N}_v^r\}\right)\right), \tag{4.10}$$

$$\mathbf{h}_v^{'} = \mathbf{a}_v = \text{SUM}\left(\{\mathbf{a}_{v,r} : r \in \mathcal{R}\}\right), \tag{4.11}$$

where $\mathbf{h}_v^{'}$ is the final hidden representation of the node, $\alpha_{(v,u),r}$ is the relation specific attention coefficient, and $\mathbf{c}_r$ and $\mathbf{W}_r$ are learnable relation specific weights. The adjacency matrix defined in Eq. (4.5) is used for masking operation for the attention coefficients to enable message passing only on the connected nodes.

We denote the total number of GNN layers with L, consequently the final hidden representation of node v is represented with $\hat{\mathbf{y}}_v = \text{softmax}(\mathbf{h}_v^{(L)})$. At the final layer, we map the hidden representation of the feature vector from the previous layer to K, which gives $d^{(L)} = K$. The following cross-entropy loss function is used to evaluate the loss on $\mathcal{D}_{\text{train}}^p$:

$$\mathcal{L} = -\sum_{d \in \mathcal{D}_{\text{train}}^p} \sum_{k=1}^{K} y_{d,k} \cdot \ln(\hat{y}_{d,k}), \tag{4.12}$$

where the individual entries of the $\hat{\mathbf{y}}_d$ are denoted with $\hat{y}_{d,k}$, and $y_{d,k}$ represents the one-hot encoded vector of the true labels for document node d.

The overall architecture of the proposed model, along with the fine-tuning and graph construction stage, is given in Fig. 4.1.

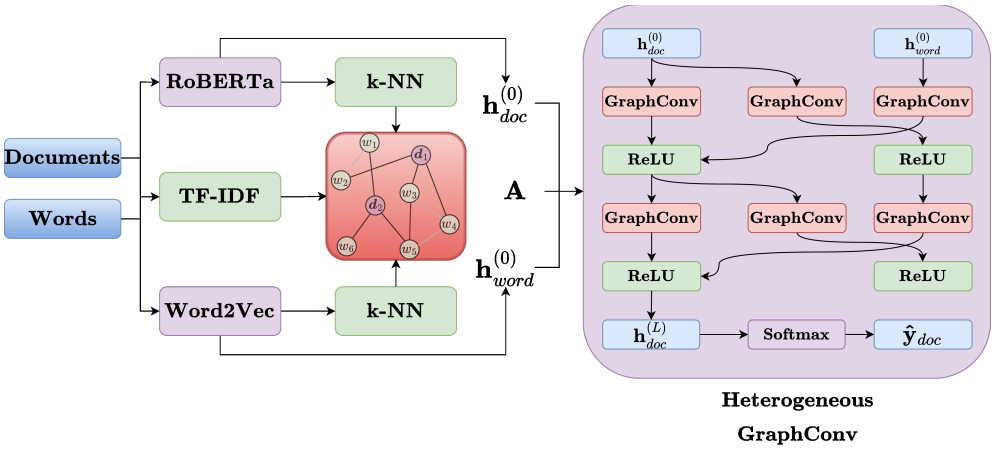


Figure 4.1: The architecture for the proposed Text-RGNNs. The Heterogeneous GraphConv block consists of a 2-layer version of our heterogeneous model, composed of homogeneous graph convolution layers.

4.3.4 Complexity Analysis

Here, we identified the complexity of the graph construction procedure and the computation cost complexity of our model compared with the baseline models.

For the graph construction, let n be the number of unique documents, T be the average length of a document, m be the vocabulary size, and d be the dimensionality of the embeddings. The TF-IDF computation has a complexity of $O(n \cdot T + n \cdot m)$. For k-NN graph construction, calculating pairwise distances has a complexity of $O(n^2 \cdot d)$, and finding the k-nearest neighbors involves $O(n^2 \log n)$. Thus, the overall complexity is $O(n \cdot T + n \cdot m)$ for TF-IDF computation and $O(n^2 \cdot d + n^2 \log n)$ for k-NN graph construction.

4.4 Experiments

This section outlines the experimental setup used to evaluate our proposed approach. We introduce the six benchmark datasets in Section 4.4.1, including their composition and processing details. Following this, in Section 4.4.2, we provide an overview of the baseline models, including supervised and semi-supervised state-of-the-art methods, detailing the specific configurations and training regimes used for fair comparisons. Lastly, Section 4.4.3 describes the hyperparameter tuning process, where we utilize the `WandB` library for automated search.

4.4.1 Datasets

We run our experiments on six benchmark corpora for document classification. Below, we provide a brief description of each dataset:

- **Ohsumed** [110]: This dataset contains 13,929 distinct abstracts on cardiovascular illnesses from the first 20,000 papers in 1991. Each document

is linked to one or more of the 23 disease categories. For our experiments, we focus on single-label text classification, retaining only 7,400 documents with a single category. The training set consists of 3,357 documents, while the test set has 4,043.

- **Reuters-21578** [111]: This dataset is divided into two subsets: R8 and R52 (all-terms version).
  - **R8**: Contains 5,485 training documents and 2,189 test documents across 8 categories.
  - **R52**: Includes 6,532 training documents and 2,568 test documents, classified into 52 categories.
- **Movie Review (MR)** [112]: This dataset consists of single-sentence movie reviews. It contains 5,331 positive reviews and 5,331 negative reviews, making it a balanced binary classification dataset.
- **SST-2**: The Stanford Treebank Sentiment Analysis dataset [120], which consists of sentences labeled with binary sentiment (positive or negative). It is one of the standard datasets in sentiment classification benchmarks.
- **CoLA**: The Corpus of Linguistic Acceptability [121] is a binary classification dataset. It consists of English sentences annotated for linguistic acceptability, used to evaluate models' syntactic understanding.

For **Ohsumed**, **R8**, **R52**, and **MR**, we create a custom package to read and process the raw versions from Google Drive[1]. The **SST-2** and **CoLA** datasets are downloaded from the *HuggingFace* library. Further details for these datasets are provided in Table 4.1.

4.4.2 Baseline Models

As baseline models, we use existing state-of-the-art models in the literature, their brief descriptions are as follows:

[1] https://github.com/ardaaras99/baseline-text-graphs

Table 4.1: Statistics of words, documents, classes, average and max length of the sequences for the datasets.

	MR	R8	R52	Ohsumed	SST-2	CoLA
Total Documents	10,662	7,674	9,100	7,400	68,220	9,592
Train Documents	7,108	5,485	6,532	3,357	67,348	8,550
Test Documents	3,554	2,189	2,568	4,043	872	1,042
Total Number Classes	2	8	52	23	2	2

- **BERT** [48]: A transformer-based model pre-trained using masked language modeling, fine-tuned for text classification tasks.
- **RoBERTa** [106]: An improved version of BERT, trained with larger datasets and optimized training strategies for better performance.
- **TextGCN** [92]: A graph convolutional network model that constructs a graph based on word and document co-occurrence, suitable for semi-supervised text classification.
- **TensorGCN** [84]: A tensor-based extension of TextGCN that models higher-order interactions between words and documents.
- **TG-Transformer** [79]: A hybrid model combining transformer encoders with graph-based representations for capturing both local and global context.
- **RoBERTaGCN** [73]: A model that integrates RoBERTa embeddings with GCNs to enhance performance by incorporating global relational information from graphs.
- **ME-GCN** [122]: A multi-view GCN approach that leverages different views of textual data, such as syntactic and semantic relations, to improve the generalization and robustness of graph-based models for text classification.
- **TextGCL** [123]: A graph contrastive learning model designed to improve the robustness of GCNs in semi-supervised text classification.

- **FewShotTextGCN** [124]: A graph-based few-shot learning model that addresses the challenge of text classification with limited labeled data.

For the baseline models, we start with RoBERTaGCN. We fine-tune a pre-trained RoBERTa model for five epochs and use these weights as initial RoBERTa weights, following the default GitHub configurations[2]. RoBERTaGCN is trained for ten epochs with ten seeds per dataset. For TextGCN, we use a hidden dimension of 200, a learning rate of 0.02, a dropout rate of 0.5, and the Adam optimizer. For ME-GCN[3], we follow the default configurations. For FewShotTextGCN[4], we modify the source code to generate results on benchmark datasets. Each dataset is trained for 1000 epochs, monitoring validation accuracy (Matthews for CoLA) to select the best model and report test accuracy.

4.4.3 Hyperparameter Tuning

We use the sweep tool of `WandB` library for hyperparameter search. We focus specifically on two layer relational GNN variants to avoid the over-smoothing problem [105], where increasing layers causes node representations to become too similar, reducing model performance. Limiting the layers helps capture essential neighborhood information while preserving distinct node features, ensuring better classification accuracy. The list for the rest of the hyperparameters, the range we tune, and the distribution of these ranges are publicly available [5].

4.5 Results & Discussion

This section presents the results and analysis of our proposed Text-RGNNs models. In Section 4.5.1, we provide a detailed comparison of Text-RGNNs against

[2] https://github.com/ZeroRin/BertGCN
[3] https://github.com/usydnlp/ME_GCN
[4] https://github.com/mrvoh/FewShotTextGCN
[5] https://wandb.ai/ardaaras99/text-rgnn-new/overview

existing state-of-the-art models across multiple datasets, focusing on scenarios with varying proportions of labeled training data. To further demonstrate the importance of relational modeling, we conduct an ablation study in Section 4.5.2, where the relational dependencies in the graph are removed, highlighting the resulting performance drop. Lastly, Section 4.5.3 discusses the computational efficiency of Text-RGNNs, showing that our models achieve superior accuracy and offer significant reductions in training time compared to existing methods.

