Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo sobre Redes Neurais de Grafos Bipartidos com Palavra-Chave e Atenção para Classificação Transdutiva de Texto

Vitor V. Oliveira

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientador
Prof. Dr. Thiago de Paulo Faleiros

Brasília
2025

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Study on Bipartite Graph Neural Networks with Keyphrase and Attention for Transductive Text Classification

Vitor V. Oliveira

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientador
Prof. Dr. Thiago de Paulo Faleiros

Brasília
2025

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

VO48s

Vasconcelos de Oliveira, Vitor
Study on Bipartite Graph Neural Networks with Keyphrase and Attention for Transductive Text Classification / Vitor Vasconcelos de Oliveira; orientador Thiago de Paulo Faleiros. -- Brasília, 2025.
94 p.

Dissertação(Mestrado em Informática) -- Universidade de Brasília, 2025.

1. Redes de Grafos Bipartidos. 2. Redes Neurais de Grafos. 3. Aprendizado Transdutivo. 4. Processamento de Linguagem Natural. 5. Graph Attention Networks. I. de Paulo Faleiros, Thiago, orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo sobre Redes Neurais de Grafos Bipartidos com Palavra-Chave e Atenção para Classificação Transdutiva de Texto

Vitor V. Oliveira

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Prof. Dr. Thiago de Paulo Faleiros (Orientador)
CIC/UnB

Prof. Dr. Antonio Fernando Lavareda Jacob Júnior
CCT/UEMA

Prof. Dr. Díbio Leandro Borges
CIC/UnB

Prof. Dr. Rodrigo Bonifácio Almeida
Coordenador do Programa de Pós-graduação em Informática

Brasília, 20 de fevereiro de 2025

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Study on Bipartite Graph Neural Networks with Keyphrase and Attention for Transductive Text Classification

Vitor V. Oliveira

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Prof. Dr. Thiago de Paulo Faleiros (Orientador)
CIC/UnB

Prof. Dr. Antonio Fernando Lavareda Jacob Júnior
CCT/UEMA

Prof. Dr. Díbio Leandro Borges
CIC/UnB

Prof. Dr. Rodrigo Bonifácio Almeida
Coordenador do Programa de Pós-graduação em Informática

Brasília, 20 de fevereiro de 2025

Dedication

I dedicate this work to all my family, friends, and my girlfriend, who supported me at all times.

Acknowledgements

I thank the faculty of the Departments of Computer Science and Mathematics at the University of Brasília for all the teachings and contributions to my education. I am grateful to the *Fundação de Apoio à Pesquisa do Distrito Federal (FAPDF)* for the financial incentives that allowed me to participate in numerous educational and research projects throughout my undergraduate studies. I would like to thank my academic supervisor Thiago de Paulo Faleiros and my academic co-supervisor Ricardo Marcondes Marcacini, for all their understanding, welcome and support; and to all the collaborators who took part in this work for their disposition and teachings.

This work was carried out with the support of the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES)*, through the Access to the Periodicals Portal.

Resumo

Na atualidade, o Processamento de Linguagem Natural (NLP) evoluiu rapidamente em uma ampla gama de tarefas, especialmente graças aos avanços do Aprendizado de Máquina (ML) e do Aprendizado Profundo (DL) ao longo dos anos. No entanto, devido à alta complexidade e aos diversos pré-requisitos dessas tecnologias, as metodologias convencionais de classificação de texto de NLP geralmente exigem uma grande quantidade documentos rotulados e alto poder computacional. Este trabalho investiga três técnicas para abordar e solucionar esses desafios. Primeiramente e mais significativamente, está o uso de abordagens baseadas em grafos transdutivos para a tarefa de classificação de texto, visando reduzir a quantidade necessária de dados rotulados. Para este processo inicial, emprega-se o já renomado modelo de Graph Convolutional Networks (GCN) e o modelo mais contemporâneo de Graph Attention Networks (GAT), ambos utilizando uma nova estrutura de grafos bipartidos de documento-conceito que usam Keyphrases (conceitos) para aquisição de conhecimento de tópicos e enriquecimento de informações dos modelos. A segunda técnica utilizada, constitui na aplicação de coarsening para promover a redução dos grafos, reduzindo assim os custos computacionais. Por fim, emprega-se Large Language Models (LLM) como rotuladores de baixo custo, removendo ou reduzindo a necessidade de rotuladores humanos. Os resultados mostram que o modelo GAT teve o melhor desempenho para tarefas de classificação de texto transdutivo usando a abordagem de grafos bipartidos de documento-conceito, sendo este um desempenho comparável aos de modelos indutivos tradicionais mesmo usando apenas de 1 a 30 documentos rotulados por classe. Referente a aplicação de coarsening, ocorreu uma redução de 40%-50% no tamanho dos grafos preservando em média 82% do desempenho dos modelos, variando de 68% a 95% em vários conjuntos de dados. LLMs foram capazes de treinar vários modelos eficientes, mas quando comparados a modelos treinados usando dados rotulados por humanos apresentaram resultados inferiores, demonstrando que o aprendizado transdutivo favorece pequenas quantidades de dados muito precisos em alternativa de uma quantidade abundante de dados moderadamente precisos.

Palavras-chave: Redes de Grafos Bipartidos, Inteligência Artificial, Processamento de

Linguagem Natural, Aprendizado Transdutivo, Aprendizado Profundo, Aprendizado de Máquina, Grande Modelos de Linguagem, Redes Neurais de Grafos, Redes de Atenção de Grafos, Redes Convolucionais de Grafos, Documentos, Conceitos

Resumo Expandido

0.1 Introdução

O crescimento exponencial de dados textuais e a necessidade de organização automatizada de informações impulsionaram o desenvolvimento de técnicas de Processamento de Linguagem Natural (NLP). A classificação de textos surge como uma das tarefas fundamentais nesse campo, sendo crucial para aplicações como filtragem de e-mails, recomendação de conteúdo e análise de sentimentos. Contudo, abordagens baseadas em aprendizado supervisionado demandam grande quantidade de dados rotulados, o que nem sempre é viável economicamente ou logisticamente.

Neste cenário, abordagens semi-supervisionadas, especialmente aquelas com foco em aprendizado transdutivo, mostram-se promissoras. Elas são capazes de aproveitar a estrutura dos dados não rotulados para propagar informação a partir de um pequeno número de exemplos anotados. As Redes Neurais de Grafos (GNNs) têm ganhado destaque nesse contexto por permitirem operações em grafos que modelam relações entre documentos ou entidades semânticas.

Essa dissertação propõe uma arquitetura baseada em grafos bipartidos documento-conceito, onde os documentos são conectados a frases-chave extraídas automaticamente. Essa representação permite capturar relações semânticas de forma mais rica do que simples conexões entre documentos. A proposta também investiga o uso de mecanismos de atenção (GATs), técnicas de coarsening para redução de complexidade, e o uso de modelos de linguagem (LLMs) como rotuladores automáticos.

O objetivo central é demonstrar que é possível obter resultados comparáveis aos métodos tradicionais mesmo em contextos com poucos dados rotulados, por meio de técnicas que exploram a estrutura semântica dos textos e a propagação de informações com grafos. A hipótese é que modelos transdutivos usando grafos bipartidos enriquecidos com frases-chave e mecanismos de atenção podem superar limitações de métodos convencionais.

0.2 Fundamentação Teórica

0.2.1 Aprendizado Transdutivo Semi-Supervisionado

O aprendizado transdutivo diferencia-se do tradicional aprendizado indutivo por seu foco em inferir rótulos apenas sobre um conjunto conhecido de instâncias, em vez de construir um modelo generalizável para dados futuros. Essa característica o torna ideal para situações onde há abundância de dados não rotulados e escassez de dados anotados. O modelo transdutivo utiliza a totalidade dos dados disponíveis (rotulados e não rotulados) no processo de treinamento, permitindo uma melhor adaptação ao conjunto de dados específico.

Além disso, o aprendizado semi-supervisionado transdutivo permite a propagação da informação entre as amostras por meio de estruturas relacionais, como grafos. Isso é particularmente eficaz quando há uma forte estrutura de similaridade entre os dados, como ocorre em coleções de documentos inter-relacionados tematicamente. A combinação de poucos exemplos anotados com muitos exemplos não anotados, organizados em um grafo, fornece uma base rica para inferência.

0.2.2 Grafos Bipartidos Documento-Conceito

Um grafo bipartido é uma estrutura que liga dois conjuntos distintos de nós, onde conexões só ocorrem entre elementos de conjuntos diferentes. Na abordagem proposta, os dois conjuntos representam documentos e conceitos (frases-chave) extraídos automaticamente. O uso de frases-chave como representações conceituais permite capturar melhor o conteúdo semântico de cada documento, ao contrário de modelos baseados exclusivamente em palavras.

Essas frases-chave são extraídas por meio da ferramenta KeyBERT, que utiliza embeddings do modelo BERT para identificar expressões semanticamente relevantes. A modelagem em grafos bipartidos possibilita a criação de redes mais densas e informativas, que refletem as relações latentes entre documentos e seus conceitos compartilhados. Isso fortalece a capacidade dos modelos de capturar similaridades e diferenças semânticas de maneira eficaz.

0.2.3 Redes Neurais de Grafos: GCN e GAT

Redes Neurais de Grafos (GNNs) são modelos de aprendizado profundo que operam diretamente sobre estruturas de grafos, aprendendo representações dos nós com base em suas conexões e vizinhanças. As Redes Convolucionais de Grafos (GCN) são um tipo de GNN

que realizam agregações simples de vizinhança, considerando todos os nós vizinhos com pesos iguais e agregando suas informações.

Essa abordagem, embora eficiente, pode não capturar adequadamente a importância relativa dos vizinhos. Em contraste, as Graph Attention Networks (GAT) incorporam mecanismos de atenção que atribuem pesos distintos para cada vizinho, permitindo que o modelo aprenda quais conexões são mais relevantes para a tarefa.

O uso de GATs em grafos bipartidos documento-conceito permite, por exemplo, que a rede aprenda que um conceito como "aprendizado profundo" é mais representativo para a classificação de um documento técnico do que um termo genérico como "dados". Essa capacidade de diferenciação é essencial em contextos com grande variação semântica.

0.2.4 Redução de Grafos (Coarsening)

Com o aumento do número de documentos e conceitos, os grafos podem se tornar densos e computacionalmente dispendiosos. Para contornar esse problema, utiliza-se a técnica de coarsening, que agrupa nós semelhantes em supernós. O algoritmo CLPb, empregado neste estudo, aplica propagação de rótulos semi-síncrona para identificar e colapsar grupos semanticamente similares.

A aplicação do coarsening não apenas reduz o tamanho do grafo como também preserva de maneira eficaz as propriedades estruturais necessárias para a inferência. Assim são reduzidos custos computacionais e de processamento, mas mantendo a performance dos modelos de grafos semlhantes.

0.2.5 Modelos de Linguagem como Rotuladores

Os Modelos de Linguagem de Grande Escala (LLMs), como o LLaMA 3 (8B), têm sido comumente empregados como reguladores automáticos na literatura. Eles são capazes de gerar rótulos com base em poucos exemplos ou até mesmo em configurações de zero-shot learning. Nessa dissertação, utiliza-se diferentes configurações de prompts para avaliar o desempenho desses modelos na rotulagem inicial de documentos.

Apesar de apresentarem desempenho inferior ao de rótulos humanos em média, os LLMs mostraram-se úteis para gerar conjuntos de dados iniciais, especialmente em tarefas repetitivas e contextos onde não há disponibilidade de especialistas. A estratégia de usar LLMs como primeiro estágio de rotulagem pode ser combinada com validação humana posterior para melhores resultados.

0.3 Metodologia

O trabalho segue uma metodologia composta por etapas encadeadas que contemplam desde a coleta dos dados até a avaliação dos modelos, e composta por sete etapas principais:

1. Seleção dos conjuntos de dados
2. Extração de frases-chave com KeyBERT
3. Construção dos grafos bipartidos documento-conceito
4. Modelagem com GCNs e GATs
5. Aplicação do algoritmo CLPb para coarsening
6. Rotulagem automática com LLMs
7. Avaliação com base em métricas padrão como F1-score, precisão e revocação

Inicialmente, foram selecionados conjuntos de dados de diferentes tamanhos e domínios, visando garantir a diversidade e a validade dos experimentos. A extração de frases-chave foi realizada utilizando a ferramenta KeyBERT, que gera representações semânticas dos documentos com base em embeddings BERT.

Com as frases-chave extraídas, os grafos bipartidos são construídos conectando cada documento às suas frases-chave correspondentes. Essa representação em grafo permite explorar conexões indiretas entre documentos por meio dos conceitos compartilhados. A etapa seguinte consiste na aplicação de Redes Neurais de Grafos. Dois modelos são utilizados: GCN (Graph Convolutional Network) e GAT (Graph Attention Network). Ambos processam os dados transdutivamente, utilizando rótulos limitados para propagar informações no grafo.

Paralelamente, é aplicado o algoritmo CLPb para realização de coarsening nos grafos. Essa técnica agrupa nós semelhantes, gerando grafos mais compactos com menor custo computacional. A eficiência dessa técnica é avaliada em termos de desempenho e economia de recursos.

Outra etapa relevante é a utilização de modelos de linguagem como rotuladores. O modelo LLaMA 3 foi empregado para gerar rótulos automaticamente com base em prompts definidos para as tarefas. Os modelos GNN foram treinados utilizando rótulos humanos e rótulos gerados por LLMs, permitindo comparação direta de desempenho.

Por fim, a avaliação dos modelos foi feita utilizando métricas padrão principalmente o F1-score. Os experimentos foram repetidos com diferentes quantidades de documentos rotulados por classe (1, 5, 10, 20, 30), buscando entender a influência da quantidade de supervisão sobre o desempenho.

0.4 Resultados e Discussão

Os resultados evidenciaram que os modelos baseados em GATs superaram os GCNs em praticamente todos os cenários. A capacidade dos GATs de ponderar a importância de diferentes vizinhos demonstrou-se crucial para o sucesso em tarefas com poucos rótulos. Em particular, a diferença de desempenho foi acentuada quando o número de documentos rotulados era reduzido.

A aplicação da técnica de coarsening também se mostrou eficiente, reduzindo em até 50% a quantidade de nós e arestas sem impacto significativo no desempenho. A média de preservação de desempenho observada foi de 82%, o que representa um excelente compromisso entre eficiência e acurácia. A técnica mostrou-se particularmente útil em cenários com grandes volumes de dados não rotulados. Em alguns casos, os modelos coarsenados apresentaram desempenho praticamente equivalente aos modelos completos, com vantagens claras em termos de eficiência computacional.

Nos experimentos com LLMs como rotuladores, observou-se que embora o desempenho seja inferior à anotação humana, os resultados foram aceitáveis. Em várias configurações, os rótulos gerados por LLMs possibilitaram o treinamento de modelos GNN com desempenho competitivo, especialmente quando os dados apresentavam classes distintas e bem definidas, além de documentos relativamente padronizados.

0.5 Conclusão

Essa dissertação apresenta uma proposta inovadora e eficaz para a classificação de textos com poucos dados rotulados, combinando grafos bipartidos, GNNs com atenção, coarsening e rotulagem via LLMs. A abordagem mostra-se especialmente adequada para cenários com limitações de anotação manual, com aplicações em contextos reais e escaláveis.

A combinação de frases-chave como intermediários semânticos, atenção contextual e simplificação estrutural oferece uma alternativa robusta aos modelos indutivos tradicionais. A integração de LLMs como rotuladores de baixo custo e a utilização de coarsening também representam um diferencial importante em termos de acessibilidade e disponibilidade.

O estudo abre portas para pesquisas futuras que explorem técnicas semelhantes em outras tarefas de NLP, bem como a adaptação da metodologia para outros idiomas, domínios e difenretes arquiteturas de grafos heterogêneos.

Abstract

In contemporary times, Natural Language Processing (NLP) has swiftly evolved in a wide range of tasks, especially thanks to Machine Learning (ML), and Deep Learning (DL) great advancements over the years. However, due to these technologies' complexity and data prerequisites, current conventional NLP text classification methodologies often require large numbers of labeled documents and large computational power. This work mainly investigates three techniques to address such challenges. Firstly and most significantly, the use of transductive graph-based approaches for the text classification task aims to reduce the amount of required labeled data. For this initial process, we employ the classic and well-established Graph Convolutional Networks (GCN) and the more contemporary Graph Attention Networks (GAT), on a novel document-concept bipartite graph framework that uses Keyphrase(concepts) for topic knowledge acquisition and model information enrichment. The second technique is applying coarsening for graph reduction, hence reducing computational costs. Lastly, we aim to employ Large Language Models (LLM) as low-cost labelers effectively removing or reducing the need for human labelers. Results show GAT as the best performing model for transductive text classification tasks using the document-concept bipartite graph approach, GAT showed that it can perform on equal levels to traditional inductive models despite using only 1 to 30 labeled documents per class. The coarsening application presented 40%-50% graph size reduction while maintaining 82% of the model performance at average, ranging from 68% to 95% on various datasets. LLMs were able to train several efficient models, but compared to models trained on human-labeled data revealed inferior results, demonstrating that transductive learning favors small amounts of highly accurate data rather than a large quantity of moderately accurate data.

Keywords: Bipartite Graph networks, Artificial Intelligence, Natural Language Processing, Transductive Learning, Deep Learning, Machine Learning, Large Language Models, Graph Neural Networks, Graph Attention Networks, Graph Convolutional Networks, Transformers, Self-Attention, Documents, Concepts

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence.

BERT Bidirectional Transformers.

BILSTM Bidirectional Long Short-Term Memory.

CD Critical Difference.

CLPb Coarsening via semi-synchronous Label Propagation for bipartite networks.

CLPk Coarsening via semi-synchronous Label Propagation for K-partite networks.

CNN Convolutional Neural Networks.

DL Deep Learning.

GAT Graph Attention Networks.

GCN Graph Convolutional Networks.

GNN Graph Neural Networks.

GPT Open AI's Generative Pre-Training Transformer model.

GQA Grouped-Query Attention.

KNN K-nearest neighbors.

LLM Large Language Models.

ML Machine Learning.

MMR Maximal Margin Relevance.

NLP Natural Language Processing.

PDE Partial Differential Equations.

RoPE Rotary Posi-ional Embeddings.