4.5.1 Main Results

We start by presenting results in Table 4.2 for the proposed Text-RGNNs and baselines. Some of the existing works do not provide code for their implementation. Therefore, the retrieved results from the existing works are denoted with an asterisk*. The proposed Text-RGNNs perform better than the state-of-the-art models in almost all benchmark datasets. In real-world problems, a significant portion of data is unlabeled, requiring semi-supervised settings. Therefore, we also evaluated our model's performance against baselines with $\mathcal{D}^{p}_{\text{train}}$ for $p \in \{1, 5, 10, 20\}$.

Upon examining Table 4.2, it can be seen in general that the performance improvements and advantages of our proposed Text-RGNNs are more pronounced as the proportion of the labeled training data becomes small. Our proposed algorithm surpasses the performance of the second-best models by up to 8.49%, 7.71%, 10.61%, 0.65%, 0.47%, and 1.01% nominal increase in accuracy on R8, R52, Ohsumed, MR, SST-2 datasets, and a nominal increase in the Matthews correlation coefficient in the CoLA dataset, respectively.

Based on the results presented in Table 4.2, Text-RGNNs surpass state-of-the-art models not only when all training data labels are accessible but also demonstrate superior performance in scenarios where labeled training data is limited. It incorporates relational modeling into heterogeneous text graphs, leveraging its inherent capacity to extract rich contextual information from the entire data even when the proportion of labeled samples is scarce.

Table 4.2: Results for the Mean of 10 seed runs on Benchmark Datasets. 1%, 5%, 10%, 20%, and 100% are the proportions of labeled training data. The Matthews Correlation Coefficient metric is reported for the CoLA dataset. The accuracy metric is reported for the rest. Results are evaluated with a t-test, and they satisfy 99% confidence interval ($p < 0.01$). The best scores are emboldened.

	R8					**R52**					**Ohsumed**				
Method	**1%**	**5%**	**10%**	**20%**	**100%**	**1%**	**5%**	**10%**	**20%**	**100%**	**1%**	**5%**	**10%**	**20%**	**100%**
TensorGCN*	-	-	-	-	98.04	-	-	-	-	95.05	-	-	-	-	70.11
TG-Transformer*	-	-	-	-	98.10	-	-	-	-	95.20	-	-	-	-	70.40
BERT	49.61	82.59	88.99	93.88	97.80	42.21	68.26	74.65	83.96	96.20	8.43	17.17	18.77	37.03	70.50
TextGCN	84.78	84.19	85.93	95.71	97.07	67.02	67.45	67.79	67.29	93.56	19.88	41.71	49.37	56.91	68.36
ME-GCN	80.13	86.17	88.92	87.31	94.92	67.22	76.33	75.41	78.92	82.31	37.33	39.11	41.52	41.52	50.13
FewShotTextGCN	82.09	88.17	89.22	93.10	94.56	65.15	74.73	79.36	80.33	83.80	28.69	38.51	42.00	47.39	62.31
TextGCL*	-	-	-	-	98.20	-	-	-	-	96.80	-	-	-	-	**73.30**
RoBERTaGCN	75.29	74.28	75.56	74.69	98.20	79.75	86.99	91.01	91.43	96.10	**53.77**	54.91	59.04	59.44	72.80
RoBERTa	87.99	96.48	96.93	97.30	98.22	72.74	87.54	91.55	93.38	96.22	22.58	50.48	53.15	61.88	70.72
Text-RGNNs-ablated	84.92	95.20	96.53	96.89	97.48	69.28	86.88	90.93	90.26	91.43	22.98	48.90	49.49	52.96	71.09
Text-RGNNs	**96.48**	**97.81**	**97.44**	**97.76**	**98.86**	**87.46**	**93.89**	**95.06**	**95.44**	**96.85**	49.45	**65.52**	**63.29**	**67.33**	72.86
	MR					**SST-2**					**CoLA**				
Method	**1%**	**5%**	**10%**	**20%**	**100%**	**1%**	**5%**	**10%**	**20%**	**100%**	**1%**	**5%**	**10%**	**20%**	**100%**
TensorGCN*	-	-	-	-	77.91	-	-	-	-	-	-	-	-	-	-
TG-Transformer*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
BERT	57.03	79.01	81.40	82.41	85.70	60.44	83.71	86.11	87.39	91.70	42.90	**59.90**	**61.62**	**62.47**	65.05
TextGCN	60.55	65.98	68.88	69.75	76.74	64.17	69.91	72.87	73.96	82.11	45.55	50.03	52.14	52.88	58.25
ME-GCN	67.72	70.13	70.67	72.59	77.82	71.77	74.30	74.76	76.98	83.27	50.94	53.17	53.50	55.03	59.07
FewShotTextGCN	58.72	69.33	71.24	72.78	76.03	72.59	77.52	75.23	77.18	80.28	17.04	15.42	13.71	10.75	28.98
TextGCL*	-	-	-	-	86.20	-	-	-	-	-	-	-	-	-	-
RoBERTaGCN	80.02	80.95	80.64	81.46	89.70	84.99	85.96	85.50	86.57	96.17	**58.59**	53.26	56.53	56.33	67.29
RoBERTa	81.40	84.72	85.98	87.70	89.33	90.13	91.39	93.46	94.15	94.84	29.21	43.55	53.28	57.38	65.85
Text-RGNNs-ablated	82.16	84.83	86.66	87.39	88.58	90.14	91.40	93.58	93.69	93.92	31.30	46.08	52.35	60.09	67.49
Text-RGNNs	**83.91**	**86.41**	**87.51**	**88.35**	**89.98**	**90.60**	**91.74**	**93.69**	**94.38**	**96.28**	38.14	47.73	56.31	61.94	**68.30**

Although our approach outperforms other models on most tasks, it falls short on CoLA across various labeled dataset percentages. This performance gap stems from the unique characteristics of the CoLA dataset, which focuses on linguistic acceptability rather than conventional text classification. While relational and contextual modeling offers significant benefits for other datasets, CoLA demands a finer understanding of syntactic correctness and subtle grammatical nuances. Moreover, more refined hyperparameter tuning could potentially improve performance. Finally, given the large number of documents in CoLA, increasing k in the k-NN graphs to enhance connectivity is impractical due to memory limitations, which may result in reduced graph connectivity and hinder model effectiveness.

4.5.2 Ablation Study

To ascertain the significance of relational modeling in Text-RGNNs, we conducted a comprehensive ablation study. The primary objective of this study was to

determine the impact of explicitly modeling different types of relationships in heterogeneous text graphs on the overall performance. Specifically, we systematically removed all relational dependencies from the graph and projected the resulting structure into a homogeneous graph with a single type of edge. This configuration mirrors the methodology adopted in classical graph-based models such as TextGCN, where the graph representation is simplified, and no relational distinctions are made between different types of nodes or edges.

In this ablated version, referred to as **Text-RGNNs-ablated**, we used a single adjacency matrix to represent the entire graph, disregarding the diverse interactions between words, documents, and other entities. The resulting model was then evaluated on the same set of benchmark datasets as the full Text-RGNNs model. The results of this ablation study, presented in Table 4.2, indicate a noticeable decline in performance across all datasets when relational modeling is omitted.

One of the key observations from this experiment is that the performance drop is most pronounced in low-resource scenarios, where the labeled training data is limited. In such cases, relational modeling becomes particularly crucial because it enables the model to leverage the structural information present in the graph more effectively. By distinguishing between different types of relationships, the full Text-RGNNs model can capture fine-grained interactions and propagate information more accurately, even when the amount of labeled data is sparse.

Furthermore, we observed that the ablated model struggled to maintain competitive performance on tasks requiring a nuanced understanding of inter-document relationships. This finding underscores the importance of relational modeling in capturing global contextual dependencies, which are often critical for tasks such as topic classification and sentiment analysis. Without relational modeling, the model's ability to aggregate and propagate meaningful information across the graph was significantly hindered, leading to suboptimal results.

Overall, the ablation study highlights the critical role of relational modeling

in Text-RGNNs. By explicitly modeling diverse relationships within heterogeneous graphs, Text-RGNNs are able to achieve superior performance, particularly in challenging low-resource settings. These results reinforce our motivation to develop models that go beyond simple graph representations and incorporate relational information to enhance text classification performance.

4.5.3 Computational Cost

While incorporating relational modeling in graph neural networks can introduce additional complexity due to the need for distinct representations of different types of nodes and edges, Text-RGNNs are designed to minimize computational overhead by optimizing matrix operations. Specifically, instead of directly applying relational modeling to the entire heterogeneous graph, Text-RGNNs decompose the heterogeneous graph into blocks and operate on smaller submatrices corresponding to different relation types. This block-wise approach allows efficient parallelization of matrix multiplications and reduces the overall computational burden during both training and inference.

We experimentally evaluate the computational efficiency of Text-RGNNs by comparing the average per-epoch duration with other baseline models, including TextGCN, BertGCN, and RoBERTaGCN. As shown in Table 4.3, Text-RGNNs achieve significantly faster per-epoch times across all datasets. Notably, the reduction in computational time is most pronounced when compared to transformer-based GNN models such as BertGCN and RoBERTaGCN. On datasets like R52 and MR, Text-RGNNs complete an epoch in less than a second, whereas BertGCN and RoBERTaGCN require over 100 seconds per epoch on average.

The primary reason for the superior speed of Text-RGNNs lies in the efficient graph representation and the lightweight nature of the GCN layers used. Unlike transformer-based GCN models that require significant computation due to their reliance on high-dimensional transformer outputs and dense matrix multiplications, Text-RGNNs employ a sparse graph representation that reduces the

number of necessary operations. Additionally, by leveraging well-optimized matrix libraries and parallel computation, Text-RGNNs further minimize runtime without compromising accuracy.