Chapter 1

Introduction

In contemporary times, Artificial Intelligence (AI) systems often rely on Machine Learning (ML) algorithms. Artificial Intelligence characterizes the capability of these systems to engage in advanced problem-solving, similar to human capacities. Machine Learning, on the other hand, describes the computer's ability to learn from specific training data and effectively solve distinct problems and tasks. Within the field of ML, Deep Learning (DL) emerges as a subset, representing the progression of artificial neural networks towards increasingly intricate architectures with enhanced learning capabilities [Janiesch et al., 2021].

Natural Language Processing (NLP) stands as a vast branch of computer science, embracing artificial intelligence and linguistics, dedicated to exploring the capabilities and constraints of computers in comprehending human language. The evolution of NLP has been swift in recent years, leveraging multiple approaches from Artificial Intelligence, Machine Learning, and Deep Learning to process, analyze, and represent human natural language texts. Its main objectives focus on solving tasks such as segmentation, information extraction, summarization, translation, and text or speech classification, among others [Otter et al., 2019]. Figure 1.1 represents how NLP is part of AI and linguistics.

Within the NLP domain, text classification is pivotal in effectively handling, retrieving, and extracting knowledge from extensive repositories of textual documents. Conventional methodologies for text classification often rely on inductive learning algorithms [Weiss et al., 2012, Sebastiani, 2002]. These algorithms include classification models capable of categorizing new or unseen texts based on prior observations and examples.

However, these conventional methodologies frequently demand many labeled documents to train accurate classification models. This process can generate considerable costs due to using traditional labeling methods. Labeling, in the context of Machine Learning and Natural Language Processing, involves assigning labels to raw data to provide context or meaning. For instance, in classification tasks, labeling entails establishing

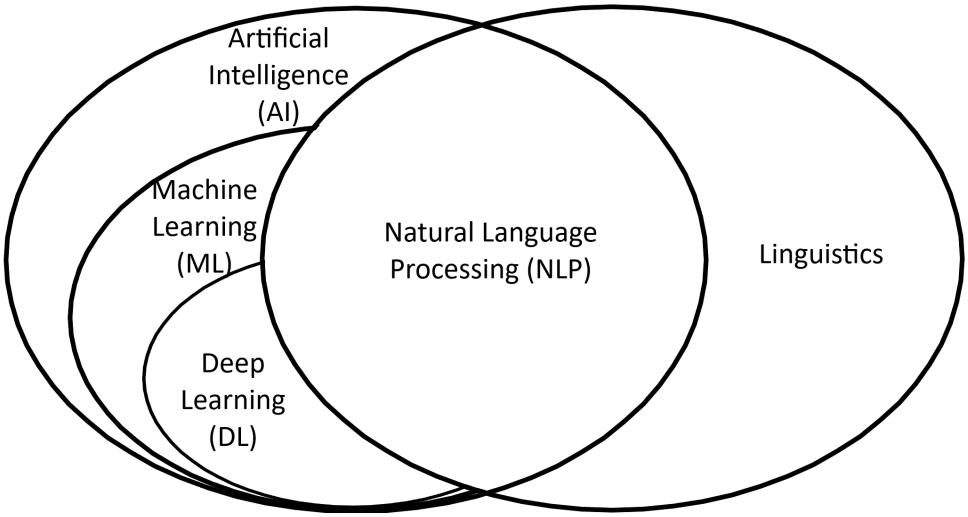


Figure 1.1: Representation of Natural Language Processing in the academic realm of artificial intelligence and linguistics.

classes for data instances [Li et al., 2022].

Typically, human labeling is the prevalent method in state-of-the-art datasets for NLP tasks, relying on human effort to manually annotate and categorize texts. This approach places the responsibility on labelers to create a robust dataset for training Deep Learning models, involving the labor-intensive process of reading, searching, identifying, circumscribing, and reviewing texts. While human labeling tends to yield more reliable and accurate datasets, its high costs attributed to human effort in repetitive tasks, time consumption, capital expenses, and the need for additional reviews to mitigate human errors are a significant impediment to creating training datasets [Zhang et al., 2021].

To overcome this cost limitation, there is a growing interest in employing methods that leverage the abundance of unlabeled texts to facilitate and enhance text classification, especially in semi-supervised methods such as the transductive learning methodology. In addition, with the growth of Deep Learning and Large Language Models (LLM), NLP-specific algorithms that use transformer models and massive datasets, the exploration of utilizing such large models as economical labelers is also becoming increasingly relevant.

Furthermore, graph-based approaches position themselves as a viable solution to handle the intricate relationships between words and documents and effectively explore contextual-aware word relations in text classification [Yao et al., 2018, Veličković et al., 2017, Zhang and Zhang, 2020]. Graphs offer a natural way to represent the connections and dependencies between elements in a document collection, enabling a higher understanding of relationships and propagating information across the graph structure, a valuable characteristic for transductive learning. Here, techniques related to graph reduction, such as coarsening, can be employed to improve computational efficiency.

Graphs have stood out as a robust data structure in various domains, especially machine learning and data mining. Their robust mathematical properties and ability to capture complex relationships between entities make them a natural choice for semi-supervised problems. One of the main advantages of using graphs is their ability to model relationships between different data elements intuitively and effectively, which is essential in scenarios where data may be partially labeled, like semi-supervised problems, or where the underlying structure of the data is complex and non-linear.

There are several approaches to converting textual data into graph structures to represent text in graphs. These approaches include representations based on word co-occurrence, topic models, and semantic networks [Wang et al., 2021, Li et al., 2023]. A natural and intuitive way to represent texts as graphs is through bipartite graphs, which usually establish relationships between documents and words or document features.

1.1 Hypotheses

Our main hypothesis is that this representation of bipartite graphs, especially in our own developed document-concept graphs applied by graph-based machine learning models, is suitable for transductive learning, a semi-supervised approach, and can effectively capture semantic relationships. The document-concept representation is a bipartite graph that creates edges between documents and their concepts, the main topics extracted from that document. In our experiments, these concepts are also named keyphrases, specific phrases that represent the main ideas of the document. Figure 1.2 represents a simple example of a bipartite document-concept graph.

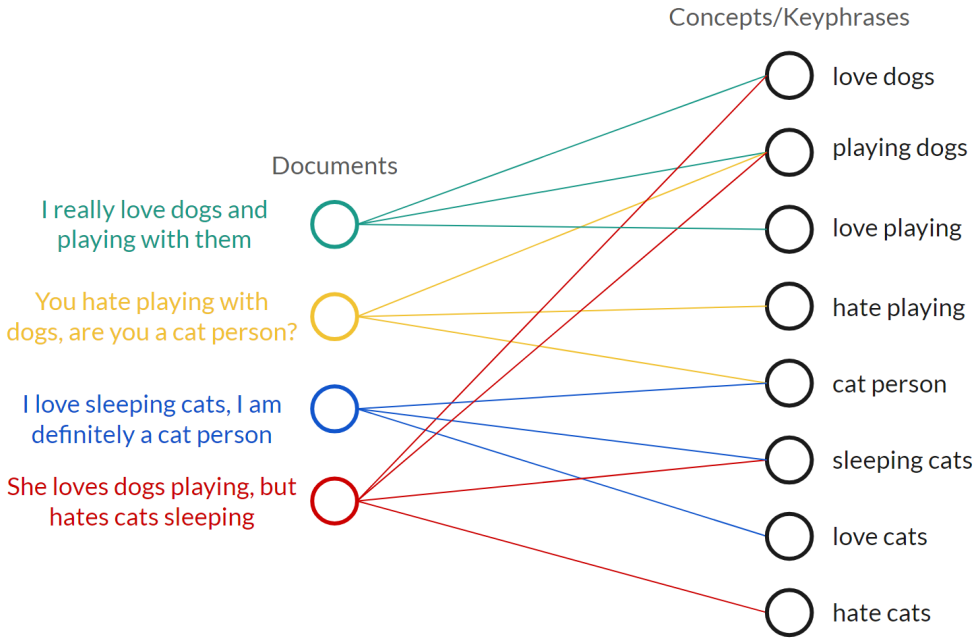


Figure 1.2: Bipartite document-concept graph example.

We believe these document-concept bipartite graphs can be effectively used for cases with high computational or data-labeling costs and further supported by other techniques. Given limited computational resources, graph coarsening techniques can be helpful since reducing graph size can drastically reduce memory usage, GPU usage, and training time. Semi-supervised learning can significantly improve the quality of predictions for scenarios with few available labels, still, Large Language Models can enrich data representation and aid in classification, removing the need for human labeling and reducing annotation costs by using LLM-labeling.

1.2 Objectives

Our research's main objective derives from our hypothesis on document-concept bipartite graphs. We aim to investigate the uses of our designed document-concept representation for semi-supervised learning, focusing on improving model performance regarding memory consumption and accuracy. To achieve this, we further delve into specific objectives such as:

- Propose a novel approach that combines concept-based representation with Graph Attention Networks (GATs) for semi-supervised text classification on bipartite graphs. This approach allows the model to automatically learn the importance of concepts in the document classification process. Also, employ the document-concept approach

on traditional Graph Convolutional Networks (GCN) used as a well-known baseline model for transductive graph learning.

- Apply and evaluate the coarsening method on the generated graphs to reduce computational costs and verify the coarsening impact on model performance.
- Investigate Large Language Models as additional information sources or low-cost labelers, replacing or supporting human annotations and promoting model efficiency improvements.
- Extensively evaluate all the proposed methods on multiple benchmark datasets, considering different cardinalities of concepts extracted from the documents (various keyphrase word sizes) and different semi-supervised scenarios. Demonstrate the effectiveness of our approach and compare it against other state-of-the-art methods.

1.3 Section Summary

Further on in this text, we delve into the development[1][2] of our Reseach. The chapters of this dissertation can be summarized by the topics below:

- Introduction: Motivation for our research and hypotheses for investigating bipartite semi-supervised document-concept graphs, coarsening, and LLMs in Graph Neural Networks.
- Background: Base contextualization on terms, concepts, fundamental ideas, and the main methodologies used during our research.
- Related Work: Presents a brief literature review, highlighting relevant related work, that originated some of our approaches, and used as inspiration.
- Methodology: Formalization of the procedures adopted in the application of each proposed approach
- Experimental Results: Our study's experiments and results, used to evaluate the proposed approaches' performance and usability.
- Conclusion: Concludes our study by recapitulating the main objectives, achieved results, limitations, and future work.

[1] All code used for this is study is available on the Github repository: `https://github.com/VitorVVO/Study_on_Trandutivs_Bipartite_Graph_Neural_Networks.git`.

[2] All datasets used for this study are available on the Github repository: `https://github.com/rmarcacini/text-collections/tree/master/complete_texts_csvs`.

Chapter 2

Background

2.1 Semi-supervised Transductive Learning

Semi-supervised learning occupies a position between supervised learning, which relies on labeled data for model training, and unsupervised learning, which constructs models without using labeled data. By incorporating information from both labeled and unlabeled data, semi-supervised methods are highly effective at utilizing significant volumes of unlabeled texts to augment the classification process [Kong et al., 2013, Faleiros et al., 2017]. For instance, In the context of semi-supervised text classification, the objective is to make optimal use of available labeled data to supply essential context and generate accurate predictions for each input sample. Simultaneously, using unlabeled data aids the model in assessing the data's inherent structure, enabling more efficient generalization to new input samples.

Transductive classification, a subtype of semi-supervised learning, focuses on directly predicting labels for unlabeled instances without constructing a generalization model specifically for classifying new texts [de Paulo Faleiros et al., 2017]. Unlike inductive learning, where the model generalizes from labeled examples to make predictions on unseen data, transductive learning involves observing all data beforehand. Its primary objective is to predict labels for the unlabeled instances (testing dataset) based on the labeled instances (training dataset) and all additional information present in the combined data during the learning process (testing+training dataset) [Gammerman et al., 2013]. In essence, inductive learning aims to derive a general function for solving a specific problem. In contrast, transductive learning seeks to develop a specific function tailored to the problem at hand [Tripodi and Pelillo, 2017]. This transductive approach is especially beneficial when labeled training data is limited.

For instance, consider a dataset $X = X_l \cup X_u$, which is composed by $X_l = x_1, x_2, ..., x_n$ that represents labeled instances and $X_u = x_{u+1}, x_{ui+2}, ..., x_k$ that represents labeled in-

stances. Our objective is to find an output $Y_u = y_1, y_2, ..., y_k$ for each unlabeled instance of our dataset. In a classic inductive approach, we use X_l to learn a function $f : X \rightarrow Y$ that minimizes a loss function L which measures the discrepancy between the predicted and real labels. Formalizing, inductive learning training uses the labeled dataset (X_l) as input to f and predicts Y_p:

$$Y_p = f(X_l) \tag{2.1}$$

the loss function L compares predicted data ($Y_p \equiv f(X_l)$) and real data(Y_l) to provide model training directions:

$$minL(Y_p, Y_l) \equiv minL(f(X_l), Y_l) \tag{2.2}$$

By the end of the training, function f can effectively predict unknown/unlabeled data. This is the testing, used to evaluate f performance on its predictions:

$$f(X_u) = Y_u \tag{2.3}$$

Transductive learning works similarly, as it also aims to minimize the L loss of function, using it as the training guide, but to achieve this, it considers both X_l and X_u, essentially training and testing simultaneously [de Paulo Faleiros et al., 2017]:

$$Y_p = f(X_l, X_u) \tag{2.4}$$

$$minL(Y_pY_u, Y_lY_u) \equiv minL(f(X_l, X_u), Y_lY_u) \tag{2.5}$$

A great benefit of leveraging unlabeled x_u and labeled X_l data for training is that it enables transductive learning to use very small portions of labeled data compared to inductive learning. But there are disadvantages as well, in the case of unknown data X_u^2, that is not X_u, we must retrain our function f, while an inductive model would not need retraining only applying $f(X_u^2)$ to retrieve predictions. Figure 2.1 illustrates this transductive and inductive learning training comparison.

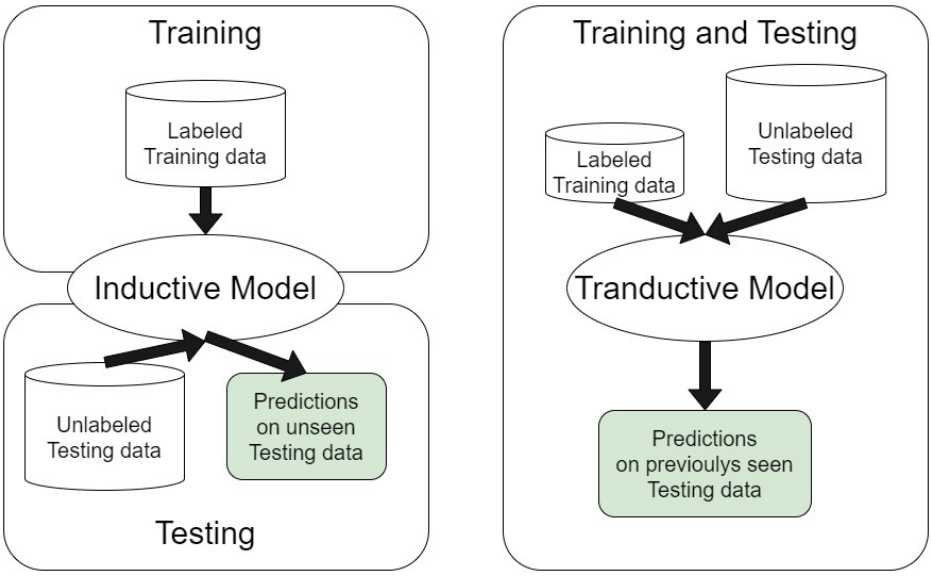


Figure 2.1: Transductive learning and inductive learning comparison.

2.2 Document-concept Bipartite Graphs and Keyphrase Extraction

2.2.1 Document-concept Bipartite Graphs

A bipartite graph is a type of graph whose nodes can be divided into two disjoint sets such that no two nodes within the same set are adjacent. Formalizing, consider a graph represented as $G = (N, E)$, where N is the set of nodes (i.e., vertices) and $E \subseteq N \times N$ is the set of edges. G is considered a bipartite graph only if:

- N can be partitioned into two disjoint sets N_1 and N_2, such that $N_1 \cap N_2 = \emptyset$ and $N_1 \cup N_2 = N$;
- No edge $e = (a, b) \in E$ between node a and b exists where both $a \wedge b \in N_1$ or $a \wedge b \in N_2$.

A common NLP approach using bipartite graphs is the document-word bipartite graph [Saifuddin et al., 2021], where one node layer represents the document texts, and the other layer represents the words present in each document. The edges represent which document contains which word.

A document-concept bipartite graph, our selected approach, is another type of bipartite graph less commonly used in natural language processing where one set of nodes represents documents, and the other set represents concepts, also named keyphrases or

keywords. The edges between the two sets represent the relationship between the documents and the concepts they contain [Zhang et al., 2011, RK Rao and Devi, 2017]. Adapting the bipartite graph formalization previously established to a document-concept bipartite graph, $N_1 = D$ would represent documents nodes and $N_2 = C$ concepts nodes in a form that $N = D \cup C$.

2.2.2 Concepts and Keyphrases

Concepts are abstract representations encapsulating a document's core ideas or topics, providing a high-level understanding of its content. In document analysis and information retrieval, concepts can be equated to keyphrases, representing the most meaningful and descriptive multi-word terms extracted from a text.

Keyphrases serve as concise, targeted summaries of a document's subject, facilitating efficient information retrieval, topic modeling, and visualization in knowledge graphs. Keyphrases are distinguished from keywords in that they consist of multi-word lexemes (e.g., "artificial intelligence" or "data mining") rather than single words (e.g., "artificial" or "data"). While keywords offer a simplified representation, they often lack the specificity and semantic depth needed to convey nuanced ideas [Siddiqi and Sharan, 2015, Beliga, 2014].

By focusing on concepts as keyphrases, we can reduce the complexity of document representations, effectively minimizing the number of concept nodes in a graph while preserving semantic richness. This approach contrasts with word-based representations, which can be overly granular and less meaningful in capturing the thematic essence of a document.

2.2.3 Keyphrases Extraction

A broad range of text mining techniques are available regarding keyphrase extraction [Papagiannopoulou and Tsoumakas, 2019]:

- Unsupervised Methods:
  - Statistics-based: Uses statistical metrics like TF-IDF, phrase frequency, or co-occurrence statistics to rank phrases.
  - Graph-based: Builds a graph where nodes represent candidate terms and edges indicate relationships (e.g., co-occurrence).
  - Embedding-based: Leverages word or phrase embeddings (e.g., Word2Vec, GloVe, BERT) to compute semantic similarity between phrases and the document.