Another critical factor contributing to the computational efficiency of Text-RGNNs is their scalability. As the size of the dataset grows, the computational cost remains manageable due to the block-wise structure and the sparse nature of the adjacency matrices. This scalability makes Text-RGNNs highly suitable for large-scale semi-supervised learning setups, where processing speed is crucial for practical applications.

Moreover, faster per-epoch durations translate into reduced training times, which is particularly beneficial for iterative experimentation and hyperparameter tuning. In real-world scenarios, where model development often involves multiple iterations of training and fine-tuning, the efficiency of Text-RGNNs allows researchers and practitioners to accelerate the development cycle and obtain results more quickly.

In summary, despite the added complexity of relational modeling, Text-RGNNs maintain low computational cost through a combination of efficient graph decomposition, sparse matrix operations, and parallel processing. The results in Table 4.3 clearly demonstrate that Text-RGNNs not only outperform existing models in terms of accuracy but also offer significant advantages in terms of computational efficiency, making them an attractive choice for real-world text classification tasks.

Table 4.3: Average per epoch Durations (secs.). Each model is run for 100 epochs. The fastest models are emboldened individually.

Method	**R8**	**R52**	**Ohsumed**	**MR**
TextGCN	3.21	0.82	0.87	0.72
BertGCN	104.71	150.03	105.34	215.17
RoBERTaGCN	105.63	152.07	109.34	218.32
Text-RGNNs	**1.21**	**0.34**	**0.32**	**0.13**

Chapter 5

VISPool: Enhancing Transformer Encoders with Vector Visibility Graph Neural Networks

In this chapter, we introduce our solution to existing inductive GNNs limitations, which typically rely on static graph construction techniques using statistical metrics such as TF-IDF, PMI, or co-occurrence statistics. These traditional methods require significant preprocessing and often face scalability challenges in large-scale applications. To address these issues, we propose a novel dynamic graph generation mechanism that constructs graphs on the fly for each document, eliminating the dependency on static preprocessing. Additionally, our method seamlessly integrates with transformer-based pipelines, enhancing adaptability and efficiency in various real-world applications. The subsequent sections provide a detailed explanation of our approach and demonstrate how it improves both scalability and performance in text classification tasks.

5.1 Methodology

This section outlines our proposed VISPool framework, which enhances transformer-based models by mapping document embeddings to dynamic graphs and applying graph convolutional network (GCN) layers. In Section 5.1.1, we describe how the transformer-generated embeddings are treated as multivariate time series, forming the basis for graph construction. The visibility graph (VG) approach, detailed in Section 5.1.1.1, defines the visibility criteria for creating graph edges based on sequential data. To improve robustness, we incorporate the concept of limited penetrability, as explained in Section 5.1.1.2, allowing a controlled number of violations in the visibility criteria. The method is further generalized to vector visibility graphs (VVGs) for multivariate time series in Section 5.1.1.3. Finally, Section 5.1.2 presents the GCN layers used for processing the generated graphs, leveraging both local contextual information from transformers and global relational information from graphs to improve classification performance.

5.1.1 Graph Construction

Broadly, the transformer encoders generate an embedding matrix of size $L \times D$ for a single document. Due to the structure's intrinsic sequential nature, each document embedding is a D-dimensional, L-length multivariate time series. This section describes how we leverage this structure with VVGs to construct dynamic graphs for each document.

5.1.1.1 Visibility Graph

For an N-length time series $\{(t_i, x_i)\}_{i=0}^{N-1}$, with values $x_i \in \mathbb{R}$ sampled at time instance $t_i \in \mathbb{R}$, the VG is introduced as a mapping from (t_i, x_i) instance to corresponding graph node v_i [95]. The graph edges are determined based on a

selected visibility criterion, so the unweighted adjacency matrix entries are:

$$\mathbf{A}_{i,j} = \begin{cases} 1 & \text{if nodes } v_i \text{ and } v_j \text{ are } \textit{visible}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

The adjacency can also be weighted based on a metric, e.g., the absolute value difference, inverse time difference, etc. Let (t_i, x_i) and (t_j, x_j) be arbitrary endpoints, with in-between instances (t_k, x_k) such that $i < k < j$. The nodes v_i and v_j are connected, i.e., *visible*, if the visibility criterion is satisfied for all in-between points. A visibility criterion is defined by *line-of-sight* principle where the two most common ones, *natural* [95] and *horizontal* [96], are defined as Eqs. (5.2) and (5.3), respectively:

$$\frac{x_j - x_i}{t_j - t_i} < \frac{x_j - x_k}{t_j - t_k}, \tag{5.2}$$

$$x_k < \min(x_i, x_j). \tag{5.3}$$

Then, $\mathbf{A}$ is defined in a *left-to-right* directed manner, i.e., for $i < j$, so it is upper triangular. However, the graph can be converted to an undirected graph by equating $\mathbf{A}_{j,i}$ to $\mathbf{A}_{i,j}$.

5.1.1.2 Limited Penetrability

The described visibility criteria are strict and require all in-between values to satisfy the criterion. However, [125] show relaxing the criterion and allowing a K number of violations increases robustness to noise and is referred to as *limited penetrability*.

5.1.1.3 Vector Visibility Graph

The VVG generalizes the VG to map a multivariate time series to a graph [99]. For an M-dimensional, N-length multivariate time series $\{(t_i, \mathbf{x}_i)\}_{i=0}^{N-1}$, the VVG is constructed by mapping each vector $\mathbf{x}_i \in \mathbb{R}^M$ at time instance $t_i \in \mathbb{R}$ to a corresponding node v_i. The projection magnitudes are used since the vectors are

not directly comparable in the visibility criteria. The projection magnitude of a vector $\mathbf{x}_j$ onto $\mathbf{x}_i$ is defined as:

$$\|\mathbf{x}_j^i\|_2 \triangleq \|\text{proj}_{\mathbf{x}_i}(\mathbf{x}_j)\|_2 = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\|\mathbf{x}_i\|_2}. \tag{5.4}$$

Then, all value comparisons in the criteria are replaced with the corresponding projection magnitudes. Both visibility criteria and limited penetrability can be directly generalized to VVGs. The vector visibility criteria are generalized for natural and horizontal as Eqs. (5.5) and (5.6), respectively:

$$\frac{\|\mathbf{x}_j^i\|_2 - \|\mathbf{x}_i^i\|_2}{t_j - t_i} < \frac{\|\mathbf{x}_j^i\|_2 - \|\mathbf{x}_k^i\|_2}{t_j - t_k}, \tag{5.5}$$

$$\|\mathbf{x}_k^i\|_2 < \min(\|\mathbf{x}_i^i\|_2, \|\mathbf{x}_j^i\|_2). \tag{5.6}$$

In this context, each document embedding is represented as a multivariate time series in the form of $\{(t_i, \mathbf{x}_i)\}_{i=0}^{L-1}$ with token embeddings $\mathbf{x}_i \in \mathbb{R}^D$ and equally spaced time instances $t_i = i$. Therefore, we propose mapping each of the B document embeddings at the transformer output to a VVG with selected criteria and limited penetrability, resulting in B number of $\mathbf{A} \in \mathbb{R}^{L \times L}$. Then, these graphs are ready to be used in GCN layers. Since the embeddings are updated with training, the graphs are dynamic and updated accordingly.

5.1.2 Graph Convolutional Network Layers

We directly feed the generated VVG adjacencies $\mathbf{A} \in \mathbb{R}^{L \times L}$ to GCN layers [74], since GCNs are the most widely used GNN structures. The ℓ^{th}-layer input-output node embedding relation of the GCN is defined as follows:

$$\mathbf{H}^{(\ell+1)} = \phi(\mathbf{A}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}), \tag{5.7}$$

where ϕ is an activation function, $\mathbf{H}^{(\ell)} \in \mathbb{R}^{L \times d_\ell}$ is the ℓ^{th}-layer d_ℓ-dimensional node embeddings, and $\mathbf{W}^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$ is a trainable weight matrix. Note that $\mathbf{H}^{(0)}$ is the initial node embeddings and can be set based on the application at hand, e.g., identity matrix, GloVe embeddings [9], etc. We use the transformer encoder document embeddings, i.e., the output of the transformer, as the initial

node embeddings $\mathbf{H}^{(0)} \in \mathbb{R}^{L \times D}$ in Eq. (5.7). Note that we propose a batched version of the GCN, which means training a single weight matrix $\mathbf{W}^{(\ell)}$ for the ℓ^{th}-layer on all B graphs in the batch.

Instead of directly passing the document embeddings to the classifier, we propose to map them to graphs and pass them through GCN layers. On top of the transformer's contextual information, our motivation is to capture relational information with GCNs and ultimately enhance the transformer's performance. For mapping, we propose a VVG-based dynamic graph construction that does not require text processing. The method is also scalable since the graph sizes are only dependent on the selected tokenizer sequence length L, which is usually chosen as $\{128, 256, 512\}$. We refer to the overall as *VISPool* and represent it as the *lower path* in Fig. 5.1.

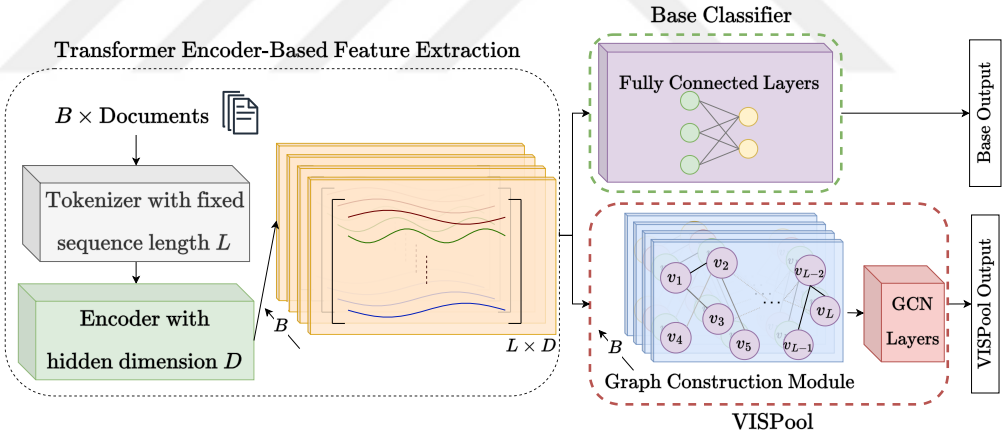


Figure 5.1: The overall representations of baseline and the proposed method. *The base classifier* (upper path) consists of fully connected layers, which is the current norm for transformer-based classification/regression. The proposed *VISPool* architecture (lower path) that maps document embeddings to graphs and passes to GCN layers.

5.2 Experiments

In this section, we provide a comprehensive description of our experimental setup. We begin by detailing the datasets used for evaluation in Section 5.2.1, which includes a variety of GLUE benchmark tasks, their evaluation metrics, and dataset splits. Following this, we describe the baseline models in Section 5.2.2, where transformer-based classifiers, such as DistilBERT [4] and ALBERT [49], are employed as the foundation for comparison with our proposed VISPool framework. Lastly, Section 5.2.3 outlines the hyperparameter tuning process, including the search space and optimization techniques used to ensure robust and reliable performance results.

5.2.1 Datasets

We obtain the GLUE datasets from the *Hugging Face* `datasets` library[1] and use the provided train and validation (dev) splits. We describe the individual datasets below, then give the training and validation splits, evaluation metrics, and the number of labels in Table 5.1.

- **CoLA**: The Corpus of Linguistic Acceptability [126] is a binary classification task for single sentences that aims to classify whether an English sentence is linguistically acceptable correctly.
- **SST-2**: The Stanford Tree Sentimentbank [120] is a binary classification task for single sentences based on film reviews containing human sentiment annotation.
- **MRPC**: The Microsoft Research Paraphrase Corpus [127] consists of sentence pairs extracted from online news sources with human annotations. A similarity and paraphrase classification task classifies whether the sentences in the pair are semantically equivalent.

[1] https://huggingface.co/datasets/glue

- **QQP**: Quora Question Pairs [128] is a binary classification task to determine whether two questions asked on Quora are semantically equivalent.
- **STS-B**: The Semantic Textual Similarity Benchmark [129] consists of sentence pairs drawn from news headlines and other sources. Compared to other tasks in GLUE, STS-B is a regression task since the pairs are annotated with a score from 1 to 5, denoting the similarity of two sentences in terms of semantic meaning.
- **MNLI**: Multi-Genre Natural Language Inference is a large-scale, crowd-sourced entailment classification task consisting of sentence pairs [130]. The aim is to predict the relation between the pair, whether the sentences are an entailment, contradiction, or neutral.
- **QNLI**: The Question Natural Language Inference is a binary classification conversion of the Stanford Question Answering Dataset [131] by [132].
- **RTE**: The Recognizing Textual Entailment (RTE) datasets come from a series of annual textual entailment challenges by combining the data from RTE1 [133], RTE2 [134], RTE3 [135], and RTE5 [136].
- **WNLI**: The Winograd Schema Challenge is a reading comprehension task [137], although the GLUE webpage indicates issues with the dataset. We obtained the same result for all models, consistent with the reported results in the original DistilBERT approach [4].

5.2.2 Baseline Models

For a given corpus batch of B documents, a tokenizer with a fixed sequence length of L, and a pre-trained transformer encoder with D-dimensional hidden output, the transformer encoder generates a $B \times L \times D$ dimensional tensor representation for the document batch. The representation contains B sequences of L-length and D-dimensional embeddings. The mainstream approach uses these embeddings with a classifier or regressor head, implemented as fully connected layers. Then,

Dataset	Evaluation Metric	Train Split	Validation Split	Number of Labels
CoLA	Matthews Corr.	8,551	1,043	2
SST-2	Accuracy	67,349	872	2
MRPC	Accuracy/F1	3668	408	2
QQP	Accuracy/F1	363,846	40,430	2
STS-B	Pearson/Spearman Corr.	5,749	1,500	1
MNLI	Accuracy	392,702	9,815/9,832	3
QNLI	Accuracy	104,743	5,463	2
RTE	Accuracy	2,490	277	2
WNLI	Accuracy	635	71	2

Table 5.1: Number of documents in training and validation splits, evaluation metric, and number of labels of the GLUE datasets. The STS-B is a regression task with a single label; the remaining are classification tasks. For the MNLI task, both matched and mismatched validation splits are provided M/MM.

fine-tune the whole network for final predictions on the NLU task. We refer to this approach as *base classifier* and represent it as the *upper path* in Fig. 5.1.

We evaluate our approach by addressing whether, given a pre-trained transformer, the VVG-based GCNs can enhance the performance in NLU compared to the base classifier. Due to their lightweight but competitive performance, we choose DistilBERT [4] and ALBERT [49] as base transformers and use their corresponding tokenizers, both provided by *Hugging Face*[2]. Up to this point, the process is identical for the base classifier and VISPool. Then, for the base classifier, we use *SequenceClassifier*[2] head.

On the other hand, for the VISPool path, we implement a fast and parallel VVG generation process, along with a batched GCN layer implementation that extends the notion in *PyTorch Geometric* [138], to minimize the computational overhead, resulting in similar training times with the base classifier.

5.2.3 Hyperparameter Tuning

For the base classifier, we conduct an individual grid search on each dataset with five seeds and four learning rates suggested by [48]. We report the scores on the

[2]We use models from huggingface.co/distilbert/distilbert-base-uncased and huggingface.co/albert/albert-base-v2.

best-performing overall seed and best learning rate for each dataset. For fairness, we use the selected seed for VISPool and employ Bayesian hyperparameter optimization [139].

[48] suggests learning rates of $\{2 \times 10^{-5}, 3 \times 10^{-5}, 4 \times 10^{-5}, 5 \times 10^{-5}\}$ for the transformer encoder. Therefore, the base classifier's fine-tuning process is set to a grid search between these learning rates and five different seeds of $\{40, 41, 42, 43, 44\}$. Within these seeds, 42 is the top performing in the overall average. Therefore, we conduct a hyperparameter search on VISPool with a single seed of 42. We provide the summary for the list and distributions of the hyperparameters used in experiments in Table 5.2. We further explain our approach as follows.

Table 5.2: The hyperparameter distributions utilized during the experiments, where the categorical distributions have the same initial sampling probability.

Hyperparameter	**Distribution**
Encoder	
learning rate	$\sim$ Uniform$(10^{-5}, 10^{-4})$
VVG	
type	$\in \{\text{natural}, \text{horizontal}\}$
penetrable limit	$\in \{0, 1, 2, 3, 4, 5\}$
GCN	
learning rate	$\sim$ Uniform$(10^{-5}, 10^{-2})$
dropout	$\sim$ Uniform$(0.1, 0.5)$
hidden dimension	$\in \{128, 512\}$
pooling	$\in \{$ `[CLS]`, mean, max $\}$

As a common practice, we have fixed selections of batch size as 32 and tokenizer length as 128. Also, with the choice of the language model as DistilBERT (base-uncased) and ALBERT (base-v2), we overall have $(B, L, D) = (32, 128, 768)$. Therefore, we only have the transformer encoder's learning rate as a hyperparameter. We choose the learning rate to be uniform in $[10^{-5}, 10^{-4}]$ to encapsulate the suggested values.

On the VVG side, we have a categorical parameter of visibility type, either

natural or horizontal. In addition, penetrable limit K is also treated as a hyperparameter. Technically, K can be as large as the sequence length L, but it would generate a fully connected graph. Since large K can create unnecessary edges in the VVG, we experiment with $K \in \{0, 1, 2, 3, 4, 5\}$ to also decrease the search grid. We highlight that $K = 0$ corresponds to strict visibility criteria.

Finally, on the GCN side, along with the classical hyperparameter selections of learning rate, dropout, and hidden dimension, we also provide an option to select the pooling method at the end. By default, BERT variants pool the `[CLS]` tokens at the end, but due to the provided architecture, we also have the flexibility to choose how to treat the pre-logits. We experiment with classical approaches of still using `[CLS]` token (taking the first embedding), using mean and max approaches. Additionally, we use a 2-layer GCN to avoid the over-smoothing [105] problem, ensuring that node representations remain distinct and informative.

Last but not least, due to common practice, we use Adam with weight decay as an optimizer for both base classifier and VISPool, with default values of *PyTorch* implementation: $(\beta_1, \beta_2) = (0.9, 0.999), \epsilon = 10^{-8}$ and weight decay $\lambda = 0.01$. We publicly share the best-performing hyperparameters and the runs that achieve maximum scores directly with `WandB` dashboard at [3].

5.3 Results & Discussions

In this section, we compare baseline models with their VISPool-enhanced counterparts, employing both natural (NVVG) and horizontal (HVVG) visibility criteria, as outlined in Section 5.3.1. Additionally, we include limited penetrability (LP) variants, which relax the strict visibility criteria by allowing up to K violations, with K treated as a hyperparameter. The results are presented on the GLUE benchmark, covering a diverse range of natural language understanding tasks. We also provide a visualization of the graph representations generated by VISPool in Section 5.3.2 to offer insight into the dynamic graph construction process.