  - Language Model-based: Applies N-gram or neural language models to evaluate the likelihood of phrases being keyphrases based on their context.

- Supervised Methods:
  - Traditional Supervised: Frames keyphrase extraction as a binary classification task. Uses features like term frequency, position, and linguistic patterns (e.g., part-of-speech).
  - Deep Learning-based: Utilizes advanced neural networks, such as encoder-decoder frameworks (e.g., CopyRNN) or sequence labeling models (e.g., Bi-LSTM-CRF).

While traditional methods for concept extraction often rely on statistical measures such as term frequency-inverse document frequency (TF-IDF) [Salton and Yang, 1973] or graph-based algorithms like TextRank [Mihalcea and Tarau, 2004], our approach is based on language models such as BERT (Bidirectional Encoder Representations from Transformers). These models have demonstrated remarkable capabilities in understanding the semantics of text. Using contextual information and word relationships, BERT-based models can be leveraged to extract representative concepts from documents.

Our work mainly focuses on the methodology proposed by Grootendorst et al. [Grootendorst, 2020], the KeyBERT framework. Two works heavily inspired KeyBERT. Sharma et al. [Sharma and Li, 2019] proposed an approach that incorporates contextual and semantic features, using Bidirectional Transformers (BERT) for feature extraction, cosine similarity for candidate keyword extraction, and a Bidirectional Long Short-Term Memory (BILSTM) model, trained based on BERT embeddings similarity, for keyphrase classification.

The second guiding work was Smires et al.'s EmbedRank [Bennani-Smires et al., 2018], an unsupervised keyphrase extracting approach that also leverages sentence embeddings but embeds both the document and candidate phrases into the same vector space to conduct semantic similarity measurements. Keyphrases are ranked using the Maximal Margin Relevance (MMR) algorithm using cosine similarity. MMR balances relevance (similarity to the document) and diversity (dissimilarity among candidates) by selecting phrases that maximize a trade-off parameter, ensuring the extracted keyphrases are informative and non-redundant.

The KeyBERT framework [Grootendorst, 2020] was modeled after [Bennani-Smires et al., 2018] EmbedRank, mainly the MMR algorithm. KeyBERT functions by first extracting document embeddings with BERT to get a numerical document-level representation, the document embeddings similar to [Sharma and Li, 2019]. Subsequently, BERT is also used to extract word embeddings, resulting in N-gram phrases based on all

word combinations comprising N words. Finally, simple cosine similarity or the MMR algorithm is used to find the phrases most similar/relevant to the document embedding. The most similar words could then be identified as the words that best describe the entire document.

2.3 Graph Neural Networks (GNNs)

Graph Neural Networks have gained considerable attention in machine learning for their effectiveness in handling structured data, including text data. First introduced by Gori et al. in 2005 [Gori et al., 2005], GNNs are a deep learning model designed to learn and reason about data represented in graph structures. Graphs consist of nodes and edges, and GNNs excel at capturing complex relationships within such data. Using their message-passing mechanism to aggregate information from neighboring nodes, GNNs can learn meaningful representations that capture the underlying structure and dependencies in the data [Khemani et al., 2024].

In a traditional GNN, nodes are defined by their features and connected nodes [Scarselli et al., 2008]. Given a graph, as mentioned previously, $G = (N, E)$ where N and E are the nodes and edges sets, respectively, the graph network aims to learn a hidden state embedding $h_n \in \mathbb{R}_s$ that encapsulates each node's features and neighborhood data in the form of a vector.

The first use of this vector h_n is aggregating information of neighbors and updating itself based on such information by using GNN's message passing mechanism, illustrated in figure 2.2. The kind of "information" this message passes consists of two main parts: structural information about the graph (i.e., degree of nodes, etc.) and the other is feature-based (i.e., nodes features or characteristics).

This h_n update process occurs interactively. At each iteration, each node strictly circles the neighborhood and collects information. After the first iteration (k = 1), each node embedding expressly retains information from its one-hop neighborhood. After the second iteration (k = 2), each node retains data from not only its neighbors but also its neighbors' neighbors who passed through their vector updated previously. To summarize, the messages coming from neighbors are based on information aggregated from their respective neighborhoods. Figure 2.2 represents the message-passing mechanism.

Another, and the most relevant, application of this h_n embedding is in predicting the expected node label. The state embedding h_n, represented as an s-dimensional vector associated with a node $n \in N$, can generate an output Y_{On}, effectively determining the node's envisioned label. This anticipated distribution of the node label, denoted as Y_{On}, is

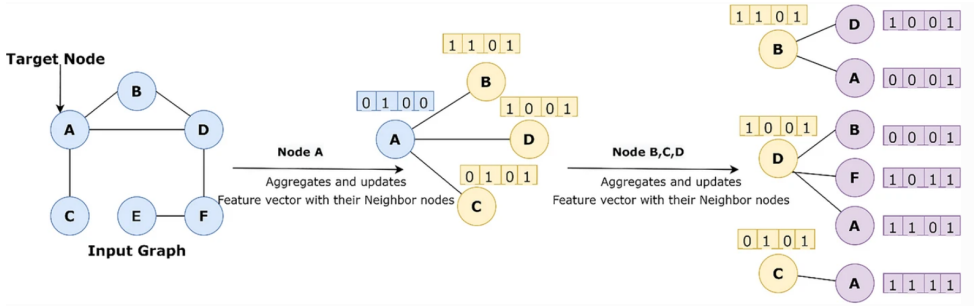


Figure 2.2: GNN aggregation/message passing example. Credits to [Khemani et al., 2024]

constructed based on the information encoded in the state embedding h_n and also based on state embeddings of neighboring nodes $h_{nb1}, h_{nb2}...h_{nbn}$.

One of the key strengths of GNNs lies in their versatility across various learning strategies [Waikhom and Patgiri, 2021]. GNNs can have many variants of graphs (e.g., Directed Graph, Bipartite Graph, Heterogeneous Graph, Dynamic Graph), levels of classification tasks (e.g., node, edge, and graph level tasks), and graph-based learning techniques (e.g., supervised, unsupervised, and semi-supervised each with many sub-techniques within itself). In our research, we employ a bipartite heterogeneous semi-supervised node classification approach. This method involves two distinct types of nodes, our objective being to classify one of these node types using transductive learning techniques.

Regarding transductive learning, Graph Neural Networks can capitalize significantly on their advantages, especially due to their inherent ability to facilitate the flow and dissemination of information across a graph through relationships among patterns [Ciano et al., 2022]. The message-passing mechanism, Figure 2.2, inherent in GNNs, proves valuable in supplying the transductive algorithm with the necessary general understanding of the network to address the problem at hand effectively.

2.3.1 Graph Convolutional Network (GCN)

One traditional and popular variant in the GNN family is the Graph Convolutional Network [Kipf and Welling, 2016a], which is based on the well-known Convolutional Neural Networks (CNN), first introduced by Lecun et al. [Lecun et al., 1998]. The main idea of a CNN is to use filters to extract important features from the input data and reduce its dimensionality.

A CNN consists of one or more convolutional layers, followed by pooling layers and fully-connected layers. The convolutional layers apply sliding weights called filters or kernels to the input data and produce feature maps that capture the presence of specific features in specific data regions or groups. The pooling layers reduce the size of the

feature maps and retain the most important information. Lastly, the fully connected layers perform classification based on the extracted features.

CNNs and GCNs 'convolution' processes are, in essence, the same [Kipf and Welling, 2016a]. The input neurons are multiplied by filters. The filters act as a sliding window across the data, allowing the network to learn information from nearby cells. Weight sharing applies the same filter within the same layer throughout the data. For instance, when CNN is used to identify photos of dogs, the same filters are employed in the same layer to detect the dog's nose, ears, and tail across the entire image [Kipf and Welling, 2016a].

Both, CNNs and GCNs, learn features by analyzing their data, the key difference is in data structure, GCN data is represented through graph nodes and neighboring nodes. CNNs are tailored to process data with a regular (Euclidean) order. At the same time, GCNs represent a generalized form of CNNs suitable for handling data with varying numbers of node connections and unordered nodes, designed explicitly for irregular or non-Euclidean structured data. Figure 2.3 illustrates convolution on CNNs and GCNs.

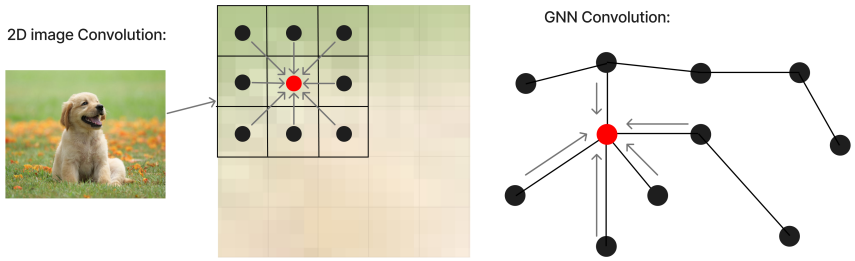


Figure 2.3: Traditional CNN and GNN convolutions.

The working of a graph convolutional neural network can be organized by the topics below:

1. Conversion to vectors: Associating of each node in the graph with a feature vector $\mathbf{x}$. These vectors can represent various attributes or characteristics depending on the GCN application, and together compose the input set $\mathbf{X} = \{x_1, x_2, ..., x_n\}$.

2. Convolution: The core of the GCN. It aims to aggregate information from neighboring nodes by performing a weighted sum of the feature vectors of neighboring nodes. The resulting aggregated information is a new feature vector for each node. The graph's adjacency matrix (similar to CNN's filter) provides weights for the aggregation process. Lastly, the aggregated features are passed through a RELU activation function to introduce non-linearity.

In a graph $G = (N, E)$, the node feature vectors can be represented by $\mathbf{H} = \{h_1, h_2, ..., h_n\}$. In the case of the first GCN layer, the inputs are the feature vectors $\mathbf{x}$ previously extracted on the first step ($\mathbf{H}^{(0)} = \mathbf{X} = \{x_1, x_2, ..., x_n\}$). Each layer l produces a new set of node features $\mathbf{H}^{(l)} = \{h_1^{(l)}, h_2^{(l)}, ..., h_n^{(l)}\}$ as its output. This step can be represented in the form of the **equation of GCN**:

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{2.6}$$

$\mathbf{H}^{(l+1)}$ represents layer l's output while $\mathbf{H}^{(l)}$ is its input. $\hat{\mathbf{A}}$ is the normalized adjacency matrix, $\mathbf{W}^{(l)}$ is the trainable weight matrix for the l-th layer (i.e, that will learn across train iterations), and $\sigma(\cdot)$ is an activation function such as ReLU.

The normalized adjacency matrix $\hat{\mathbf{A}}$ effectively refines a node's feature vector based on the feature vectors of its close graph neighbors. This matrix captures the graph structure. $\hat{\mathbf{A}}$ is normalized to make each neighboring node's contribution proportional to the network's connectivity. It is also important to highlight that we commonly have many convolution layers in sequence on a GCN architecture, meaning $\mathbf{H}^{(l+1)}$ can serve as another GNC layer input for further feature extraction.

3. Task-Specific Output: The last step is adapting the GCN output to solve the desired task. Its output presents valuable insights that can be used, with the help of a fully-connected layer or a softmax classifier, to solve many graph-based tasks such as node classification, graph classification, or link prediction.

GCN challenges

Although GCN has been successfully used for various semi-supervised classification problems, limitations and drawbacks still need to be addressed. In particular, traditional GCN approaches adopt a homogeneous treatment of nodes, considering documents and concepts as homogeneous entities and disregarding their inherent differences. Moreover, GCNs employ a simple aggregation scheme that fails to fully capitalize on the rich semantic connections between documents and concepts. In practice, these limitations can be summarized as a lack of attention mechanism, as GCNs do not incorporate mechanisms to weigh the importance of different nodes and edges dynamically. Consequently, the model may assign equal importance to all concepts, overlooking the potential relevance of specific concepts to documents and their respective classes.

2.3.2 Graph Attention Networks (GATs)

The limitations of GCNs in handling bipartite graphs and differentiating the importance of each concept within the graph have motivated exploring alternative models. In this context, Graph Attention Networks have gained attention. GATs, introduced by [Veličković et al., 2017], incorporate masked attention mechanisms that allow nodes to selectively attend to their neighbors during the information aggregation step (i.e., message passing) and improve graph modeling. This attention mechanism enables GATs to capture the importance of different nodes and their relationships, leading to improved performance and enhanced interpretability.

Attention

The Attention Mechanism, first introduced by Bahdanau et al. [Bahdanau et al., 2016] in the context of neural machine translation, revolutionized sequence-to-sequence modeling. Traditionally, neural machine translation systems relied on fixed-length vector representations to encode source sentences, posing limitations on their ability to capture long or complex relationships between words. Bahdanau et al. [Bahdanau et al., 2016] proposed an innovative extension by integrating an attention mechanism with Recurrent Neural Networks or convolutions, enabling models to dynamically weigh the importance of different parts of the source sentence during translation. This allowed the model to soft-search relevant information without the constraint of fixed-length vectors and enhanced performance.

Building upon this foundation, Vaswani et al. [Vaswani et al., 2017a] further refined the attention mechanism and introduced the transformer architecture, which relies solely on attention mechanisms without recurrent or convolutional layers. The transformer model demonstrated superior performance in various sequence transduction tasks, including machine translation. Unlike traditional models, the transformer achieved remarkable results while being more parallelizable and requiring significantly less training time. This exemplifies how the attention mechanism, as central to the transformer architecture, not only improves performance but also streamlines the training process, making it more efficient and scalable across different tasks.

The core process behind the attention mechanism is self-attention, also known as intra-attention, and can be defined as attention applied to a single context, or sequence (e.g., a sentence or a document), instead of across multiple contexts [Ramachandran et al., 2019]. It allows a model to weigh different input elements, usually tokens in a single sequence, based on their relevance to each other to compute a representation of the sequence [Vaswani et al., 2017a, Lin et al., 2017].

The self-attention workflow is as follows:

1. Given an input sequence, each token is represented as an embedding vector.
2. For each token, self-attention computes a weighted sum of all tokens in the sequence, considering their importance relative to each other.
3. The relevance (i.e., attention score) between two tokens is calculated using a similarity function.
4. The attention scores are normalized using a softmax function to ensure they sum up to 1.
5. The weighted sum of all tokens, using attention scores, produces the context vector for each token in the sequence.

Working of GAT

Graph Attention Networks leverage masked self-attention to address the shortcomings of prior methods that rely on graph convolutions. By using these attention mechanisms, nodes can attend to their neighborhoods' features, specifying different weights to different nodes in a neighborhood without requiring complex matrix operations or prior knowledge of the graph structure and making the model suitable for both inductive and transductive problems [Veličković et al., 2017].

The working of a graph convolutional neural network can be organized by the topics below:

1. Conversion to vectors: Initialization process, similar to GCNs, associating each graph node with a feature vector $\mathbf{x}$ and creating the input set $\mathbf{X} = \{x_1, x_2...x_n\}$.
2. Self-Attention Mechanism: GAT implements a self-attention mechanism on the nodes and computes attention coefficients, also named attention scores or context vectors, for each node, indicating the importance of one node to another. These attention coefficients are based on the features of the central node and its neighbors. The attention coefficients are calculated using a weighted sum of the features of the central node and its neighbors.

   Considering the same graph $G = (N, E)$, the node feature vectors can be represented by $\mathbf{H} = \{h_1, h_2, ..., h_n\} \in \mathbb{R}^F$, F being the number of features in each node of N. The layer l produces a new set of node features $\mathbf{H}^{(l)} = \{h_1^{(l)}, h_2^{(l)}, ..., h_n^{(l)}\} \in \mathbb{R}^{F^{(l)}}$ as its output. The attention coefficients update process is:

   (a) The attention coefficients are calculated with the application of the self-attention similarity function, the following equation, where $\mathbf{W} \in \mathbb{R}^{F' \times F}$ is the weight

matrix of each node $\{i, j, k, ..., n\}$ and $a : \mathbb{R}^{F'} \times \mathbb{R}^{F} \rightarrow \mathbb{R}$ represents the self-attention mechanism:

$$e_{ij} = a(\mathbf{W}\mathbf{H}_i^{(l)}, \mathbf{W}\mathbf{H}_j^{(l)}) \tag{2.7}$$

(b) To make coefficients easily comparable across different nodes, we normalize them across all choices of j using the softmax function

$$\alpha_{ij} = \text{softmax}_j(\mathbf{e}_{ij}), \tag{2.8}$$

3. Node feature vector update: After the calculation of the attention coefficients, we proceed to update each feature vector by employing a weighted sum of the features of the neighboring nodes:

$$\mathbf{h}_i^{(l+1)} = \sigma(\sum_{j \in N_{bhd}(h)} \alpha_{ij} \cdot \mathbf{W}^{(l)} \cdot \mathbf{h}_j^{(l)}) \tag{2.9}$$

$\mathbf{h}_i^{(l+1)}$ represents layer l's output while $\mathbf{h}_j^{(l)}$ is its input, which corresponds to all feature vectors on its one-hop neighborhood $j \in N_{bhd}(n)$. $\mathbf{W}^{(l)}$ is the trainable weight matrix for the l-th layer, α_{ij} are our normalized attention head coefficients, and $\sigma(\cdot)$ denotes the non-linearity ReLU function.

4. Task-Specific Output: Adapting GAT output to solve the desired graph-based tasks using techniques such as a fully-connected layer or a softmax classifier.

2.4 Coarsening

Graph coarsening, or graph reduction, is a dimensionality reduction technique for approaching large-scale graphs in machine learning problems [Kumar et al., 2022, Cai et al., 2021]. It aims to create a smaller tractable graph, often more computationally efficient, while still preserving the original graph's main properties and general structure [Kumar et al., 2022].

In general, the coarsening algorithm traditionally has two main steps to ensure the graph reduction occurs correctly:

- Node aggregation: Nodes of the original graph are grouped into clusters based on a certain criterion, reducing the overall number of nodes in the graph.
- Edge Contraction: The edges between nodes are rearranged, merged, or removed according to the new node structure.

Figure 2.4 represents the coarsening of a graph.