[3] https://wandb.ai/tunakasif/vispool

5.3.1 Main Results

The baseline models described in Section 5.2.2 are used as the base classifiers and compared against their VVG-based VISPool counterparts, incorporating unweighted and undirected graph variants with natural (NVVG) and horizontal (HVVG) visibility criteria, as detailed in Section 5.1.1. Additionally, we include limited penetrability (LP) variants, allowing up to $K \in \{1, \ldots, 5\}$ violations, where K is treated as a hyperparameter. Since the test labels are not publicly available, we follow standard practice and report the highest achieved scores on the validation (dev) sets in Table 5.3, with the best scores highlighted in bold.

Table 5.3: Maximum achieved scores on the GLUE validation (dev) sets (scaled by 100). We report the corresponding metric for each task Matthews correlation for CoLA Pearson/Spearman correlation for STS-B accuracy/F1 for MRPC and QQP accuracy for all other tasks, where, for MNLI, both on the matched and mismatched sets.

Model	AVG	Single Sequence		Similarity and Paraphrase			Natural Language Inference			
		CoLA	SST-2	MRPC Acc/F1	QQP Acc/F1	STS-B P/SP	MNLI m/mm	QNLI	RTE	WNLI
DistilBERT										
Base Classifier	79.44	55.54	91.63	85.78/89.78	90.39/86.61	82.23/80.01	81.83/82.32	89.62	60.65	**56.34**
VISPool-NVVG	79.07	53.87	90.14	83.58/88.74	90.51/86.83	82.22/80.17	**82.44**/81.88	88.43	62.82	**56.34**
VISPool-HVVG	78.97	53.67	90.48	84.07/88.86	89.44/85.54	**82.56/80.56**	81.90/82.54	89.62	61.01	**56.34**
VISPool-LP-NVVG	79.76	54.77	91.40	**86.27/90.23**	89.76/85.67	82.39/80.26	82.39/**82.71**	89.04	**65.70**	**56.34**
VISPool-LP-HVVG	**79.89**	**57.09**	**91.74**	85.78/89.98	**90.52/86.93**	82.42/80.20	82.08/82.41	**89.93**	63.18	**56.34**
ALBERT										
Base Classifier	82.07	53.77	91.17	88.97/91.76	90.28/86.48	**87.67/85.54**	84.01/84.47	90.24	76.17	**56.34**
VISPool-NVVG	81.73	55.70	92.89	89.71/92.54	90.32/86.48	85.60/83.12	83.93/81.81	90.43	73.65	**56.34**
VISPool-HVVG	81.83	57.54	88.76	89.71/92.52	89.82/85.81	85.87/83.57	84.23/83.11	90.40	76.17	**56.34**
VISPool-LP-NVVG	82.60	**60.45**	93.00	**89.95/92.81**	**90.57/86.86**	85.52/82.82	**84.34/84.88**	90.85	75.45	**56.34**
VISPool-LP-HVVG	**82.74**	59.41	**93.12**	89.46/92.48	90.55/86.80	86.23/83.56	84.25/84.54	**91.21**	**77.62**	**56.34**

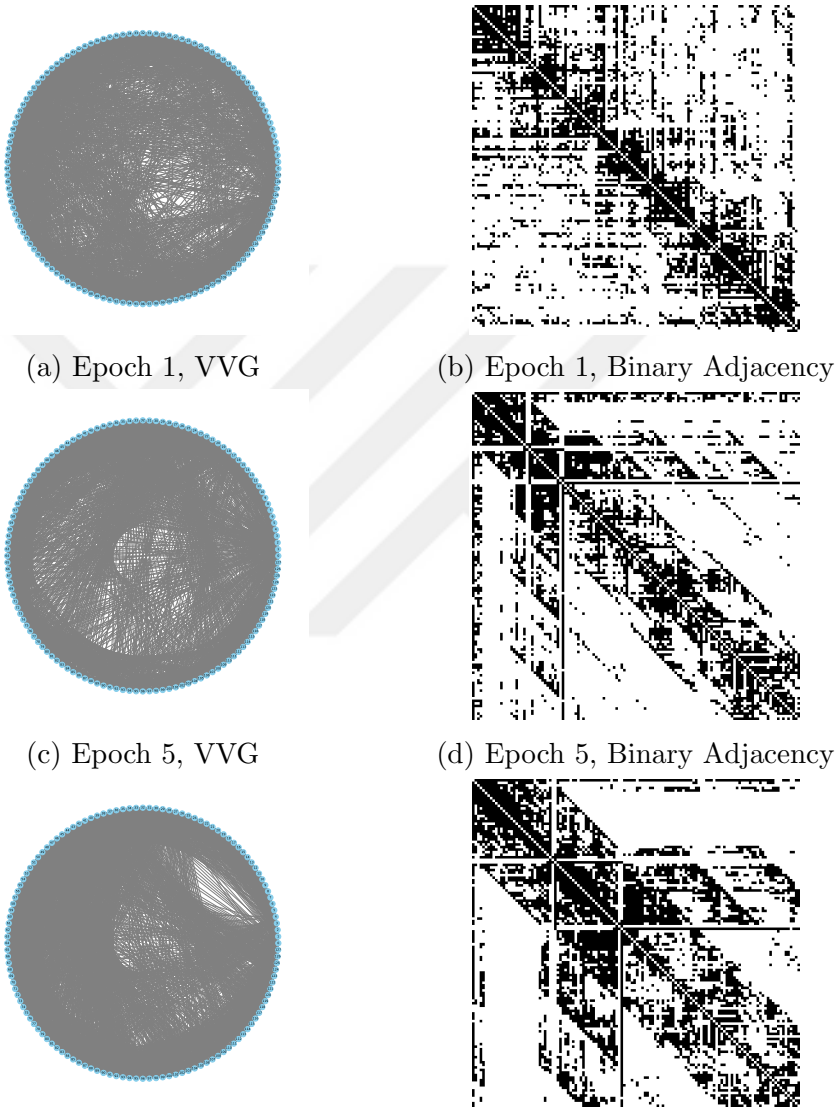
On the GLUE benchmark, with fewer trainable parameters, the VISPool-LP variants systematically outperform baselines, improving the overall average score. On the other hand, even though the strict versions' overall average is not better, they still provide improvements in some datasets; there are even cases where they are top performers.

Since relaxing the visibility criteria with limited penetrability results in denser graphs, i.e., graphs with more connections, it can be said that VISPool variants usually perform better with less sparse graph representations. We highlight that the only differences between the variants are the parameters they generate for

the VVG graph. Therefore, as the results in Table 5.3 suggest, the performance of VISPool is highly dependent on the quality of the graph construction, which is adjusted by the hyperparameters of visibility criteria (natural or horizontal) and the limited penetrability K. With the hyperparameter space we cover, VISPool provides performance improvements over base classifiers up to 13% for certain tasks in GLUE benchmarks, resulting in a promising approach for enhancing transformer performance.

5.3.2 Graph Visualization

Finally, we demonstrate the gradual change of the VVG representations during the training in Fig. 5.2. We provide an example for the first document of the RTE dataset with the setting of a DistilBERT transformer and VISPool-NVVG variant. We both provide the graph and the binary representations of the adjacency matrix to provide intuition for the generated graphs, where they visualize the connections between $L = 128$ tokens. We provide `eps` format images and suggest zooming in for details of the figures.



(a) Epoch 1, VVG

(b) Epoch 1, Binary Adjacency

(c) Epoch 5, VVG

(d) Epoch 5, Binary Adjacency

(e) Epoch 10, VVG

(f) Epoch 10, Binary Adjacency

Figure 5.2: Gradual change of the first document of the RTE dataset with the setting of a DistilBERT transformer and VISPool-NVVG variant. Left-hand side subfigures represent the generated VVG at each epoch, whereas right-hand side subfigures are the binary representation of their adjacency matrices, where black pixels indicate a connection between nodes. For epochs 1, 5, and 10, we obtain validation accuracies of 47.29%, 52.70%, and 62.82%, with graph sparsities of 29.15%, 28.44%, and 33.29%, respectively.

Chapter 6

Conclusion

In this thesis, we tackled several significant limitations of encoder-only transformer models for text classification by introducing novel models that incorporate GNNs. While transformer models such as BERT have become the dominant choice for text classification tasks due to their remarkable ability to capture local contextual information within individual documents, they suffer from notable weaknesses. These include an inability to model inter-document relationships, limited capacity to leverage unlabeled data effectively during fine-tuning, and reliance on simple pooling strategies that inadequately capture the sequential structure of data. Our work aims to bridge these gaps by proposing three distinct models, each addressing a specific challenge in text classification.

In Chapter 3, we presented the GRTE model, designed to address the challenges associated with transductive GNNs, which require the entire graph to be available during both training and inference. GRTE leverages the contextual strengths of pre-trained transformer models and extends them with a scalable transductive GNN that can also perform inductive inference. This approach allows GRTE to utilize both labeled and unlabeled data effectively through semi-supervised learning, leading to significant performance improvements in scenarios where labeled data is scarce. By combining the benefits of transductive learning with inductive generalization capabilities, GRTE provides a robust solution for

large-scale, real-world applications that involve continuously changing data.

In Chapter 4, we proposed the Text-RGNNs framework, which addresses the oversimplification issue in existing GNN-based methods that model heterogeneous graphs using homogeneous GNNs. Many prior approaches use a single-weight matrix to process all types of nodes and edges in heterogeneous text graphs, failing to capture the nuanced relationships between different node types, such as words and documents. Text-RGNNs introduces a relational modeling approach that explicitly differentiates between various node types and their interactions, thereby offering a more fine-grained and accurate representation of the data. This approach enhances the model's ability to learn complex patterns in text corpora and results in superior classification performance across multiple benchmark datasets.

In Chapter 5, we presented the VISPool approach, a novel inductive model that dynamically generates graphs for each document using vector visibility graphs (VVGs). Unlike traditional methods that rely on static graph construction techniques based on statistical metrics such as TF-IDF or point-wise mutual information (PMI), VISPool constructs graphs directly from transformer-generated embeddings without requiring text preprocessing. This dynamic graph generation preserves the sequential structure inherent in document embeddings and seamlessly integrates with transformer-based pipelines. Furthermore, VISPool includes a mechanism for controlling graph sparsity through the use of limited penetrability, ensuring that the generated graphs remain both meaningful and computationally efficient. As a result, VISPool achieves notable improvements in classification accuracy while maintaining scalability for large datasets.

The models developed in this thesis offer a unified framework for addressing the key challenges in text classification using transformer and GNN-based methods. By integrating transformer encoders' contextual learning capabilities with GNNs' ability to model relational information, we provide solutions that enhance both local and global context modeling, leverage unlabeled data effectively, and maintain computational efficiency. Our contributions are particularly valuable in low-resource settings where labeled data is scarce and in dynamic environments where data evolves continuously.

6.1 Key Findings

Through rigorous experimentation and analysis, we derived several key findings that demonstrate the effectiveness of our proposed models:

- **Improved performance in low-resource settings**: GRTE's ability to leverage both labeled and unlabeled data through semi-supervised learning enables significant performance gains in scenarios where labeled data is limited. This makes GRTE particularly suitable for real-world applications where data annotation is expensive or time-consuming.
- **Relational modeling for heterogeneous graphs**: Text-RGNNs relational approach to handling heterogeneous text graphs addresses a major limitation in existing GNN-based methods. By explicitly modeling different types of nodes and their interactions, Text-RGNNs achieve superior classification accuracy compared to models that treat heterogeneous graphs as homogeneous.
- **Dynamic graph generation with minimal preprocessing**: VISPool's use of vector visibility graphs (VVGs) for dynamic graph generation eliminates the need for extensive text preprocessing, making it a highly efficient solution for large-scale text classification. The ability to generate graphs on the fly while preserving the sequential structure of transformer embeddings is a key advantage of this approach.
- **Scalability and adaptability**: All three proposed models are designed with scalability in mind. GRTE and VISPool, in particular, are well-suited for large datasets due to their efficient use of graph-based representations and lightweight GNN architectures.

6.2 Future Directions

While the proposed models offer significant advancements, several avenues for future research remain open. One promising direction is to explore more sophisticated relational modeling techniques that go beyond the current formulation in Text-RGNNs. Additionally, extending the proposed models to other NLP tasks, such as question answering, information retrieval, and text summarization, could yield further insights and applications. Another potential direction is to investigate the integration of graph-based methods with advanced pre-training strategies, such as self-supervised or contrastive learning, to enhance the models' generalization capabilities further.

Moreover, given the increasing importance of multi-modal learning, an exciting research avenue would be to adapt the proposed methods for multi-modal tasks that involve both text and other data types, such as images or audio. This would require developing new graph construction techniques capable of representing multi-modal relationships and learning from them effectively.

6.3 Final Remarks

This thesis contributes to the ongoing research in text classification by proposing novel models that combine the strengths of transformers and GNNs. By addressing key limitations of existing methods, such as the inability to model inter-document relationships, reliance on labeled data, and simplistic pooling strategies, our work provides a comprehensive solution that enhances both classification accuracy and scalability. We hope that the models and methodologies presented in this thesis will serve as a foundation for future research and inspire further innovations in the field of NLP.

Bibliography

[1] A. C. Aras, T. Alikaşifoglu, and A. Koç, "Graph receptive transformer encoder for text classification," *IEEE Transactions on Signal and Information Processing over Networks*, 2024.

[2] A. C. Aras, T. Alikaşifoğlu, and A. Koç, "Text-rgnns: Relational modeling for heterogeneous text graphs," *IEEE Signal Processing Letters*, 2024.

[3] T. Alikaşifoğlu, A. C. Aras, and A. Koç, "VISPool: Enhancing transformer encoders with vector visibility graph neural networks," pp. 2547–2556, 2024.

[4] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," arXiv, 2019.

[5] R. A. Sinoara, J. Camacho-Collados, R. G. Rossi, R. Navigli, and S. O. Rezende, "Knowledge-enhanced document embeddings for text classification," *Knowledge-Based Systems*, vol. 163, pp. 955–971, 2019.

[6] T. Çabuk, N. Sevim, E. Mutlu, A. E. A. Yağcıoğlu, A. Koç, and T. Toulopoulou, "Natural language processing for defining linguistic features in schizophrenia: A sample from turkish speakers," *Schizophrenia Research*, vol. 266, pp. 183–189, 2024.

[7] O. M. Battal and A. Koç, "Automatic construction of sememe knowledge bases from machine readable dictionaries," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 1023–1035, 2024.

[8] M. Bozdag, N. Sevim, and A. Koç, "Measuring and mitigating gender bias in legal contextualized language models," *ACM Trans. Knowl. Discov. Data*, vol. 18, Feb. 2024.

[9] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, (Doha, Qatar), pp. 1532–1543, Association for Computational Linguistics, Oct. 2014.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st Int. Conf. on Learn. Repr., (ICLR)*, 2013.

[11] C. E. Öztürk, E. H. Yilmaz, O. Köksal, and A. Koç, "Software module classification for commercial bug reports," in *2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, pp. 1–5, 2023.

[12] C. Çetindağ, B. Yazıcıoğlu, and A. Koç, "Named-entity recognition in turkish legal texts," *Natural Language Engineering*, vol. 29, no. 3, p. 615–642, 2023.

[13] N. Sevim, F. Şahinuç, and A. Koç, "Gender bias in legal corpora and debiasing it," *Natural Language Engineering*, vol. 29, no. 2, p. 449–482, 2023.

[14] I. Chalkidis, E. Fergadiotis, P. Malakasiotis, and I. Androutsopoulos, "Large-scale multi-label text classification on EU legislation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 6314–6322, Association for Computational Linguistics, July 2019.

[15] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, (Valencia, Spain), pp. 427–431, Apr. 2017.

[16] S. F. Yilmaz, E. B. Kaynak, A. Koç, H. Dibeklioğlu, and S. S. Kozat, "Multi-label sentiment analysis on 100 languages with dynamic weighting for label imbalance," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 1, pp. 331–343, 2023.

[17] F. Şahinuç and A. Koç, "Fractional fourier transform meets transformer encoder," *IEEE Signal Processing Letters*, vol. 29, pp. 2258–2262, 2022.

[18] C. E. Öztürk, O. Köksal, S. B. Özçelik, and A. Koç, "Prior case retrieval for the court of cassation of turkey," in *2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pp. 539–544, 2022.

[19] N. Sevim, E. O. Özyedek, F. Şahinuç, and A. Koç, "Fast-fnet: Accelerating transformer encoder models via efficient fourier layers," 2023.

[20] M. Cemri, T. Çukur, and A. Koç, "Unsupervised simplification of legal texts," 2022.

[21] A. C. Aras, C. E. Öztürk, and A. Koç, "Feedforward neural network based case prediction in turkish higher courts," in *2022 30th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2022.

[22] C. E. Öztürk, S. B. Ozçelık, and A. Koç, "Predicting outcomes of the court of cassation of turkey with recurrent neural networks," in *2022 30th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2022.

[23] L. K. Şenel, F. Şahinuç, V. Yücesoy, H. Schütze, T. Çukur, and A. Koç, "Learning interpretable word embeddings via bidirectional alignment of dimensions with semantic concepts," *Information Processing & Management*, vol. 59, no. 3, p. 102925, 2022.

[24] A. Yüksel, B. Uğurlu, and A. Koç, "Semantic change detection with gaussian word embeddings," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3349–3361, 2021.

[25] E. Mumcuoğlu, C. E. Öztürk, H. M. Ozaktas, and A. Koç, "Natural language processing in law: Prediction of outcomes in the higher courts of turkey," *Information Processing & Management*, vol. 58, no. 5, p. 102684, 2021.

[26] Q. Ma, L. Yu, S. Tian, E. Chen, and W. W. Y. Ng, "Global-local mutual attention model for text classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 12, pp. 2127–2139, 2019.

[27] Y. Meng, Y. Zhang, J. Huang, C. Xiong, H. Ji, C. Zhang, and J. Han, "Text classification using label names only: A language model self-training approach," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9006–9017, Association for Computational Linguistics, Nov. 2020.

[28] N. Sevim and A. Koç, "Investigation of gender bias in turkish word embeddings," in *2021 29th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2021.

[29] F. Şahinuç, C. Toraman, and A. Koç, "Topic detection based on deep learning language model in turkish microblogs," in *2021 29th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2021.

[30] F. Şahinuç and A. Koç, "Zipfian regularities in "non-point" word representations," *Information Processing & Management*, vol. 58, no. 3, p. 102493, 2021.

[31] L. K. Şenel, I. Utlu, F. Şahinuç, H. M. Ozaktas, and A. Koç, "Imparting interpretability to word embeddings while preserving semantic structure," *Natural Language Engineering*, vol. 27, no. 6, p. 721–746, 2021.

[32] F. Şahinuç, V. Yücesoy, and A. Koç, "Intent classification and slot filling for turkish dialogue systems," in *2020 28th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2020.

[33] L. K. Şenel, V. Yücesoy, A. Koç, and T. Çukur, "Topic change detection on dialog based text," in *2019 27th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2019.

[34] V. Yücesoy and A. Koç, "Co-occurrence weight selection in generation of word embeddings for low resource languages," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 18, Jan. 2019.

[35] L. K. Şenel, I. Utlu, V. Yücesoy, A. Koç, and T. Çukur, "Generating semantic similarity atlas for natural languages," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 795–799, 2018.