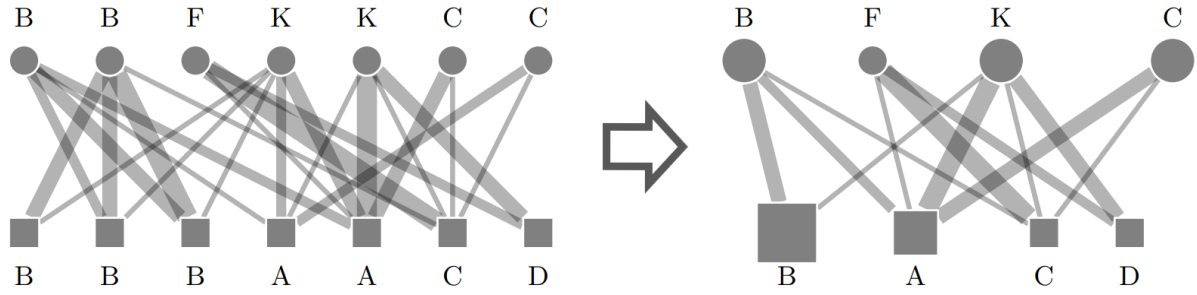


Figure 2.4: Visual representation of coarsening algorithms results. Image credits to [Eduardo Althoff et al., 2023].

In scientific computing, graph coarsening techniques can be traced back to the development of multigrid methods in numerical analysis, particularly in solving partial differential equations (PDEs) and optimization problems [Chen et al., 2021]. Graph coarsening emerged as a key component of multigrid methods as a way to accelerate the convergence of iterative solvers for large-scale linear systems arising from PDEs, especially on irregular networks, where traditional multigrid approaches could not be directly applied [Bakhvalov, 1966, Brandt, 1977].

Over time, graph coarsening techniques have been further developed and adapted to various applications in computer science, particularly in graph algorithms, machine learning, and network analysis. In semi-supervised learning, where graphs are built using labeled and unlabeled data, training a model, storage and computational costs can become prohibitive obstacles as the data grows. This can make using it impractical in specific applications [Walshaw, 2004]. Therefore, graph coarsening offers a valuable alternative to address these challenges [Liu et al., 2018].

Coarsening via semi-synchronous Label Propagation for K-partite networks (CLPk)

In our research, we will use the CLPk algorithm [Eduardo Althoff et al., 2023][1], an innovative coarsening method applicable to k-partite networks and capable of significantly maintaining classification performance. Most coarsening algorithms analyze networks with only one type of nodes and edges, known as uni-partite networks. However, real-world information networks are often heterogeneous, consisting of various types of node and edges, thus the distinctions between node types are overlooked and do not efficiently represent the graph's separate structure and treatment of its nodes.

Althoff et al. [Eduardo Althoff et al., 2023] propose the CLPk algorithm that can efficiently represent heterogeneous graphs with multiple types of nodes and edges, each with

[1]Code for the CLPk algorithm available on `https://github.com/pealthoff/CoarseKlass`.

its entities, attributes, or relationships. The proposed method uses a label propagation technique that orders partitions, the k-groups of nodes on the graph, and selects paths in the graph's schema to improve coarsening performance. Subsequently, the coarsening via semi-synchronous label propagation for bipartite networks (CLPb) algorithm, introduced by Valejo et al. [Valejo et al., 2021a], is applied to partition pairs performing the reduction process.

Considering a heterogeneous graph $G = (N, E)$, the algorithm first identifies the graph partitions, mutually exclusive groups (e.g, bipartite, tripartite, etc.) of the set of nodes N, and utilizes labeled nodes from the target N-partition N_t to guide the reduction process. The k-partite network is first decomposed into a series of bipartite networks, with pairs of partitions selected from the original network. The matching approach for each partition and thus the order in which CLPb will be applied follows a breadth-first search starting on the target partition N_t and following the shortest path between the partitions and N_t since shorter paths are more likely to indicate stronger relationships between nodes.

Next, the CLPb coarsening algorithm is adapted to the selected partition pairs, with the closer partition to N_t acting as the propagator partition and the other as the receptor partition. The coarsening process is executed semi-synchronously, grouping nodes with the same labels into super-nodes, and is applied to all non-target partitions following the breadth-first search tree.

In the case of bipartite graphs and not k-partite heterogeneous graphs, in our case study, the CLPb is applied to the entirety of the graph on both target and non-target partitions without the decomposition and breadth-first ordering.

Coarsening via semi-synchronous Label Propagation for bipartite networks (CLPb)

In this section, we explain the coarsening strategy using semi-synchronous label propagation for bipartite networks, as introduced in Valejo et al.'s research [Valejo et al., 2021a], initially designed for the unsupervised scenario. The CLPb employs the concept of cross-propagation to diffuse labels across layers. Upon convergence, nodes within the same layer sharing the same label merge into a single super-node.

A label is defined as a tuple $\mathcal{L}_n(l, \beta)$, where l represents the current label and $\beta \in [0, 1] \subset \mathbb{R}^+$ its associated score. At first, each node $n \in N$ is initialized with a starting label $\mathcal{L}_n = (n, 1.0/\sqrt{\kappa(n)})$, where $\mathcal{L}_n$ is identified by its "id" and has a maximum score of $\beta = 1.0$.

In each step, a new label is propagated to a receiving node n by selecting the label with the highest β from the collective labels of its k neighboring nodes, $k \in N_{bhd}(n)$. This propagation process operates according to the subsequent filtering rules:

1. Equal labels $\mathcal{L}^{eq} \subseteq \mathcal{L}_n$ are merged and the new β' is composed by the sum of its belonging scores:

$$\beta' = \sum_{(l,\beta)\in\mathcal{L}^{eq}} \beta, \tag{3}$$

2. The belonging scores of the remaining labels are normalized, where γ is the number of remaining labels.:

$$\mathcal{L}_n = \left\{ \left(l_1, \frac{\beta_1}{\beta^{\text{sum}}} \right), \left(l_2, \frac{\beta_2}{\beta^{\text{sum}}} \right), \ldots, \left(l_\gamma, \frac{\beta_\gamma}{\beta^{\text{sum}}} \right) \right\} \tag{4}$$

$$\beta^{\text{sum}} = \sum_{i=1}^{\gamma} \beta_i \tag{5}$$

3. The label with the largest β is selected:

$$\mathcal{L}'_n = \arg\max_{(l,\beta)\in\mathcal{L}_n} \mathcal{L}_n. \tag{6}$$

4. The size of the coarsest network is controlled by the user, thus the minimum number of labels for each layer is a user-defined parameter η. This means a node $n \in N_i$, where i represents the graph partitioning layer, is only allowed to update its label if, and only if, the number of labels in the layer $\|L_i\|$ remains equal to or greater than η_i:

$$\|L_i\| \leq \eta_i \tag{7}$$

5. Lastly, a classical issue in the multilevel context is that super-nodes tend to be highly unbalanced at each level [Valejo et al., 2020]. Therefore, it is common to constrain the size of the super-nodes using an upper-bound $\mu \in [0, 1] \subset \mathbb{R}^+$, which limits the maximum size of a group of labels in each layer:

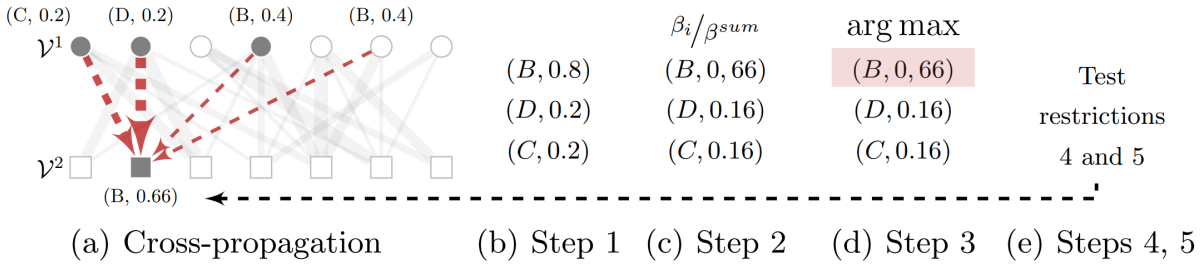
$$\mathcal{S}_i = \frac{1.0 + (\mu * (\eta_i - 1)) * \|V_i\|}{\eta_i} \tag{8}$$

where $\mu = 1.0$ and $\mu = 0$ imply highly imbalanced and balanced groups of nodes, respectively. Therefore, a node n with weight $\sigma(n)$ can update its current label l to a new label l' if, and only if:

$$\sigma(n) + \sigma\left(l'\right) \leq S_i \quad \text{and} \quad \sigma\left(l'\right) = \sum_{k \in l'} \sigma(k) \tag{9}$$

If restrictions 4 or 5 are not attained, the algorithm returns to step 3; the label with the maximum β is removed, and a new ordered label is selected. The process is repeated until a label that satisfies the restrictions 4 and 5 is obtained. Figure 2.5 represents the complete application process of the abovementioned rules.

After the cross-propagation convergence, the algorithm collapses each group of matched nodes (i.e., nodes with the same label) into a single super-node. Links that are incident to matched nodes are collapsed into the so-called super-links. It is important to note that the CLPb process does not guarantee that the desired minimum number of labels η will be reached at the current level, and thus, the algorithm can stop with a greater number of labels than the desired one.



(a) Cross-propagation (b) Step 1 (c) Step 2 (d) Step 3 (e) Steps 4, 5

Figure 2.5: Visual example of the five CLPb steps. **a** shows that the cross-propagation process is being performed from the top layer N^1 (propagator nodes), to the bottom layer N^2 (receiver nodes). **b** shows that equal propagator N^1 labels are merged. **c** shows the labels' normalization step. **d** select B as it present the larger β value. Lastly **e** tests restrictions 4 and 5, if they are satisfied the selected label is propagated to the receiver layer, if not the algorithm returns to step **d**. This algorithm can be repeated t times until convergence. Upon cross-propagation convergence, receiver N^2 nodes with the same labels are aggregated into super-nodes. Image credits to [Eduardo Althoff et al., 2023].

2.5 Large Language Models (LLM)

Large Language Models are artificial intelligence models that exhibit remarkable capabilities in natural language tasks and beyond, such as answering questions, translating languages, and generating complex texts with human-like proficiency [Naveed et al., 2023]. As by their name, these models are trained on a large amount of text data like books, articles, and web pages, adjusting the immense quantity of parameters and weights composing their architecture and learning the underlying patterns and structures of human language.

LLMs are becoming highly sought-after worldwide due to their capacity as general-purpose task solvers and, thanks to fine-tuning, even highly accurate solvers for specific tasks [Fan et al., 2023]. These models also present emergent abilities such as improved zero-shot learning and few-shot learning [Rahman et al., 2018]. This means they are capable of solving the task at hand with none or few contextualization prompts, showing adaptation and reasoning capacity to learn thought inputs and outputs actively. Prompts are a core concept of an LLM and can be defined as a piece of text used to guide the production of a Large Language Model [Cain, 2023]. It can be understood as a starting point that provides context and constraints for the model to generate text relevant to the desired tasks with the desired format and level of complexity.

Training a billion-scale model is difficult as compared to a smaller model. LLMs are prone to various instabilities during training, such as hardware failure [Zhao et al., 2023]. In general, the training of an LLM can be divided into two main steps: pre-training and fine-tuning. Pre-training refers to the phase where the LLM is trained on a large and diverse corpus of text data, mainly using unsupervised learning techniques to enable the LLM to learn general language skills, such as grammar, syntax, semantics, and common-sense knowledge. Fine-tuning is the phase where the LLM is re-trained on a smaller and more specific dataset, usually related to a particular task, using supervised or semi-supervised learning techniques. The complexity of designing and training such models leads to various strategies and architectures tailored for specific NLP tasks or based on remarkable academic findings.

A famous example of LLMs is GPT-4 [OpenAI et al., 2024], a descendant of the GPT and chat-GPT family, with the architecture composed of unidirectional transformers that focuses on autoregressive text generation, making it adept at tasks like creative writing, dialogue systems, and summarization. BERT [Devlin et al., 2019], in contrast, employs a bidirectional transformer architecture, allowing it to process text in both directions simultaneously. This bidirectionality makes it highly effective for question answering, sentiment analysis, and named entity recognition tasks. T5 (Text-to-Text Transfer Transformer) [Raffel et al., 2023] adopts an encoder-decoder architecture, framing all NLP problems as text-to-text tasks, enabling seamless multitask learning for applications like translation, summarization, and text classification. These models exemplify LLMs' various architectures, training strategies, and practical applications in numerous tasks.

Considering the variety of LLM models, we propose using Llama 3 8B [Dubey et al., 2024], with 8 billion parameters, as our low-cost labeler, replacing the high cost of human effort. In this strategy, the costs would lay on the model itself, considering the model needs payment for its use in the development, prompt engineering, labeling process, and computational costs. As Llama 3 is open-source, the monetary costs for its use are zero,

with only development and computational expenses remaining.

Llama 3

Llama 3 (i.e., Large Language Model Meta AI 3) [Touvron et al., 2023a] is a LLM foundation model developed by Meta AI, designed as a general-purpose language model that supports many AI tasks. The model extends the Llama series' and focuses on efficiency and accessibility, aiming to achieve the best possible performance at various inference budgets, showing advanced capabilities while using publicly available datasets. The resulting models range from 8B, 70B to 405B parameters with competitive performance compared to the best existing LLMs.

The architecture of Llama 3 builds on the transformer design [Vaswani et al., 2017a], integrating several innovations to enhance efficiency and performance. Similarly to Llama 1 [Touvron et al., 2023a], it employs pre-normalization using RMSNorm [Zhang and Sennrich, 2019] for normalizing inputs rather than outputs within transformer layers, also adopts the SwiGLU activation function [Shazeer, 2020], which improves non-linear transformations and enhances model expressiveness, and incorporates Rotary Positional Embeddings (RoPE) [Su et al., 2023], which replace traditional positional encodings, allowing the model to manage long input sequences better.

From Llama 2 [Touvron et al., 2023b] it mainly inherits grouped-query attention GQA [Ainslie et al., 2023] that improves inference scalability for larger models, by grouping tokens together into a shared query vector which attend all key-value heads, maintaining model's ability to capture long-range components while reducing the number of query computations.

Llama 3 main improvements from other Llama models are in data quality, diversity, and increased training scale. Key architectural updates include adopting GQA with eight key-value heads for faster inference and reduced memory requirements, an updated attention mask to handle long sequences effectively, and a larger 128K-token vocabulary that improves tokenization for both English and non-English texts. The RoPE base frequency parameter is also increased to support longer context lengths up to 32,768 tokens.

The training process of Llama 3 has two main stages: language model pre-training and language model post-training [Vaswani et al., 2017a]. During the pre-training phase, the model was trained on a vast corpus of 15.6 trillion tokens, covering multilingual, general knowledge, reasoning, and coding domains. This phase focuses on next-token prediction, enabling the model to acquire a foundational understanding of language and extensive world knowledge. A significant aspect of this stage was data curation, employing rigorous

deduplication, quality filtering, and domain-specific enhancements, such as including high-quality coding and reasoning datasets.

Also, scaling laws and annealing (i.e., slowly decreasing learning rates) strategies were crucial in optimizing model size and training efficiency while improving downstream performance. Overall, the pre-training recipe consists of the initial pre-training using AdamW with a 0.00008 learning rate and context windows of 8K tokens, next step uses long context pre-training to gradually expand the context window to 128K tokens for increased model context range, and lastly, a linear annealing step is applied to decrease the learning rate to 0 to refine model performance on key data.

The post-training phase refined the pre-trained Llama 3 model to better align with human expectations and enhance specific capabilities, such as instruction following and multilingual understanding. This was achieved through supervised fine-tuning (SFT), rejection sampling, and Direct Preference Optimization (DPO). The process involved leveraging human-annotated preference data and synthetic examples, focusing on fine-tuning the model's performance on tasks requiring precise instruction-following, tool use, and coding. Additionally, safety measures were integrated during this phase to mitigate potential harm, ensuring a balance between helpfulness and harmlessness.

Llama 3 demonstrated competitive performance across a broad range of benchmarks [Vaswani et al., 2017a]. It was evaluated on tasks encompassing general knowledge (e.g., MMLU), reasoning (e.g., ARC Challenge), coding (e.g., HumanEval), mathematical problem-solving (e.g., GSM8K), tool use (e.g., BFCL), Long context (e.g., Zero-SCROLLS), and multilingual(e.g., MGSM) achieving state-of-the-art results on several benchmarks within its size classes and outperforming alternative models with similar numbers of parameters. Overall, Llama 3 delivers notable safety and alignment metrics improvements compared to its predecessors, showing versatility and potential as a foundational AI model across diverse applications.

Chapter 3

Related Work

3.1 Concept/Keyphrase Extraction

In the context of generating bipartite graphs using documents and concepts, it is important to highlight the works on keyword and keyphrase extraction. Surveys like [Beliga, 2014], [Siddiqi and Sharan, 2015], and [Papagiannopoulou and Tsoumakas, 2019] present us with an excellent overview of how the field has significantly evolved over the past few decades, and its modern perspectives. Mainly, Papagiannopoulou et al. [Papagiannopoulou and Tsoumakas, 2019] describes a systematization of state-of-the-art methods focused on keyphrase extraction, classifying them based on their characteristics and similarities, as shown in subsection 2.2, and addresses the main works on these methods.

Nomoto et al. [Nomoto, 2022] does a similar job but focuses mainly on the methods themselves and their history. According to Nomoto, early techniques, such as the term frequency-inverse document frequency (TF-IDF), emerged in the 1970s to determine the importance of terms within a document by balancing their frequency against their rarity across a corpus [Salton and Yang, 1973]. In 1990, most attempts to tackle the issue were primarily based on text statistics, and the field of information retrieval was led by DARPA's Topic Detection and Tracking (TDT) [Allan, 2002]. While effective, these statistical methods primarily relied on shallow text representations, limiting their ability to capture semantic and contextual nuances.

In the 2000s, advances in graph-based methods like TextRank [Mihalcea and Tarau, 2004], inspired by Google's famous PageRank algorithm, began leveraging the structural relationships between words and phrases. Such methods treated documents as networks and graphs, where the importance of terms was determined based on their connectivity within these networks. Around the same period, topic modeling approaches, particularly Latent Dirichlet Allocation (LDA) [Blei et al., 2003a], introduced probabilistic models

to identify topics and their associated terms, adding depth to keyword extraction by revealing latent structures in the text.

State-of-the-art techniques now predominantly rely on deep learning paradigms. Generative models, such as sequence-to-sequence architectures with attention mechanisms, have enabled systems to generate keywords that may not explicitly appear in the source text, addressing limitations of previous extractive methods [Meng et al., 2017]. Moreover, hybrid approaches combining supervised classification frameworks with neural embeddings further enhance the precision of keyword detection [Zhang et al., 2016]. These modern methods improve extraction accuracy and enable applications across diverse domains, making them robust tools in contemporary natural language processing.

Sharma et al. [Sharma and Li, 2019] methodology based on BERT and BILSTM keyphrase classification, and Smires et al. [Bennani-Smires et al., 2018] MMR algorithm are the main referential points on the developing of the minimal and easy-to-use keyword extraction framework KeyBERT, all previously discussed in subsection 2.2. Grootendorst et al.'s KeyBERT [Grootendorst, 2020] can accurately capture a document's main concepts by leveraging BERT embeddings and cosine similarity in an N-gram strategy. In our work, in particular, we investigated and employed the use of KeyBERT for concept extraction to support the construction of our bipartite graphs.

Recent studies for text classification show improvement in classification performance when incorporating information from concepts [Gôlo and Marcacini, 2023] or keywords [de Souza et al., 2024] during learning, which shows that representation through concepts and keywords enhances semi-supervised classification [Neogi et al., 2020], especially for data modeled using graphs [de Souza et al., 2024]. Finally, [Xie et al., 2021] improved classification performance when exploring topic modeling in bipartite graphs for text classification, showing the importance of exploring graph neural networks in bipartite graphs.

3.2 Graph Neural networks

The primary techniques employed in this article revolve around two fundamental models: Graph Convolutional Network and Graph Attention Network. The survey [Khemani et al., 2024] by Khemani et al. does an incredible study about GNNs in general, addressing in depth the functioning of GCN and GAT. The work of [Gori et al., 2005] pioneered the concept of GNN, drawing inspiration from the recurrent neural network architecture. Subsequently, the work of [Veličković et al., 2017] introduced GAT, which utilized masked attention to enhance graph modeling. Building upon this, [Ding et al., 2018] further improved GAT's classification performance by utilizing Generative Adversarial Nets (GANs).

Below, we provide a literature review highlighting the increasing interest in graphs and deep learning, the evolution of GCNs to GATs, and the use of concepts and key-topic information on GNNs.

Most corpus-level graphs typically consist of word and document nodes with word-document edges. However, certain studies have stood out by incorporating word-word edges. In the TextGCN approach [Yao et al., 2018], the authors construct a corpus-level graph comprising training document nodes, test document nodes, and word nodes. Before graph construction, a standard preprocessing method involves removing words that appear fewer than five times or are listed in NLTK [Bird et al., 2009] stopwords. The edge value between document and word nodes is determined using TF-IDF, while the edge value between word nodes is based on PMI.

To enhance the efficiency of Graph Convolutional Networks, the SGC (Simplified Graph Convolutional Network) method [Wu et al., 2019] removes the nonlinear activation function in GCN layers. Additionally, the S^2GC approach [Zhu and Koniusz, 2021] addresses over-smoothing issues in GCNs by incorporating self-loops using the Markov Diffusion Kernel. In contrast to S2GC, the NMGC method [Sun et al., 2022] utilizes the sum of each GCN layer. Furthermore, NMGC (Multihop Neighbor Information Fusion Graph Convolutional Network) applies the Multi-hop Neighbour Information Fusion (MIF) operator, employing min pooling to mitigate over-smoothing problems.

Considering GCN's limitations in handling heterogeneous graphs and node importance differentiation, in the TG-Transformer approach [Zhang and Zhang, 2020], TextGCN is modified to introduce heterogeneity into the graph by treating document and word nodes as distinct types of nodes during propagation. To handle large corpus graphs, subgraphs are sampled from the TextGCN graph using the PageRank algorithm [Page et al., 1998]. The input embedding is computed as the sum of three types of embeddings: pre-trained GloVe embedding, node-type embedding, and Weisfeiler-Lehman structural encoding [Niepert et al., 2016]. Self-attention [Vaswani et al., 2017b] with graph residual [Zhang and Meng, 2019] is applied during propagation.

Incorporating topic/concept information from each document in corpus-level graph neural networks can provide additional insights. Several models extend the graph by including topic nodes. In the HGAT (Heterogeneous Graph Attention Network) approach [Linmei et al., 2019], HGAT utilizes LDA (Latent Dirichlet Allocation) [Blei et al., 2003b] to extract topic information for each document. The top K topics with the highest probabilities are selected and connected to the document. Instead of directly using words, HGAT leverages external knowledge by employing the entity linking tool TAGME [Piccinno and Ferragina, 2014] to identify entities within the document and establish connections. The connectedness between entity nodes is defined based on semantic similarity

using pre-trained Word2Vec with a threshold. As the graph is heterogeneous, a HIN (heterogeneous information network) model is implemented to propagate solely on each sub-graph based on the node type. HGAT incorporates a dual attention mechanism, where type-level attention learns the weights of different types of neighboring nodes. In contrast, node-level attention captures the importance of neighboring nodes while ignoring the type.

Other works like [Brody et al., 2021] look to further enhance the GAT attention mechanism. Brody et al. state that traditional GAT computes a limited kind of attention, named static attention, that calculates attention scores unconditioned on the query nodes. GATv2 is proposed to remove this limitation via a simple fix by modifying the order of the operations and introducing a strictly more expressive dynamic attention. GATv2 indeed outperforms GAT across 11 OGB and other benchmarks, but its use is mainly recommended upon complex and larger datasets, in the case of simpler and "easy-to-overfit" data traditional GAT is sufficient and can consistently outperform GATv2.

3.3 Coarsening

Using coarsening for computational cost reduction is a promising but complex idea, as most existing methods primarily reduce uni-partite networks with a single type of vertex and edge. Our main inspiration for coarsening and the technique chosen for our implementations, previously mentioned in subsection 2.4, is Eduardo et al.'s [Eduardo Althoff et al., 2023] CLPk algorithm. CLPk introduces a new coarsening method applicable to k-partite networks that can maintain classification performance while solving large networks' storage and processing problems. Results indicate that the proposed coarsening algorithm significantly improved storage efficiency and classification runtime, leading to over one-third savings in storage and twice as fast classifications with metrics exhibiting low variation. Additionally, [Valejo et al., 2021a] introduces the CLPb algorithm, further detailed in subsection 2.4, which is heavily referenced and serves as the base for the CLPk algorithm.

Considering the use of coarsening algorithms focused mainly on bipartite graphs, [Valejo et al., 2021b] presents a comparative analysis of many state-of-the-art coarsening algorithms. It presents a formal and illustrative description of such algorithms, illustrates their usage on a set of emblematic problems, and evaluates their accuracy using quality and runtime measures, highlighting strengths and shortcomings. Additionally in the realm of bipartite graphs, [Valejo et al., 2021a] introduces the CLPb algorithm, further detailed in subsection 2.4, which is heavily referenced and serves as the base for the CLPk algorithm previously mentioned.

Another more specific example of coarsening applied on bipartite graphs is [dos Santos et al., 2024], which, similarly to our study, is also based on Eduardo et al. [Eduardo Althoff et al., 2023] CLPk algorithm. Here, CLPk is used as a hierarchical coarsening technique to reduce the graph, and then a target algorithm is applied to the coarsened graph projecting the output back to the original graph. The graph is gradually coarsened and imputed to a GNN text classification model, the GraphSAGE algorithm, having its multilevel optimization evaluated on performance, training time, and memory consumption. It is emphasized that the experiments are conducted on text classification, but the proposed method is not bound to a specific task and, thus, can be generalized to different problems. Results show that on graphs with a 25% coarsening reduction, memory consumption reduced considerably achieving 1/7 of its original values (i.e., 85% reduction), training time reduced by 40% at average, but the F1-score and accuracy performance metrics also reduced by approximately 40%.

Lastly, using a different coarsening algorithm, neither CLPk nor CLPb, [Huang et al., 2021] introduces a generic and computationally efficient framework that employs graph coarsening to reduce the size of the input graph. This method enables sublinear memory and time costs during GNN's training while maintaining competitive performance. The paper also demonstrates that coarsening supports scalability and acts as a form of regularization, potentially improving model generalization. Results from empirical evaluations of real-world datasets show that coarsening can reduce the number of nodes by up to tenfold with minimal impact on classification accuracy, highlighting graph coarsening as a simple yet powerful tool for scaling GNNs.

3.4 Large Language Models

In the context of LLMs as data labelers, [Lee et al., 2023] stands out as an overview of data creation using Large Language Models. It presents a formal framework for data creation using LLMs. It proposes a single-shot formatting example-based data creation pipeline that leverages an instruction following LLM and applies to a broad range of tasks. Experimentation shows that instruction-following LLMs are highly cost-effective data creators and that models trained with these data perform better than those trained with human-labeled data by up to 17.5%.

In contrast, [Liu, 2023] presents concerns about the widespread use of LLM as data labelers, supporting the ongoing relevance of human-labeled data in the post-LLM era. It discusses the standards for using machines in labeling tasks, comparing the well-established safety protocols for human-labeled data to the LLM-generated data, considering potential risks and bias towards high trust in machine outputs, and emphasizing

the need for transparent auditing processes to ensure accountability. The challenges of aligning LLMs to generate helpful, harmless, and truthful outputs are acknowledged, and alternatives that focus on reinforcement learning from human feedback to fine-tune pre-trained LLM models are suggested.

Considering the integration of LLMs and GNNs [Ren et al., 2024] is a survey that explores the use of LLMs to enhance graph learning capabilities. It comprehensively reviews the latest state-of-the-art LLMs applied in graph learning. It introduces a novel taxonomy to categorize existing integration methods based on their framework design, highlighting the strengths and limitations of each approach, as well as architectural and pipeline innovations. The taxonomy delineates four primary architectural designs: GNNs as Prefix, where graph neural networks (GNNs) process graph data into structure-aware tokens for subsequent inference by LLMs; LLMs as Prefix, where LLMs handle graph-related textual data to produce embeddings or labels that enhance GNN training; LLMs-Graphs Integration, which focuses on more profound fusion techniques like joint training or creating LLM-driven agents for graph interaction; and LLMs-Only, which reformulates graph data into sequences for direct inference by LLMs, often incorporating multi-modal features.

Also, on applying LLMs to Graph Neural Networks, [Chen et al., 2023b] proposes an LLM-GNN approach that combines the strengths of both GNNs and LLMs while mitigating their limitations. Specifically, LLMs are leveraged to annotate a small portion of nodes, and GNNs are trained on LLMs' annotations to make predictions for the remaining large portion of nodes. Also, it presents a unique challenge in optimally selecting nodes for LLMs-annotation, ensuring high quality, representativeness, and diversity, and enabling the enhancement of GNN training. Results show that although LLMs are noisy labels, LLM-GNN can achieve high accuracy on many benchmark datasets and outperform other label-free classification methods.

Regarding the use of GATs combined with LLM models [Chen et al., 2023a] introduces GATGPT. This framework merges a graph attention mechanism with LLMs, focusing specifically on spatiotemporal imputation and applying several time series tasks like forecasting, classification, and anomaly detection. Unlike conventional methods that rely on specialized but computationally intensive architectures, GATGPT leverages the pre-trained knowledge of LLMs to model temporal dependencies. At the same time, the GAT module captures spatial relationships within the data. This hybrid approach enables efficient fine-tuning of upper layers for specific tasks while preserving the generalization power of the LLM pre-trained model. Results show that GATGPT achieves performance comparable to state-of-the-art benchmarks, highlighting its potential as a versatile and scalable solution.

Chapter 4

Research Methodology

Our research methodology was designed to explore and unite all the technologies explored in our background section 2, especially the document-concept bipartite graph architecture using keyphrases, Graph Neural Networks focusing on GAT and GCN implementations, graph coarsening, and Large Language Models using Llama 3 as labeler. We also delve into the datasets, and evaluation metrics utilized for our experiments.

Considering such factors, below we present a brief description of the methodologies employed on each topic previously mentioned, all of which will be further detailed across this section. Additionally, Figure 4.1 displays the general methodology flux employed in our works.

- Datasets: Present and examine the 12 textual document collections from different domains used as datasets for our experiments.
- Document-Concept Bipartite Graph Creation: Focuses on keyphrase extraction using KeyBERT and the bipartite graph generation and arrangement such as using K-nearest neighbors to ensure graph structure and connectivity.
- Document-Concept Graph Neural Networks: Focuses on the application of GNNs introduced in the background subsection 2.3, especially considering their functioning in the proposal of document-concept graphs.
  - Document-Concept Graph Convolution Networks: Define a GCN architecture adapted to fit into the document-concept bipartite graph methodology.
  - Document-Concept Graph Attention Networks: Define a GAT architecture adapted to fit into the document-concept bipartite graph methodology.
- Training GAT and GCN Models: Delves into how the GAT and GCN models developed will be trained and applied to produce the expected results for our evaluation.

Coarsening and LLM applications will also generate graphs that will be fitted to GAT and GCN training process.

- Coarsening: Applying the CLPk algorithm on our bipartite graphs, generating coarsened graphs that will be supplied for GATs and GCNs training and evaluation.
- Large Language Models Labeling: Using Llama 3 as a low-cost labeler, investigation zero-shot and few-shot prompts to ultimately label diverse volumes of data (i.e, 10, 50, and 100 documents per class), generating graphs with LLM-labeled data that also will be supplied for GATs and GCNs training and evaluation.
- Evaluation: The overall approach to assessing the results observed in each trained model, focusing mainly on the f1-Score metric.

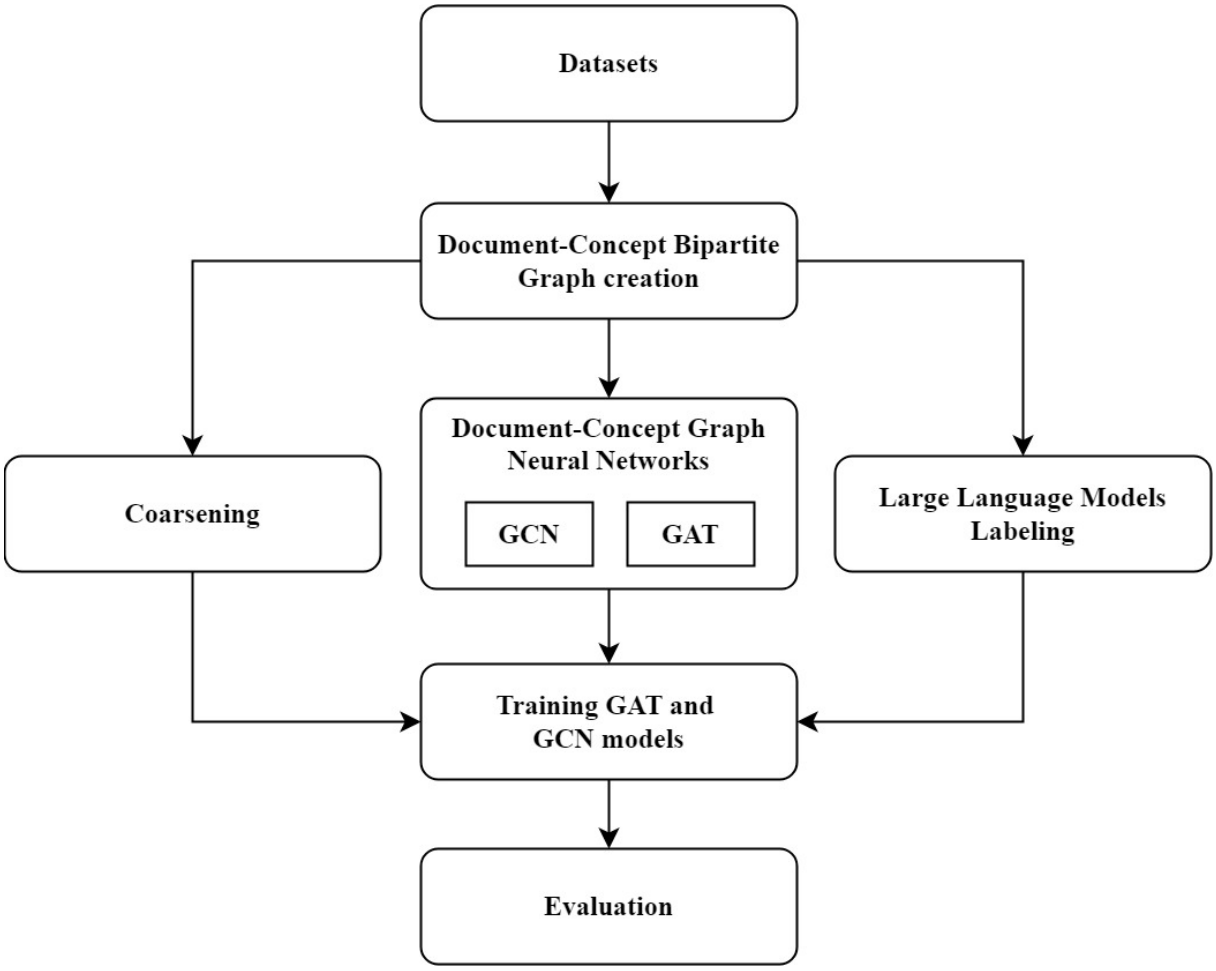


Figure 4.1: Methodology Flowchart.

4.1 Datasets

We used 12 textual document collections from different domains such as Sentiment Analysis (SA), Scientific Documents (SD), News Articles (NA), and Web Pages (WP) [Rossi et al., 2013]. Table 4.1 presents the text collections and the characteristics of these collections: the number of documents ($\|\mathcal{D}\|$), the number of terms ($\|\mathcal{T}\|$), the average number of terms per document ($\|\overline{\mathcal{T}}\|$), the number of classes ($\|\mathbb{C}\|$), the standard deviation considering the class percentages in each collection ($\sigma(\mathbb{C})$), and the percentage of the majority class ($\max(\mathbb{C})$). More details about the collections are presented in [Rossi et al., 2013], which delves further into the datasets' characteristics and benchmarks them for various classification models.

All twelve collections contain English texts with various lengths and class distributions. All collections are also explicitly adapted for the classification task. Each text can represent multiple things, such as an article title, text summary, web pages, reviews, and commentaries. For all collections, all of its composing texts make part of a determined class that better represents its contents considering the whole.

Simply put, our classification process consists of identifying the document's class based on the document's text.

Table 4.1: Characteristics of the textual document collections.