[36] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1746–1751, Association for Computational Linguistics, Oct. 2014.

[37] L. K. Şenel, V. Yücesoy, A. Koç, and T. Çukur, "Interpretability analysis for turkish word embeddings," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2018.

[38] L. K. Şenel, V. Yücesoy, A. Koç, and T. Çukur, "Measuring cross-lingual semantic similarity across european languages," in *2017 40th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 359–363, 2017.

[39] L. K. Sjenel, V. Yücesoy, A. Koç, and T. Çukur, "Semantic similarity between turkish and european languages using word embeddings," in *2017 25th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2017.

[40] V. Yücesoy and A. Koç, "Effect of cooccurance weighting to english word embeddings," in *2017 25th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2017.

[41] V. Yücesoy and A. Koç, "Effect of the training set on the word embeddings and similarity test set for turkish," in *2016 24th Signal Processing and Communication Application Conference (SIU)*, pp. 1005–1008, 2016.

[42] Z. Lin, X. Jin, X. Xu, Y. Wang, X. Cheng, W. Wang, and D. Meng, "An unsupervised cross-lingual topic model framework for sentiment classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 3, pp. 432–444, 2016.

[43] Q. Ma, J. Yan, Z. Lin, L. Yu, and Z. Chen, "Deformable self-attention for text classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1570–1581, 2021.

[44] D. Tang, B. Qin, F. Wei, L. Dong, T. Liu, and M. Zhou, "A joint segmentation and classification framework for sentence level sentiment classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 11, pp. 1750–1761, 2015.

[45] Z. Shao, Z. Wu, and M. Huang, "Advexpander: Generating natural language adversarial examples by expanding text," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 1184–1196, 2022.

[46] X.-D. Zhang, "A general decision layer text classification fusion model," in *2010 2nd International Conference on Education Technology and Computer*, vol. 5, pp. V5–239–V5–241, 2010.

[47] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, p. 2873–2879, AAAI Press, 2016.

[48] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the ACL: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.

[49] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," *CoRR*, vol. abs/1909.11942, 2019.

[50] E. Koç, Y. E. Ekiz, H. Ozaktas, and A. Koç, "Maximally selective fractional fourier pooling," in *2024 32nd Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2024.

[51] T. Alikaşifoğlu, B. Kartal, and A. Koç, "Wiener filtering in joint time-vertex fractional fourier domains," *IEEE Signal Processing Letters*, vol. 31, pp. 1319–1323, 2024.

[52] E. Koç, T. Alikaşifoğlu, A. C. Aras, and A. Koç, "Trainable fractional fourier transform," *IEEE Signal Processing Letters*, vol. 31, pp. 751–755, 2024.

[53] E. Koç and A. Koç, "Fractional fourier transform in time series prediction," *IEEE Signal Processing Letters*, vol. 29, pp. 2542–2546, 2022.

[54] A. Koç and H. M. Ozaktas, "Relationship between the beam propagation method and linear canonical and fractional fourier transforms," *Appl. Opt.*, vol. 61, pp. 10275–10282, Dec 2022.

[55] A. Y. Yıldız, E. Koç, and A. Koç, "Multivariate time series imputation with transformers," *IEEE Signal Processing Letters*, vol. 29, pp. 2517–2521, 2022.

[56] A. Koç and H. M. Ozaktas, "Operator theory-based computation of linear canonical transforms," *Signal Processing*, vol. 189, p. 108291, 2021.

[57] T. Alikaşifoğlu, B. Kartal, E. Özgünay, and A. Koç, "Joint time-vertex fractional fourier transform," 2024.

[58] A. Koç, B. Bartan, and H. M. Ozaktas, "Discrete scaling based on operator theory," *Digital Signal Processing*, vol. 108, p. 102904, 2021.

[59] C. Işil, F. S. Oktem, and A. Koç, "Deep learning-based hybrid approach for phase retrieval," in *Imaging and Applied Optics 2019 (COSI, IS, MATH, pcAOP)*, p. CTh2C.5, Optica Publishing Group, 2019.

[60] Çağatay Işil, F. S. Oktem, and A. Koç, "Deep iterative reconstruction for phase retrieval," *Appl. Opt.*, vol. 58, pp. 5422–5431, Jul 2019.

[61] A. Koç, B. Bartan, and H. M. Ozaktas, "Discrete linear canonical transform based on hyperdifferential operators," *IEEE Transactions on Signal Processing*, vol. 67, no. 9, pp. 2237–2248, 2019.

[62] A. Koç, H. M. Ozaktas, and L. Hesselink, "Fast and accurate computation of two-dimensional non-separable quadratic-phase integrals," *JOSA A*, vol. 27, no. 6, pp. 1288–1302, 2010.

[63] H. M. Özaktaş and A. Koç, "Fast and accurate linear canonical transform algorithms," in *2015 23nd Signal Processing and Communications Applications Conference (SIU)*, pp. 1409–1412, IEEE, 2015.

[64] A. Koç, H. M. Ozaktas, and L. Hesselink, "Fast and accurate algorithms for quadratic phase integrals in optics and signal processing," in *Three-Dimensional Imaging, Visualization, and Display 2011*, vol. 8043, pp. 26–30, SPIE, 2011.

[65] A. Koç, H. M. Ozaktas, and L. Hesselink, "Fast and accurate algorithm for the computation of complex linear canonical transforms," *JOSA A*, vol. 27, no. 9, pp. 1896–1908, 2010.

[66] A. Koç, H. M. Ozaktas, and L. Hesselink, "Fast and accurate computation of two-dimensional non-separable quadratic-phase integrals," *JOSA A*, vol. 27, no. 6, pp. 1288–1302, 2010.

[67] A. Koç, H. M. Ozaktas, C. Candan, and M. A. Kutay, "Digital computation of linear canonical transforms," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2383–2394, 2008.

[68] R. A. Sevimli, M. Üçüncü, and A. Koç, "Radgt: Graph and transformer-based automotive radar point cloud segmentation," *IEEE Sensors Letters*, vol. 7, no. 11, pp. 1–4, 2023.

[69] T. Alikaşifoğlu, B. Kartal, and A. Koç, "Graph fractional fourier transform: A unified theory," *IEEE Transactions on Signal Processing*, vol. 72, pp. 3834–3850, 2024.

[70] R. A. Sevimli, M. Üçüncü, and A. Koç, "Graph signal processing based object classification for automotive radar point clouds," *Digital Signal Processing*, vol. 137, p. 104045, 2023.

[71] B. Kartal, Y. E. Bayiz, and A. Koç, "Graph signal processing: Vertex multiplication," *IEEE Signal Processing Letters*, vol. 28, pp. 1270–1274, 2021.

[72] C. Ozturk, H. M. Ozaktas, S. Gezici, and A. Koç, "Optimal fractional fourier filtering for graph signals," *IEEE Transactions on Signal Processing*, vol. 69, pp. 2902–2912, 2021.

[73] Y. Lin, Y. Meng, X. Sun, Q. Han, K. Kuang, J. Li, and F. Wu, "BertGCN: Transductive text classification by combining GNN and BERT," in *Findings of the Association for Computational Linguistics: ACL 2021*, pp. 1456–1462, Association for Computational Linguistics, Aug. 2021.

[74] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[75] P. Chuanakrud, T. Leelanupab, C. Damrongrat, and N. Kanungsukkasem, "Keyword-text graph representation for short text classification," in *2021 13th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pp. 24–29, 2021.

[76] G. Lan, Y. Li, M. Hu, Y. Sun, and Y. Zhang, "Knowledge graph integrated graph neural networks for Chinese medical text classification," in *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 682–687, 2021.

[77] H. Linmei, T. Yang, C. Shi, H. Ji, and X. Li, "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)*, (Hong Kong, China), pp. 4821–4830, Association for Computational Linguistics, Nov. 2019.

[78] Y. Liu and X. Gou, "A text classification method based on graph attention networks," in *2021 International Conference on Information Technology and Biomedical Engineering (ICITBE)*, pp. 35–39, 2021.

[79] H. Zhang and J. Zhang, "Text graph transformer for document classification," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8322–8327, Association for Computational Linguistics, Nov. 2020.

[80] Y. Zhang, X. Yu, Z. Cui, S. Wu, Z. Wen, and L. 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*, pp. 334–339, Association for Computational Linguistics, July 2020.

[81] Y. Yang, Q. Cui, L. Ji, and Z. Cheng, "Graph convolution word embedding and attention for text classification," in *2022 The 6th International Conference on Machine Learning and Soft Computing*, ICMLSC 2022, (New York, NY, USA), pp. 160–166, Association for Computing Machinery, 2022.

[82] R. Boutalbi, M. Ait-Saada, A. Iurshina, S. Staab, and M. Nadif, "Tensor-based graph modularity for text data clustering," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, (New York, NY, USA), pp. 2227–2231, Association for Computing Machinery, 2022.

[83] B. Wang, T. Shen, G. Long, T. Zhou, Y. Wang, and Y. Chang, "Structure-augmented text representation learning for efficient knowledge graph completion," in *Proceedings of the Web Conference 2021*, WWW '21, (New York, NY, USA), pp. 1737–1748, Association for Computing Machinery, 2021.

[84] X. Liu, X. You, X. Zhang, J. Wu, and P. Lv, "Tensor graph convolutional networks for text classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 8409–8416, Apr. 2020.

[85] A. ImaniGooghari, M. Jalili Sabet, L. K. Senel, P. Dufter, F. Yvon, and H. Schütze, "Graph algorithms for multiparallel word alignment," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, (Online and Punta Cana, Dominican Republic), pp. 8457–8469, Association for Computational Linguistics, Nov. 2021.

[86] A. Imani, L. K. Senel, M. Jalili Sabet, F. Yvon, and H. Schütze, "Graph neural networks for multiparallel word alignment," in *Findings of the Association for Computational Linguistics: ACL 2022*, (Dublin, Ireland), pp. 1384–1396, Association for Computational Linguistics, May 2022.

[87] A. ImaniGooghari, S. Severini, M. Jalili Sabet, F. Yvon, and H. Schütze, "Graph-based multilingual label propagation for low-resource part-of-speech tagging," in *Proceedings of the 2022 Conference on Empirical Methods in NLP*, (Abu Dhabi, United Arab Emirates), pp. 1577–1589, Association for Computational Linguistics, Dec. 2022.

[88] V. Hofmann, X. Dong, J. Pierrehumbert, and H. Schütze, "Modeling ideological salience and framing in polarized online groups with graph neural networks and structured sparsity," in *Findings of the Association for Computational Linguistics: NAACL 2022*, (Seattle, United States), pp. 536–550, Association for Computational Linguistics, July 2022.

[89] M. Schmitt, L. F. R. Ribeiro, P. Dufter, I. Gurevych, and H. Schütze, "Modeling graph structure via relative position for text generation from knowledge graphs," in *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*, (Mexico City, Mexico), pp. 10–21, Association for Computational Linguistics, June 2021.

[90] I. Kuznetsov, J. Buchmann, M. Eichler, and I. Gurevych, "Revise and Resubmit: An Intertextual Model of Text-based Collaboration in Peer Review," *Computational Linguistics*, vol. 48, pp. 949–986, 12 2022.

[91] L. F. R. Ribeiro, M. Liu, I. Gurevych, M. Dreyer, and M. Bansal, "FactGraph: Evaluating factuality in summarization with semantic graph representations," in *Proceedings of the 2022 Conference of the North American*

Chapter of the Association for Computational Linguistics: Human Language Technologies, (Seattle, United States), pp. 3238–3253, Association for Computational Linguistics, July 2022.

[92] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 7370–7377, 2019.

[93] Y. Zhang, X. Yu, Z. Cui, S. Wu, Z. Wen, and L. 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* (D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, eds.), (Online), pp. 334–339, Association for Computational Linguistics, July 2020.

[94] Y.-H. Huang, Y.-H. Chen, and Y.-S. Chen, "ConTextING: Granting document-wise contextual embeddings to graph neural networks for inductive text classification," in *Proceedings of the 29th International Conference on Computational Linguistics*, (Gyeongju, Republic of Korea), pp. 1163–1168, International Committee on Computational Linguistics, Oct. 2022.

[95] L. Lacasa, B. Luque, F. Ballesteros, J. Luque, and J. C. Nuño, "From time series to complex networks: The visibility graph," *Proceedings of the National Academy of Sciences*, vol. 105, no. 13, pp. 4972–4975, 2008.

[96] B. Luque, L. Lacasa, F. Ballesteros, and J. Luque, "Horizontal visibility graphs: Exact results for random time series," *Phys. Rev. E*, vol. 80, p. 046103, Oct 2009.

[97] E. Bozkurt and A. Ortega, "Non-negative kernel graphs for time-varying signals using visibility graphs," in *2022 30th European Signal Processing Conference (EUSIPCO)*, pp. 1781–1785, 2022.

[98] M. Gao and R. Ge, "Mapping time series into signed networks via horizontal visibility graph," *Physica A: Statistical Mechanics and its Applications*, vol. 633, p. 129404, 2024.

[99] W. Ren and N. Jin, "Vector visibility graph from multivariate time series: a new method for characterizing nonlinear dynamic behavior in two-phase flow," *Nonlinear Dynamics*, vol. 97, pp. 2547–2556, July 2019.

[100] T. Alikasifoglu, *GRAPH FRACTIONAL FOURIER TRANSFORM*. PhD thesis, bilkent university, 2024.

[101] A. Sandryhaila and J. Moura, "Discrete signal processing on graphs," *IEEE Trans. on Sig. Proc.*, vol. 61, no. 7, pp. 1644–1656, 2013.

[102] D. I. Shuman, S. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Sig. Proc. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[103] A. Ortega, P. Frossard, J. Kovačević, J. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[104] R. Ragesh, S. Sellamanickam, A. Iyer, R. Bairi, and V. Lingam, "HeteGCN: Heterogeneous graph convolutional networks for text classification," WSDM '21, (New York, NY, USA), pp. 860–868, Association for Computing Machinery, 2021.

[105] T. K. Rusch, M. M. Bronstein, and S. Mishra, "A survey on oversmoothing in graph neural networks," 2023.

[106] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," 2019.

[107] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "SpanBERT: Improving pre-training by representing and predicting spans," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 64–77, 2020.

[108] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in *Proceedings of the 2019 Conference on*

Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 11 2019.

[109] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, p. II–1188–II–1196, JMLR.org, 2014.

[110] W. Hersh, C. Buckley, T. J. Leone, and D. Hickam, "Ohsumed: An interactive retrieval evaluation and new large test collection for research," in *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, (Berlin, Heidelberg), pp. 192–201, Springer-Verlag, 1994.

[111] D. Padmanabhan, S. Bhat, S. K. Shevade, and Y. Narahari, "Topic model based multi-label classification from the crowd," *CoRR*, vol. abs/1604.00783, 2016.

[112] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," in *Proceedings of EMNLP*, pp. 79–86, 2002.

[113] K. Lang, "Newsweeder: Learning to filter netnews," in *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 331–339, 1995.

[114] R. Dror, G. Baumer, S. Shlomov, and R. Reichart, "The Hitchhiker's guide to testing statistical significance in natural language processing," in *Proceedings of the 56th Annual Meeting of the ACL (Volume 1: Long Papers)*, (Melbourne, Australia), pp. 1383–1392, Association for Computational Linguistics, July 2018.

[115] A. Jung, *Machine Learning: The Basics*. Springer, 2022.

[116] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Int. Conf. on Learn. Repr. (ICLR)*, 2019.

[117] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," *arXiv*, 2017.

[118] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. of the 31st Int. Conf. on Neural Info. Proc. Sys. (NIPS)*, 2017.

[119] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *Int. Conf. on Learn. Repr. (ICLR)*, 2018.

[120] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of EMNLP*, pp. 1631–1642, 2013.

[121] A. Warstadt, A. Singh, and S. R. Bowman, "Neural network acceptability judgments," *arXiv preprint arXiv:1805.12471*, 2018.

[122] K. Wang, S. C. Han, S. Long, and J. Poon, "ME-GCN: Multi-dimensional edge-embedded graph convolutional networks for semi-supervised text classification," in *Int. Conf. on Learn. Repr. (ICLR)*, 2022.

[123] Y. Zhao and X. Song, "TextGCL: Graph contrastive learning for transductive text classification," in *2023 Int. Joint Conf. on Neural Networks (IJCNN)*, 2023.

[124] N. van der Heijden, E. Shutova, and H. Yannakoudakis, "K-hop neighbourhood regularization for few-shot learning on graphs: A case study of text classification," in *Proc. of the 17th Conf. of the European Chapter of the Assoc. for Comp. Ling. (EACL)*, 2023.

[125] Z. Ting-Ting, J. Ning-De, G. Zhong-Ke, and L. Yue-Bin, "Limited penetrable visibility graph for establishing complex network from time series," *Acta Physica Sinica*, vol. 61, no. 3, p. 030506, 2012.

[126] A. Warstadt, A. Singh, and S. R. Bowman, "Neural network acceptability judgments," *arXiv preprint 1805.12471*, 2018.

[127] W. B. Dolan and C. Brockett, "Automatically constructing a corpus of sentential paraphrases," in *Proceedings of the International Workshop on Paraphrasing*, 2005.

[128] Z. Chen, H. Zhang, X. Zhang, and L. Zhao, "Quora question pairs," 2017.

[129] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, (Vancouver, Canada), pp. 1–14, Association for Computational Linguistics, Aug. 2017.

[130] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of NAACL-HLT*, 2018.

[131] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of EMNLP*, pp. 2383–2392, Association for Computational Linguistics, 2016.

[132] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, (Brussels, Belgium), pp. 353–355, Association for Computational Linguistics, Nov. 2018.

[133] I. Dagan, O. Glickman, and B. Magnini, "The PASCAL recognising textual entailment challenge," in *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pp. 177–190, Springer, 2006.

[134] R. Bar Haim, I. Dagan, B. Dolan, L. Ferro, D. Giampiccolo, B. Magnini, and I. Szpektor, "The second PASCAL recognising textual entailment challenge," 2006.

[135] D. Giampiccolo, B. Magnini, I. Dagan, and B. Dolan, "The third PASCAL recognizing textual entailment challenge," in *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pp. 1–9, Association for Computational Linguistics, 2007.

[136] L. Bentivogli, I. Dagan, H. T. Dang, D. Giampiccolo, and B. Magnini, "The fifth PASCAL recognizing textual entailment challenge," 2009.

[137] H. J. Levesque, E. Davis, and L. Morgenstern, "The Winograd schema challenge," in *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, vol. 46, p. 47, 2011.

[138] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[139] I. Dewancker, M. McCourt, and S. Clark, "Bayesian optimization primer," 2015.

Appendix A

Source Code

The codebase supporting this thesis is organized across several GitHub repositories. Details of each repository are provided below:

- For GRTE, the code and related experiments are available at https://github.com/koc-lab/grte.
- The repository for Text-RGNNs, including its implementation and experimental setup, can be accessed at https://github.com/koc-lab/text-rgnn.
- VISPool's definition and corresponding experiments are hosted at https://github.com/koc-lab/vispool.