Collection	$\|\mathcal{D}\|$	$\|\mathcal{T}\|$	$\|\overline{\mathcal{T}}\|$	$\|\mathbb{C}\|$	$\sigma(\mathbb{C})$	$\max(\mathbb{C})$
Classic4 (SD)	7095	7749	35.28	4	1.94	45.16
CSTR (SD)	299	1726	54.27	4	18.89	42.81
Dmoz-Computers (WP)	9500	5011	10.83	19	0.00	5.26
Dmoz-Health (WP)	6500	4217	12.40	13	0.00	7.69
Dmoz-Science (WP)	6000	4821	11.52	12	0.00	9.63
Dmoz-Sports (WP)	13500	5682	11.87	27	0.00	3.70
Industry-Sector (PW)	8817	21490	88.49	12	7.37	11.24
NSF (CD)	10524	3888	6.65	16	3.82	13.39
Re8 (NA)	7674	8901	35.31	8	18.24	51.12
Review Polarity (SA)	2000	15698	205.06	2	0.00	50.00
SyskillWebert (WP)	334	4340	93.16	4	10.75	41.02
WebKB (WP)	8282	22892	89.78	7	15.19	45.45

Below is a simple description of each collection:

- Classic4: The Classic4 collection is composed of 4 distinct collections: CACM (titles and abstracts from the journal Communications of the ACM), CISI (information retrieval papers), CRANFIELD (aeronautical system papers), and MEDLINE (medical journals).
- CSTR: The CSTR (Computer Science Technical Reports) collection is composed of abstracts and technical reports published in the Department of Computer Science

at the University of Rochester from 1991 to 2007. The documents belong to 4 areas: Natural Language Processing, Robotics/Vision, Systems, and Theory.

- Dmoz-Computers: Dmoz-Computers is composed of web pages of the computers category extracted from DMOZ - Open Directory Project. The document classes are the subcategories of the computers category.
- Dmoz-Health: Dmoz-Health is composed of web pages of the health category extracted from DMOZ - Open Directory Project. The document classes are the subcategories of the health category
- Dmoz-Science: Dmoz-Science is composed of web pages of the science category extracted from DMOZ - Open Directory Project. The document classes are the subcategories of the science category
- Dmoz-Sports: Dmoz-Sports is composed of web pages of the sports category extracted from DMOZ - Open Directory Project. The document classes are the subcategories of the sports category.
- Industry-Sector: Industry-Sector collection is composed of web pages of companies from various economic sectors.
- NSF: NSF (National Science Foundation) collection is composed of abstracts of grants awarded by the National Science Foundation8 between 1999 and August 2003.
- Re8: Re8 collection is composed of articles from Reuters-21578 collection.
- Review Polarity: The review-polarity collection is composed of 1000 positive reviews and 1000 negative reviews about movies.
- SyskillWebert: The SyskillWebert collection is composed of web pages about bands, sheep, goats, and biomedicals.
- WebKB: WebKB collection is composed of web pages collected from computer science departments of various universities in January 1997 by the World Wide Knowledge Base15 (WebKb) project of the CMU Text Learning Group.

4.2 Document-Concept Bipartite Graph Creation

To create our bipartite graph structure, our proposed method explores using concepts to represent relevant parts of a document. We argue that this representation offers advantages in both classification efficiency and interpretability. From an interpretability

perspective, concepts provide a higher-level representation of the document content, allowing us to understand the key ideas and topics. Additionally, using concepts can enhance classification efficiency by reducing the dimensionality of the feature space. Instead of considering every word in the document, we focus on a compact set of concepts more informative and representative of the classes.

We start our approach utilizing KeyBERT [Grootendorst, 2020], which can extract a range of concepts from documents considering their contextual information and word relationships, as detailed in subsection 2.2. The extraction of concepts involves using BERT to calculate sentence embeddings from the documents' texts. A set of keyphrases is then extracted using a sliding window approach (N-gram). Still, only those with high cosine similarity to the sentences are selected as concepts, since they better describe the document. KeyBERT enables us to map keyphrases by specifying the number of words that determine the size of the keyphrase. We opted for the keyphrases size: 2, 3, and 2 or 3 words to evaluate how varying sizes would affect the GNNs models' performance. Also, We request KeyBERT to select at least 3 and at maximum 5 keyphrases for each document.

Next, each of the extracted concepts and their source documents are converted into embeddings. We use BERT to numerically represent their textual information, transforming them into the numerical format, ideal for input GNN models. These embeddings are what fundamentally compose our graph, being our feature vector, and each embedding represents one document node or concept node, depending on its origin.

In possession of the extracted concepts (i.e., keyphrases), and their wrapping in embedding format, we begin the construction of the bipartite graphs. Similarly to the definition in subsection 2.2.1, we create the bipartite graph, $G = (N, E)$ where $N = D \cup C$ represents the set of nodes and E represents the set of edges. We denote the set of documents as D and the set of concepts as C. A naturally created edge $(d, c) \in E$ indicates that the concept c was extracted from the document d.

Besides natural edges, our method utilizes an incremental concept extraction approach to introduce artificial edges until there is a path in the graph connecting any pair of documents, ensuring no disconnected components exist. This incremental strategy is simple and intuitive, creating edges between documents and concepts based on their K-nearest neighbors (KNN) [Cunningham and Delany, 2021].

for the creation of artificial edges, we first tokenize documents into sentences to increase the possible similarities between concepts and documents, meaning each specific sentence or part of a document can be similar to other concepts. Next, for each tokenized document sentence d_s, an edge is added to connect it with its k most similar neighbor concepts c until there are no more disconnected components in the graph. We selected $k = 15$ for all

datasets since it promoted full connectivity of our graph without causing over-connection and increasing graph complexity. Lastly, the case of duplicated edges originated by KNN incrementation is unified.

4.3 Document-Concept Graph Neural Networks

With the graph architecture defined, we return to our main goal, which is to address the problem of semi-supervised text classification, where only a small portion of documents per class are initially labeled. We propose using Graph Attention Networks for bipartite document-concept networks to achieve this. GATs leverage the graph's topology and labeled documents to automatically learn the importance of concepts for document classification. Also, we implement Graph Convolutional Networks as a comparative baseline model. Due to GCN's limitations, we must adopt a homogeneous treatment of document and concept nodes for this approach.

We further elaborate on GAT and GCN graph neural network models' overall definitions, functioning, and specific characteristics in the background subsection 2.3. In this sub-section, we focus specifically on GAT and GCN applications in the context of document-concept bipartite graphs, concentrating on the implementation performed as well as the necessary adaptations.

Consider this time, for both GNNs below (GCN and GAT), our bipartite graph, mentioned earlier in subsection 4.2, represented as $G = (D \cup C, E)$, where D is the set of document nodes and C is the set of concept nodes. The edges E connect documents to concepts. Each node in the graph is associated with a feature vector. Let $\mathbf{X} \in \mathbb{R}^{\|D \cup C\| \times F}$ be the feature matrix, where F is the number of input features per node. The feature matrix $\mathbf{X}$ is the input to the GNNs, our documented and concept embeddings.

4.3.1 Document-Concept Graph Convolution Networks

Graph Convolutional Networks are among the pioneering strategies in semi-supervised classification and are still considered state-of-the-art [Wu et al., 2023]. The goal of the GCN is to learn node representations that capture the graph structure and semantic information. The GCN layer computes new representations for each node by aggregating information from its neighbors [Kipf and Welling, 2016b]. Yet, GCN aggregation adopts a homogeneous treatment of nodes and thus fails to interpret the differences and importance between documents and concept nodes.

Let $\mathbf{H}^{(l)} \in \mathbb{R}^{\|\mathcal{D} \cup \mathcal{C}\| \times D_{ims}}$ be the node representations at the l-th layer of the GCN, where D_{ims} is the number of dimensions in the representation. The initial node representations $\mathbf{H}^{(0)}$ are set to the input features vector $\mathbf{X}$, our embeddings. The propagation rule in the

GCN, which composes the graph convolutional layers, also known as **equation of GCN** shown in subsection 2.3.1, can be defined as:

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

where $\hat{\mathbf{A}}$ is the normalized adjacency matrix, $\mathbf{W}^{(l)}$ is the trainable weight matrix for the l-th layer, and $\sigma(\cdot)$ is an activation function such as ReLU.

This rule is responsible for the node aggregation processes and for updating node feature vectors through iterations: $\mathbf{H}^{(l)} \rightarrow \mathbf{H}^{(l+1)}$. We emphasize that this equation is unable to grasp node importance, meaning every node will equally impact $\mathbf{H}^{(l+1)}$ in a homogeneous treatment.

After the graph convolutional layers, responsible for applying the **equation of GCN** and updating nodes feature vectors, we add a traditional softmax classifier to perform semi-supervised classification and effectively predict our text classifications.

4.3.2 Document-Concept Graph Attention Networks

We employ Graph Attention Networks, designed explicitly for document and concept nodes in bipartite document-concept graphs to capture the importance of concepts for document classification. The GATs in our models employ a self-attention mechanism to calculate attention weights that determine the importance of neighboring nodes during information propagation. The architecture of our method is illustrated in Figure 4.2. In this figure, $\mathbf{H}_D^{(0)} = \mathbf{X}_D$ and $\mathbf{H}_C^{(0)} = \mathbf{X}_C$ represent the initial feature matrices of the documents and concepts in the bipartite graph, respectively. These feature matrices are composed of our BERT embeddings as pointed out in subsection 4.2.

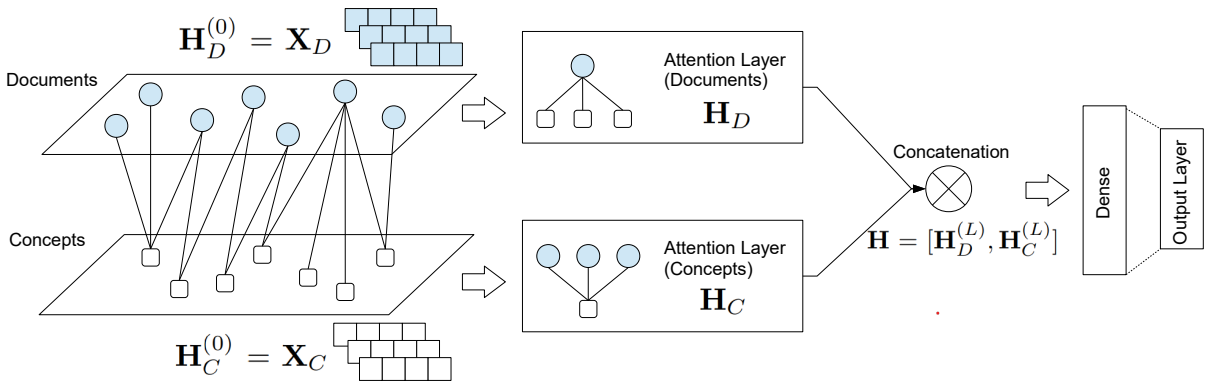


Figure 4.2: Overview of the proposal that explores Graph Attention Networks applied to the document-concept bipartite graph.

In the proposed bipartite graph, GATs utilize a self-attention mechanism to capture the relationships between documents and concepts during the information propagation

process. The self-attention mechanism in GAT involves calculating attention weights that determine the importance of neighboring nodes for each node in the graph. Specifically, for document nodes, the attention weights are computed based on the features of the neighboring concept nodes. Similarly, for concept nodes, the attention weights are computed based on the features of the neighboring document nodes. Consider a document node d_i and its neighboring concept nodes c_j in the bipartite graph. The self-attention mechanism in GAT calculates the attention weights a_{ij} for each neighboring concept node c_j as follows:

$$a_{ij} = \text{softmax}(\mathbf{e}_{ij}), \tag{4.1}$$

where $\mathbf{e}_{ij}$ is a trainable parameter representing the compatibility between the document node d_i and the concept node c_j, as shown in subsection 2.3.2. The softmax function is applied to ensure that the attention weights sum up to 1. The attention weights are the big differential between GAT and GCN since they express different importance for different neighboring nodes.

We use the attention weights a_{ij} to compute the weighted sum of the features of the neighboring concept nodes:

$$\mathbf{h}_i^{'} = \sigma(\sum_{j \in N_{bhd}(n)} \alpha_{ij} \cdot \mathbf{W}^{(l)} \cdot \mathbf{h}_j) \tag{4.2}$$

where $\mathbf{h}_i^{'}$ (GAT layer) represents the updated feature vector for the document node d_i, $\mathbf{h}_j$ is the iterating feature vector of d_i's neighboring concept nodes c_j, and $\mathbf{W}$ is a trainable weight matrix.

This process is repeated for all document nodes, allowing each document node to aggregate information from its neighboring concept nodes based on their respective attention weights. Similarly, for concept nodes, the self-attention mechanism calculates the attention weights based on the features of neighboring document nodes and computes the weighted sum of their features. Thus, at the end of the process, $\mathbf{H}_D$ represents the feature matrix for all documents, and $\mathbf{H}_C$ represents the feature matrix for all concepts.

Formally, for a bipartite document-concept graph, let $\mathbf{H}_D^{(0)} = \mathbf{X}_D$ and $\mathbf{H}_C^{(0)} = \mathbf{X}_C$ be the initial feature matrices for document and concept nodes, respectively. The GAT propagation is performed iteratively as follows:

$$\mathbf{H}_D^{(l+1)} = \text{GAT}_{\text{doc}}^{(l+1)}(\mathbf{H}_D^{(l)}, \hat{\mathbf{A}}_{DC}), \tag{4.3}$$

$$\mathbf{H}_C^{(l+1)} = \text{GAT}_{\text{con}}^{(l+1)}(\mathbf{H}_C^{(l)}, \hat{\mathbf{A}}_{CD}), \tag{4.4}$$

where $\mathbf{H}_D^{(l+1)}$ and $\mathbf{H}_C^{(l+1)}$ represent the feature matrices for document and concept nodes at iteration $l+1$, respectively. $\text{GAT}_{\text{doc}}^{(l+1)}$ and $\text{GAT}_{\text{con}}^{(l+1)}$ denote the GAT layers at iteration $l+1$ specifically designed for document and concept nodes, respectively. The adjacency matrices $\hat{\mathbf{A}}_{DC}$ and $\hat{\mathbf{A}}_{CD}$ capture the relationships between documents and concepts in the bipartite graph. It is important to point out that each GAT layer GAT_{doc} and $\text{GAT}_{\text{con}}^{(l+1)}$ will present its own attention weights α_{ij}, meaning nodes are treated differently between documents and concepts, therefore a node that has high relevance for documents nodes updates do not necessarily have the same importance for concept nodes

After L iterations of GAT propagation, we obtain the final feature matrices $\mathbf{H}_D^{(L)}$ and $\mathbf{H}_C^{(L)}$ for document and concept nodes, respectively. These feature matrices encode the learned representations incorporating the relationships between documents and concepts in the bipartite graph.

Finally, the graph representation $\mathbf{H} = [\mathbf{H}_D^{(L)}, \mathbf{H}_C^{(L)}]$ is fed into a classification layer to predict the labels for all documents. The model is trained using a semi-supervised learning approach, where the labeled documents are used to compute the classification loss, and the unlabeled documents contribute to the overall learning process by leveraging the learned concept-document relationships. The parameters of the GAT and classification layers are optimized jointly to minimize the classification loss and improve the model's predictive performance.

4.4 Training GAT and GCN Models

We utilized the transductive training setup to evaluate the effectiveness of our proposed document-concept bipartite Graph Attention model for text classification. We compared it with the Graph Convolutional Network model also built on document-concept bipartite graphs. We trained both models on the bipartite graphs created at subsection 4.2, considering different keyphrases, numbers of labeled instances per class, and repeated multiple iterations.

For concept extraction, as mentioned in subsection 4.2, we used the KeyBERT library to extract keyphrases and map concepts of different word numbers: two words (keyphrase=2), three words (keyphrase=3), and two or three words (keyphrase=(2,3)). These separate keyphrases created separate graphs, each representing the relationships and patterns associated with different concepts per document (i.e., concept sizes, numbers, and semantic nature itself). Each selected keyphrase graph passes through the same training process.

After generating the graphs and defining the models' architecture, as per subsection 4.3, we conducted transductive training for each case. We trained five cases for each

generated graph and model, varying the number of labeled instances ($n_{labeled}$) in each class. The $n_{labeled}$ cases included 1, 5, 10, 20, and 30 labeled instances randomly selected. The only exception was for the CSTR dataset, which, due to data unbalancing, one class had only 27 documents, therefore not allowing the realization of the $n_{labeled} = 30$ test case.

During training, the models learned to classify the unlabeled instances based on the information from the labeled instances and the graph structure. To ensure the robustness of the results, we repeated the training process for each keyphrase and $n_{labeled}$ combination for a total of ten consecutive iterations and recorded the resulting classification reports.

In total, considering all the number of keyphrases (2,3 and 2-3), labeled instances (1, 5, 10, 20, and 30), and repeated iterations (10), we trained a total of 150 models for GAT and GCN on each text collection. Considering we used 12 collections, and CSTR did not include the $n_{labeled} = 30$, we trained 1770 models for GAT e GCN, as presented in figure 4.3.

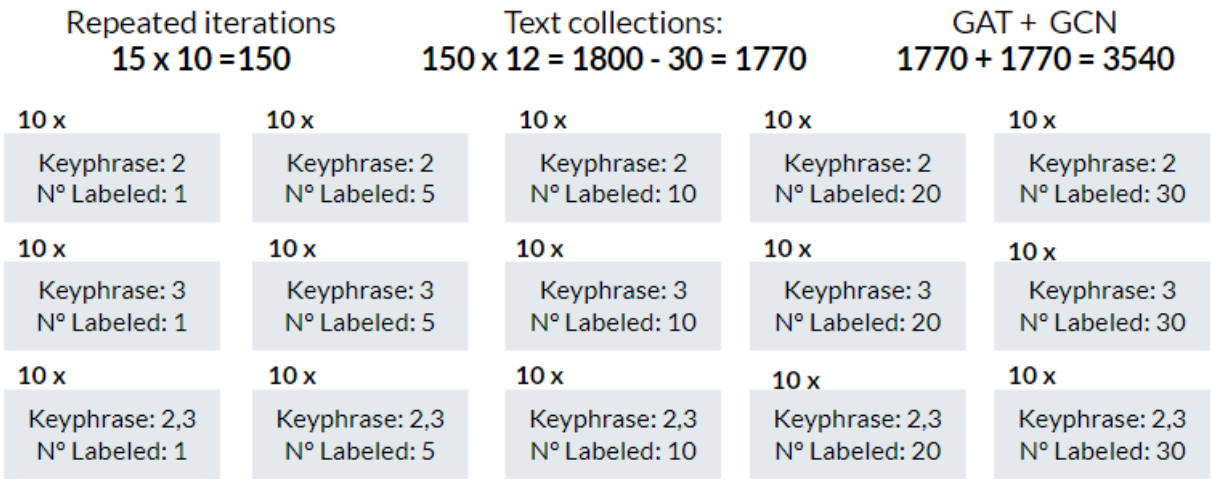


Figure 4.3: GAT and GCN trained models count.

The classification reports provided various metrics such as accuracy, f1-score, precision, recall, and support, allowing us to evaluate the models' performance. We also calculated the average and standard deviation across all ten training runs to assess the consistency and stability of the model's performance.

This comprehensive analysis gave us insights into the overall performance of the document-concept bipartite GAT and GCN models and their consistency across multiple iterations and concept sizes.

Moreover, after completing the application of coarsening, subsection 4.5, and Large Language Models, subsection 4.6, methodologies addressed below, their resulting graphs (i.e., coarsened graphs and LLM-labeled graphs) were submitted to the same training processes documented above. Yet, to delve further into the nuances of each employed

technique, small variations of this training regiment were employed, such as, increasing the number of labeled instances ($n_{labeled}$) and training only using LLM-labeled data. All this variation will be introduced in the experimental results section 5 opportunely.

4.5 Coarsening

After defining our document-concept bipartite graph and evaluating our GAT and GCN architectures, we advance on applying graph coarsening. As previously mentioned, our graph reduction process involves the application of the CLPk algorithm [Eduardo Althoff et al., 2023] on bipartite graphs contextualized in subsection 2.4.

Considering the coarsened graph will also be used for semi-supervised training on GAT and GCN, we must first decide which nodes, labeled, not labeled, or both, will go through the coarsening process. We opt to apply coarsening only to the not-labeled nodes. This means we remove labeled instances ($n_{labeled}$ = [1, 5, 10, 20, and 30]) mentioned in subsection 4.4, apply the CLPk graph coarsening, and re-add the removed labeled instances. We argue that by reducing and grouping only not-labeled nodes, we better preserve information from labeled nodes and do not contaminate this information with uncertain data from not-labeled nodes.

Further elaborating, upon CLPk application, the nodes of the document and concept layers are grouped into super nodes by the label propagation technique. A single super-node can represent one or more original nodes. For instance, suppose that we use labeled and not-labeled data for coarsening. The algorithm could group labeled and unlabeled nodes into the same super-node, essentially indirectly labeling the unlabeled nodes. We rebuke this behavior since GNNs are our main classification algorithms evaluated in this research, and they are responsible for efficiently interpreting documents-concept information and relations to perform classification.

The CLPk creates super-nodes based only on graph structural information, node, and edge organization and does not consider documents or concepts, textual data, or embeddings. Therefore, we must add a step for creating super-node embeddings after graph reduction. for this step, we first perform a concatenation of textual information of the nodes that compose a single super-node. Secondly, we apply BERT on the concatenated result to generate the super-node feature embedding.

After consolidating the coarsened graphs and their textual data, we reintegrate the previously removed labeled instances. Lastly, we perform the training and scoring process for GAT and GCN models as per subsection 4.4.

Summarizing the creation of the coarsened graphs.

1. Labeled nodes are withdrawn from the bipartite graph.

2. CLPk algorithm is applied only on unlabeled nodes, creating unlabeled super-nodes based on graph structure.

3. Super-node textual information is concatenated, and BERT is applied, generating embeddings.

4. Labeled nodes are reintegrated into the graph.

5. GATs and GCNs are retrained and evaluated on coarsened graph data.

It is relevant to highlight that we demanded CLPk to reduce graphs by 50% of their original size, however, not in all cases the algorithm can meet this exact reduction size. Also, we performed only one CLPk iteration since we did not observe significant increases in graph reduction or GNNs classification when using more iterations of the coarsening algorithm.

4.6 Large Language Models Labeling

Regarding using LLMs as low-cost data labelers, we mainly aim to observe how GAT and GCN models perform by adding artificially labeled data. As our labeler LLM model, we selected Llama 3.1 foundation model [Dubey et al., 2024], further detailed in subsection 2.5, and made use of the Ollama framework[1] to easily include Llama in our processes. For the LLM application, we have three steps:

1. Study and select the best prompt for labeling each collection.

2. Generate datasets of LLM artificially labeled data with 10, 50, and 100 instances of each class ($LLMn_{labeled} = [10, 50, 100]$) for each keyphrase (2,3 and 2 or 3), number of human-labeled instances ($n_{labeled} = [1, 5, 10, 20, 30]$), iteration (10), and collection.

3. Retraining GAT and GCN models, including only the LLM-labeled datasets and combinations with the Human-labeled datasets.

In the first step, we analyze how to optimally use LLMs to label data by investigating two approaches:

1. **Zero-shot labeling**: employing the LLM to label data without previous examples or particular training cases. Here, we ask the model to provide the classification predictions given a set of possible classes and a document text. A prompt example of this approach is:

[1] `https://ollama.com`

'Classify the text below according to its main subject using only one label from the list: [Artificial, Education, Robotics, Software]'

2. **Few-shot labeling**: In this approach, we will provide the LLM with randomly selected classification examples of different classes before asking for its labeling predictions. These examples are derived from human-labeled instances available on each dataset. A prompt example of this approach is:

'Given the text classification examples below: - Text: 'LibML A machine learning library. New implementations of various machine learning algorithms.' Class: Artificial - Text: 'HexWorks A site about Hamlet, a hexapod autonomous robot.' Class: Robotics Classify the text below according to its main subject using only one label from the list: [Artificial, Education, Robotics, Software]'

One can presume that few-shot labeling is typically more accurate because it provides additional information for Llama's improved functioning. Yet, that is not always the case. LLM foundation models such as Llama are excellent generalist agents and flexibly learn by their prompt data. Still, this flexibility also makes them prone to mistakes and hallucinations, especially in the case of longer or complex prompts due to lengthy example texts. To avoid such occurrences we adjust hyperparameters, setting the model temperature to 0.3 to ensure lower model hallucination without negatively impacting the model interpretability of texts.

Considering the model output, we requested Llama to return only one class name for each document, respecting this scenario, Llama's output would not exceed 2 or 3 tokens. However, models can have hallucinations. Lower levels of hallucinations can still give correct predictions, but in the form of small sentences that don't completely align with the requested output. Therefore, we increased the output limit size to 25 tokens and designed a mechanism to select the first class occurrence on the output text as the predicted class. This way, low hallucinations are also viable outputs.

Yet there are cases of high hallucination, where no class is detected on the output, these cases are considered *null-predictions*. If this happens, Llama is given one more labeling chance at the end of the labeling process, presuming the previous hallucination was a natural lapse. If classification fails again, these cases are then labeled at random. Below, some possible outputs are presented (predicted classes are underlined):

- **Correct requested output**: Class: Artificial
- **Low hallucination output**: I predict this document class to be Artificial
- **High hallucination output**: The text talks about many important questions at the helm of artificial intelligence, education, and robotics. I believe... (No class prediction due to high hallucination exceeding output size)

By following this output logic and the previously shown zero-shot and few-shot labeling prompts, we perform the first step, evaluating how both prompts perform on each text collection. Both prompts are tested on the smaller class case, $LLMn_{labeled} = 10$ instances of each class, repeating 10 iterations to ensure robustness on generated label predictions. Based on the mean accuracy of the predictions, we select the best prompt type for each text collection. Table 4.2 presents this analysis searching the most accurate prompts.

Table 4.2 focuses on selecting each prompt, maximizing accuracy by investigating prompts with a higher hit rate and minimizing the number of *null-predictions* by escaping cases where the prompts could not make class predictions due to high hallucination. Oriented by the results of Table 4.2, we selected the best-performing prompts for each collection, represented by the colors blue and green.

Below, we list all selected prompts and demonstrate their structure. Each prompt is composed of a brief description of the dataset, the task requested, and lists of labeling classes. Only in the case of few-shot learning, we also include annotation examples randomly selected from the human-labeled instances ($n_{labeled}$), because of this random nature these examples are not explicitly shown in the prompt samples below.

	zero-shot - accuracy	zero-shot - null predictions	few-shot:- accuracy	few-shot - null predictions	Best Prompt
CSTR	**0.797500**	**1.1**	0.765000	0.0	zero-shot
Dmoz_Computers	**0.484211**	**0.0**	0.467368	0.0	zero-shot
Dmoz_Health	0.605385	0.0	**0.753846**	**0.0**	few-shot
Dmoz_Science	0.672500	0.0	**0.692500**	**0.1**	few-shot
Dmoz_Sports	0.846667	2.3	**0.855556**	**2.4**	few-shot
Industry_Sector	**0.439167**	**4.9**	0.295833	12.8	zero-shot
NSF	0.630625	1.5	**0.673125**	**0.2**	few-shot
SyskillWebert	**0.682500**	**1.7**	0.620000	5.4	zero-shot
classic4	0.630000	0.0	**0.817500**	**0.2**	few-shot
re8	0.666250	0.0	**0.718750**	**0.0**	few-shot
review_polarity	**0.935000**	**0.1**	0.605000	5.0	zero-shot
webkb_parsed	**0.644286**	**1.0**	0.600000	4.3	zero-shot

Table 4.2: Few-shot and Zero-shot prompt evaluation. The selected best-performing prompts are colored in blue and green.

- Zero-shot CSTR:

You are a text classification AI. Your job is to classify documents from the CSTR (Computer Science Technical Reports) collection, composed of abstracts and technical reports published in the Department of Computer Science at University of Rochester from 1991 to 2007. The documents belong to 4 areas: Natural Language Processing, Robotics/Vision, Systems, and Theory.
The collection classes are the following:
[‘ArtificiallIntelligence’, ‘Robotics’, ‘Systems’, ‘Theory’]
You must not talk to the user. Limit your response to only the predicted class.

- Zero-shot Dmoz_Computers:

You are a text classification AI. Your job is to classify documents from the Dmoz-Computers-500 collection, composed of web pages of the computers category extracted from DMOZ - Open Directory Project. The document classes are the subcategories of the computers category.
The collection classes are the following:
[‘Artificial_Intelligence’, ‘CAD_and_CAM’, ‘Companies’, ‘Computer_Science’ ‘Consultants’, ‘Data_Formats’, ‘Data_Communications’, ‘Education’, ‘Graphics’ ’Hardware’, ‘Internet’, ‘Mobile_Computing’, ‘Multimedia’, ‘Open_Source’ ’Programming’, ‘Robotics’, ‘Security’, ‘Software’, ‘Systems’]
You must not talk to the user. Limit your response to only the predicted class.

- Few-shot Dmoz_Health:

You are a text classification AI. Your job is to classify documents from the Dmoz-Health-500 collection, composed of web pages of the health category extracted from DMOZ - Open Directory Project. The document classes are the subcategories of the health category.
The collection classes are the following:
[’Addictions’, ‘Alternative’, ‘Animal’, ‘Conditions_and_Diseases’, ‘Medicine’ ’Mental_Health’, ‘Nursing’, ‘Nutrition’, ‘Pharmacy’, ‘Professions’ ’Public_Health_and_Safety’, ‘Reproductive_Health’, ‘Senior_Health’]
You must not talk to the user. Limit your response to only the predicted class.
Below are examples of your task:

- Few-shot Dmoz_Science:

> You are a text classification AI. Your job is to classify documents from the Dmoz-Science-500 collection, composed of web pages of the science category extracted from DMOZ - Open Directory Project. The document classes are the subcategories of the science category.
> The collection classes are the following:
> ['Agriculture', 'Astronomy', 'Biology', 'Chemistry', 'Earth_Sciences' 'Environment', 'Instruments_and_Supplies', 'Math', 'Physics' 'Sciencez_in_Society', 'Social_Sciences', 'Technology']
> You must not talk to the user. Limit your response to only the predicted class. Below are examples of your task:

- Few-shot Dmoz_Sports:

> You are a text classification AI. Your job is to classify documents from the Dmoz-Sports-500 collection, composed of web pages of the sports category extracted from DMOZ - Open Directory Project. The document classes are the subcategories of the sports category.
> The collection classes are the following:
> ['Baseball', 'Basketball', 'Bowling', 'Cricket', 'Cycling', 'Equestrian' 'Fencing', 'Flying_Discs', 'Football', 'Golf', 'Gymnastics', 'Hockey' 'Lacrosse', 'Martial_Arts', 'Motorsports', 'Paintball', 'Running', 'Skating' 'Soccer', 'Softball', 'Strength_Sports', 'Tennis', 'Track_and_Field' 'Volleyball', 'Water_Sports', 'Winter_Sports', 'Wrestling']
> You must not talk to the user. Limit your response to only the predicted class. Below are examples of your task:

- Zero-shot Industry_Sector:

> You are a text classification AI. Your job is to classify documents from the Industry-Sector collection, composed of web pages of companies from various economic sectors.
> The collection classes are the following:
> ['basic_materials', 'capital_goods', 'conglomerates_industry' 'consumer_cyclical', 'consumer_non-cyclical', 'energy', 'financial' 'healthcare', 'services', 'technology', 'transportation', 'utilities']
> You must not talk to the user. Limit your response to only the predicted class

- Few-shot NSF:

> You are a text classification AI. Your job is to classify documents from the NSF (National Science Foundation) collection, composed of abstracts of grants awarded by the National Science Foundation8 between 1999 and August 2003.
> The collection classes are the following:
> ['data_management', 'ecology', 'economic', 'geophysics', 'gravitional_theory' 'hydro', 'math', 'metals', 'networking', 'neuroscience', 'oceanography' 'politic', 'sociology', 'software_engineering', 'statistics' 'theory_computing']
> You must not talk to the user. Limit your response to only the predicted class. Below are examples of your task:

- Zero-shot SyskillWebert:

> You are a text classification AI. Your job is to classify documents from the SyskillWebert collection, composed of web pages about bands, sheeps, goats, and biomedical.
> The collection classes are the following:
> ['Bands', 'BioMedical', 'Goats', 'Sheep']
> You must not talk to the user. Limit your response to only the predicted class.

- Few-shot classic4:

> You are a text classification AI. Your job is to classify documents from the Classic4 collection, composed of 4 distinct collections: CACM (titles and abstracts from the journal Communications of the ACM), CISI (information retrieval papers), CRANFIELD (aeronautical system papers), and MEDLINE (medical journals).
> The collection classes are the following:
> ['cacm', 'cisi', 'cran', 'med']
> You must not talk to the user. Limit your response to only the predicted class. Below are examples of your task:

- Few-shot re8:

> You are a text classification AI. Your job is to classify documents from the Re8 collection, composed of articles from Reuters-21578 collection.
> The collection classes are the following:
> ['acq', 'crude', 'earn', 'grain', 'interest', 'money', 'ship', 'trade']
> You must not talk to the user. Limit your response to only the predicted class. Below are examples of your task:

- Zero-shot review_polarity:

> You are a text classification AI. Your job is to classify documents from the Review-Polarity collection, composed of 1000 positive reviews and 1000 negative reviews about movies.
> The collection classes are the following:
> ['neg', 'pos']
> You must not talk to the user. Limit your response to only the predicted class.

- Zero-shot webkb_parsed:

> You are a text classification AI. Your job is to classify documents from the WebKB collection, composed of web pages collected from computer science departments of various universities in January 1997 by the World Wide Knowledge Base (WebKb) project of the CMU Text Learning Group.
> The collection classes are the following:
> ['course', 'department', 'faculty', 'other', 'project', 'staff', 'student']
> You must not talk to the user. Limit your response to only the predicted class.

Having defined the best usable prompts, we used them as the labeling prompts to label $LLMn_{labeled} = 50$ and $LLMn_{labeled} = 100$ instances for each class, considering the first case, with $LLMn_{labeled} = 10$ instances, was already covered while evaluating prompts. In specific cases where the class overall count presents less than 50 or 100 occurrences in the entire text collection, we limit the Llama to label to only half of this specific class count.

Finally, after labeling all these instances, we retrain GAT and GCN models using the LLM-labeled data of 10, 50, and 100 instances by themselves and combined with human-labeled data of 1,5,10,20, and 30 instances as per subsection 4.4.

4.7 Evaluation

We present and discuss the experimental results, focusing primarily on the average F1-Score of the models trained for each dataset. F1-Score is a metric widely used to evaluate the performance of classification models, mainly when dealing with imbalanced datasets. It quantifies the balance between precision and recall [Sasaki, 2007]. Precision represents the accuracy of positive predictions made by the model, identifying how many positive predictions the model was correct. Recall measures the model's ability to identify all positive instances in the dataset, effectively how many positive predictions the model captured. The F1-score is calculated as the harmonic mean of precision and recall, pro-

viding a single numerical value that reflects both metrics and penalizes extreme values of precision or recall.

One reason the F1-score is favored as an evaluation metric is its effectiveness in handling class imbalance [Goutte and Gaussier, 2005]. In datasets where one class significantly outnumbers the other, accuracy alone can be misleading, as a classifier might predict the majority class most of the time. However, the F1-score considers false positives and false negatives equally, making it robust in scenarios where both types of errors are critical, thus providing a more comprehensive assessment of model performance.

Considering we have a multi-class classification problem, we used the macro average F1-score that calculates the F1-score independently for each class and later averages the resulting scores. The macro average treats all classes equally, regardless of size. This makes it particularly useful when evaluating models on datasets with imbalanced classes, where minority classes are of interest. This approach ensures that performance in smaller classes is not overshadowed by the dominance of larger classes, offering a balanced evaluation metric for datasets with uneven distributions.

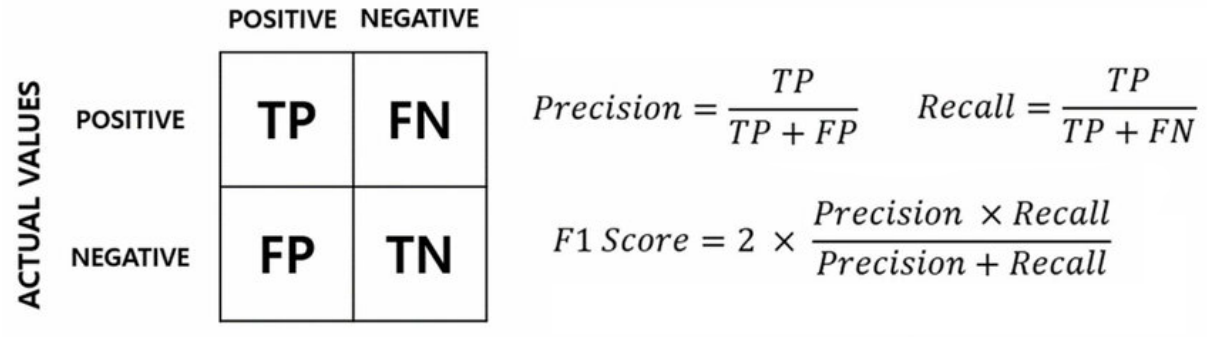


Figure 4.4: Overview of the F1-socre, precision and recall metrics formulas, image based on [Seol et al., 2023].

We also analyzed the F1-score and the critical difference (CD) diagram using Friedman's test with Nemenyi's posttest with a 95% confidence level. The CD is a metric established at [Demšar, 2006] and determines if one or more values in a specific domain are statistically different. Simply put, it represents the real statistical difference between two or more values.

The CD value is calculated considering each algorithm's results and their relative difference. This value represents a critical difference threshold that, if surpassed by the relative difference between algorithms, makes it possible to affirm their statistical difference. Suppose the distance between the two values, in our case, the models' resulting F1-score, is superior to the CD's value. In that case, that one can be considered statistically superior or inferior.

Chapter 5

Experimental Results

This section presents our study's results based on the methodology previously established in subsection 4.4. We evaluate the performance of the Graph Convolutional Network and Graph Attention Network models in a bipartite graph-based document classification task using transductive learning and various iterations and combinations of labeled document numbers, concept sizes (i.e., keyphrases), and text collections.

The methodology of subsection 4.4 is firstly examined simply by evaluating GCN and GAT results on the proposed document-concept graphs. Secondly, it explores the combination of GCN and GAT transductive capacities with graph coarsening to reduce computational costs. Lastly, it combines GCN and GAT with Large Language Model labeling to reduce annotation costs.

5.1 Document-Concept GAT and GCN

First, we assess how the proposed GAT and the classic GCN perform on our designed approach of bipartite document-concept graphs, especially their abilities to interpret concepts (i.e., keyphrases) and document relationships.

Table 5.1 shows the macro F1-score metric of both models for each dataset, keyphrase, and number of labeled documents. We can observe that Graph Attention Network models achieved, in the great majority of cases, better results, shown in bold on the table, when compared to the traditional Graph Convolutional Network that obtained the highest F1-score only once, in the *CSTR* collection with $n_{labeled} = 1$.

Figure 5.1 represents this same comparison, but in the form of an evolutionary graph that follows the progression of the labeled document numbers, showing how GAT and GCN perform by adding more labeled instances ($n_{labeled} = [1, 5, 10, 20, 30]$) at each dataset. Here it is easier to grasp GAT gains in reason of GCN at each dataset, and also possible to observe that F1-score values sharply spike with the addition of only a few labeled data.

Figure 5.2 shows a statistical analysis using the critical difference CD diagram computed in all twelve datasets. In the charts of Figure 5.2, our CD values are represented by a measuring line in the top left of each chart, and the models that do not have a critical difference higher than this value are grouped by a highlighted bold line. Thus, any models out of this grouping of bold lines are statistically different.

Observing Figure 5.2, GAT models present higher F1-score values and consequently are represented in the right section of the charts as a group. In contrast, GCNs, with lower performance values, are represented on the left. Yet there were two cases where the best performing GCN models had no statistical difference from the worst performing GAT models, $n_{labeled} = 1$ and $n_{labeled} = 30$.

Considering keyphrase numbers, we observed that keyphrase = 3 achieved the best scores in both GAT and GCN models, followed by keyphrase = (2,3) in most cases, and lastly, keyphrase = 2. Therefore, it is arguable that lengthier keyphrases can bring more information to the concept-based graph models and increase performance.

Regarding datasets, they presented similar behaviors, considering the evolution of labeled instances ($n_{labeled} = [1, 5, 10, 20, 30]$). The main outliers were *Industry_Sector* and *webkb_parsed* where GAT presented a more significant performance increase with greater labeled instances numbers, and *review_polarity*, our binary dataset, that presented a very variable behavior. The higher F1-score occurred on *classic4*, averaging 0.8711 across the dataset models, and the lower F1-score was 0.2887 on the *webkb_parsed* dataset. This variation of F1-scores may not occur necessarily based on dataset characteristics (size, class percentages) but simply because the classification task on this dataset is more challenging.

As an additional observation, we simulated how GAT and GCN would perform with more labeled documents. For this, we tested F1-Score considering $n_{labeled}$ = [20%, 40%, 60%, 80%] of labeled data to each dataset, being 80% a standard training size for inductive learning algorithms. Figure 5.3 details this experiment, we can observe that, in most cases, the further the increase in the number of labeled instances the lower the actual gain on model performance is, showing that transductive models gain significant performance even with smaller training data and adding much training data might not be substantially beneficial. Detailed results of the percentage GAT and GCN models are in table 5.2.

Lastly, we compared the best-performing GAT and GCN transductive models to the best-performing inductive models benchmarked by Rossi et al. [Rossi et al., 2013]. Rossi et al. models were Naïve Bayes [Vikramkumar et al., 2014], Multinomial Naïve Bayes algorithm [Xu et al., 2017], J48, which is an implementation of the C.45 classification tree algorithm [Quinlan, 2014], Sequential Minimal Optimization (SMO), that optimizes the process of the Support Vector Machines (SVM) [Platt, 1998] and, IBk, an implementation

of the k-Nearest Neighbors algorithm [Cunningham and Delany, 2021].

Table 5.3 presented this visualization, in bold are the best of each group of models. Transductive models performed similarly to the inductive ones, achieving comparable results despite having considerably less labeled data for training. By adding more data, transductive models could surpass, as shown by the $n_{labeled} = [20\%, 40\%, 60\%, 80\%]$ GAT and GCN models, the best-performing inductive ones.

In summary, our proposed GAT method consistently achieved higher F1-Score values than GCN models. This demonstrates that GAT effectively captures the importance of concepts for document classification, leading to improved results. This superiority is particularly evident when the number of labeled data equals 5 and 20 instances per class. Regarding the comparison of the GNN transductive and traditional inductive models, the GNNs presented results that were on par and even superior to the more well-known traditional models. This represents that the proposed approach achieves great classification results while using a smaller and more practical amount of labeled data for real-world applications, where labeling is often expensive or dependent on domain experts.

Dataset	Keyphrases	GNN Model	Number of Labeled Data					Average
			1	5	10	20	30	Performance
CSTR	[2,3]	GAT	0.5803 ± 0.132	0.812 ± 0.047	0.8361 ± 0.049	0.8114 ± 0.025	-	**0.76 ± 0.063**
CSTR	[2,3]	GCN	**0.595 ± 0.214**	0.795 ± 0.049	0.8193 ± 0.054	0.8015 ± 0.047	-	0.7527 ± 0.091
CSTR	[2]	GAT	0.5358 ± 0.12	0.8243 ± 0.031	**0.8474 ± 0.02**	**0.816 ± 0.031**	-	0.7559 ± 0.051
CSTR	[2]	GCN	0.5852 ± 0.189	0.7964 ± 0.052	0.8068 ± 0.055	0.8031 ± 0.04	-	0.7479 ± 0.084
CSTR	[3]	GAT	0.5431 ± 0.112	**0.828 ± 0.046**	0.8331 ± 0.045	0.8114 ± 0.037	-	0.7539 ± 0.06
CSTR	[3]	GCN	0.5729 ± 0.135	0.8225 ± 0.046	0.8114 ± 0.043	0.7906 ± 0.064	-	0.7494 ± 0.072
Dmoz_Computers	[2,3]	GAT	0.2604 ± 0.031	0.4991 ± 0.013	0.5571 ± 0.011	0.5937 ± 0.009	0.6145 ± 0.008	0.505 ± 0.014
Dmoz_Computers	[2,3]	GCN	0.2034 ± 0.044	0.4563 ± 0.023	0.5311 ± 0.015	0.5708 ± 0.01	0.5861 ± 0.01	0.4695 ± 0.021
Dmoz_Computers	[2]	GAT	0.2617 ± 0.035	0.4898 ± 0.014	0.5506 ± 0.014	0.5922 ± 0.01	0.6106 ± 0.009	0.501 ± 0.016
Dmoz_Computers	[2]	GCN	0.2122 ± 0.046	0.4519 ± 0.019	0.5191 ± 0.015	0.5625 ± 0.01	0.5791 ± 0.011	0.465 ± 0.02
Dmoz_Computers	[3]	GAT	**0.2752 ± 0.034**	**0.502 ± 0.011**	**0.5614 ± 0.013**	**0.5996 ± 0.008**	**0.6193 ± 0.007**	**0.5115 ± 0.015**
Dmoz_Computers	[3]	GCN	0.2231 ± 0.039	0.4642 ± 0.018	0.5345 ± 0.013	0.5779 ± 0.011	0.6005 ± 0.007	0.4801 ± 0.018
Dmoz_Health	[2,3]	GAT	0.4599 ± 0.067	0.7133 ± 0.016	0.7716 ± 0.011	**0.7996 ± 0.01**	**0.8114 ± 0.006**	0.7112 ± 0.022
Dmoz_Health	[2,3]	GCN	0.3588 ± 0.073	0.6139 ± 0.037	0.7187 ± 0.016	0.7671 ± 0.015	0.7878 ± 0.007	0.6493 ± 0.03
Dmoz_Health	[2]	GAT	0.4689 ± 0.058	0.7131 ± 0.022	0.7714 ± 0.011	0.7972 ± 0.008	0.8063 ± 0.004	0.7114 ± 0.021
Dmoz_Health	[2]	GCN	0.3854 ± 0.083	0.6532 ± 0.035	0.7156 ± 0.013	0.7635 ± 0.014	0.7782 ± 0.008	0.6592 ± 0.031
Dmoz_Health	[3]	GAT	**0.4741 ± 0.062**	**0.7141 ± 0.018**	**0.7739 ± 0.013**	0.7985 ± 0.008	**0.8114 ± 0.006**	**0.7144 ± 0.021**
Dmoz_Health	[3]	GCN	0.3704 ± 0.056	0.6489 ± 0.025	0.7244 ± 0.023	0.7697 ± 0.011	0.7949 ± 0.009	0.6617 ± 0.025
Dmoz_Science	[2,3]	GAT	0.3539 ± 0.027	0.5965 ± 0.026	0.6564 ± 0.017	0.6966 ± 0.011	0.7167 ± 0.01	0.604 ± 0.018
Dmoz_Science	[2,3]	GCN	0.2954 ± 0.032	0.539 ± 0.03	0.6211 ± 0.019	0.6833 ± 0.013	0.7022 ± 0.01	0.5682 ± 0.021
Dmoz_Science	[2]	GAT	0.3566 ± 0.043	**0.6053 ± 0.022**	0.6546 ± 0.017	0.6986 ± 0.011	0.7155 ± 0.011	0.6061 ± 0.021
Dmoz_Science	[2]	GCN	0.3018 ± 0.058	0.5461 ± 0.038	0.6217 ± 0.014	0.6469 ± 0.111	0.7008 ± 0.013	0.5635 ± 0.047
Dmoz_Science	[3]	GAT	**0.3596 ± 0.031**	0.6029 ± 0.03	**0.6564 ± 0.014**	**0.7016 ± 0.01**	**0.7211 ± 0.008**	**0.6083 ± 0.019**
Dmoz_Science	[3]	GCN	0.3162 ± 0.041	0.5481 ± 0.038	0.6326 ± 0.021	0.6883 ± 0.013	0.6862 ± 0.079	0.5743 ± 0.038
Dmoz_Sports	[2,3]	GAT	0.4402 ± 0.038	0.734 ± 0.009	0.7751 ± 0.01	0.8043 ± 0.004	0.8212 ± 0.004	0.715 ± 0.013
Dmoz_Sports	[2,3]	GCN	0.3202 ± 0.039	0.6542 ± 0.019	0.7112 ± 0.009	0.7554 ± 0.008	0.7735 ± 0.004	0.6429 ± 0.016
Dmoz_Sports	[2]	GAT	**0.4569 ± 0.03**	0.7356 ± 0.009	0.7763 ± 0.009	0.806 ± 0.005	0.8231 ± 0.004	0.7196 ± 0.011
Dmoz_Sports	[2]	GCN	0.3384 ± 0.061	0.6567 ± 0.015	0.7104 ± 0.013	0.7446 ± 0.005	0.7598 ± 0.005	0.642 ± 0.02
Dmoz_Sports	[3]	GAT	0.4464 ± 0.036	**0.7385 ± 0.008**	**0.7779 ± 0.011**	**0.8158 ± 0.006**	**0.8275 ± 0.004**	**0.7212 ± 0.013**
Dmoz_Sports	[3]	GCN	0.3252 ± 0.032	0.6607 ± 0.024	0.725 ± 0.01	0.7639 ± 0.006	0.7838 ± 0.006	0.6517 ± 0.016
Industry_Sector	[2,3]	GAT	0.123 ± 0.03	0.31 ± 0.035	**0.3901 ± 0.022**	0.4528 ± 0.012	0.4813 ± 0.015	0.3514 ± 0.023
Industry_Sector	[2,3]	GCN	0.1 ± 0.022	0.2261 ± 0.025	0.2996 ± 0.028	0.3104 ± 0.072	0.3452 ± 0.064	0.2563 ± 0.042
Industry_Sector	[2]	GAT	**0.1344 ± 0.032**	0.2964 ± 0.039	0.3845 ± 0.022	0.4497 ± 0.009	0.4772 ± 0.017	0.3485 ± 0.024
Industry_Sector	[2]	GCN	0.1035 ± 0.026	0.2052 ± 0.022	0.2865 ± 0.022	0.2451 ± 0.097	0.2436 ± 0.11	0.2168 ± 0.056
Industry_Sector	[3]	GAT	0.1253 ± 0.028	**0.3102 ± 0.034**	0.3896 ± 0.02	**0.4536 ± 0.015**	**0.4827 ± 0.013**	**0.3523 ± 0.022**
Industry_Sector	[3]	GCN	0.107 ± 0.022	0.2189 ± 0.033	0.2788 ± 0.051	0.24 ± 0.119	0.3123 ± 0.098	0.2314 ± 0.065
NSF	[2,3]	GAT	0.5372 ± 0.055	0.7409 ± 0.02	0.782 ± 0.008	0.7869 ± 0.014	0.8055 ± 0.007	0.7305 ± 0.021
NSF	[2,3]	GCN	0.4999 ± 0.056	0.6955 ± 0.028	0.7558 ± 0.019	0.7589 ± 0.017	0.7783 ± 0.012	0.6977 ± 0.026
NSF	[2]	GAT	0.532 ± 0.051	0.7367 ± 0.017	0.7717 ± 0.012	0.7813 ± 0.012	0.7997 ± 0.008	0.7243 ± 0.02
NSF	[2]	GCN	0.4804 ± 0.043	0.6925 ± 0.041	0.7454 ± 0.012	0.7561 ± 0.013	0.7666 ± 0.017	0.6882 ± 0.025
NSF	[3]	GAT	**0.5573 ± 0.047**	**0.7434 ± 0.016**	**0.783 ± 0.009**	**0.7876 ± 0.012**	**0.8072 ± 0.006**	**0.7357 ± 0.018**
NSF	[3]	GCN	0.4929 ± 0.068	0.6969 ± 0.033	0.7557 ± 0.013	0.7628 ± 0.014	0.7794 ± 0.014	0.6975 ± 0.028
SyskillWebert	[2,3]	GAT	0.6357 ± 0.08	0.8469 ± 0.045	0.8779 ± 0.021	0.9067 ± 0.013	0.9016 ± 0.02	0.8338 ± 0.036
SyskillWebert	[2,3]	GCN	0.5628 ± 0.122	0.7528 ± 0.032	0.8231 ± 0.025	0.8497 ± 0.045	0.8574 ± 0.028	0.7692 ± 0.05
SyskillWebert	[2]	GAT	**0.6547 ± 0.08**	**0.8557 ± 0.033**	0.8845 ± 0.023	**0.9126 ± 0.019**	**0.9044 ± 0.022**	**0.8424 ± 0.035**
SyskillWebert	[2]	GCN	0.5795 ± 0.134	0.7352 ± 0.057	0.8391 ± 0.04	0.8635 ± 0.034	0.8499 ± 0.039	0.7734 ± 0.061
SyskillWebert	[3]	GAT	0.5815 ± 0.085	0.8433 ± 0.044	**0.8894 ± 0.014**	0.901 ± 0.018	0.9026 ± 0.02	0.8236 ± 0.036
SyskillWebert	[3]	GCN	0.5535 ± 0.099	0.7421 ± 0.06	0.8208 ± 0.027	0.8294 ± 0.045	0.8637 ± 0.023	0.7619 ± 0.051
classic4	[2,3]	GAT	0.7612 ± 0.085	0.8866 ± 0.045	0.9249 ± 0.013	0.9343 ± 0.02	0.94 ± 0.016	0.8894 ± 0.036
classic4	[2,3]	GCN	0.6576 ± 0.133	0.7978 ± 0.074	0.8774 ± 0.031	0.9193 ± 0.023	0.9188 ± 0.022	0.8342 ± 0.057
classic4	[2]	GAT	0.7708 ± 0.081	**0.896 ± 0.044**	**0.9263 ± 0.012**	0.9394 ± 0.013	0.9394 ± 0.017	0.8944 ± 0.033
classic4	[2]	GCN	0.7269 ± 0.121	0.8422 ± 0.08	0.9118 ± 0.026	0.8916 ± 0.093	0.9285 ± 0.02	0.8602 ± 0.068
classic4	[3]	GAT	**0.7956 ± 0.059**	0.8914 ± 0.046	0.923 ± 0.017	**0.9405 ± 0.014**	**0.9419 ± 0.013**	**0.8985 ± 0.03**
classic4	[3]	GCN	0.6863 ± 0.121	0.8114 ± 0.053	0.9084 ± 0.023	0.9209 ± 0.022	0.9243 ± 0.028	0.8503 ± 0.049
re8	[2,3]	GAT	0.5199 ± 0.101	0.7503 ± 0.035	0.811 ± 0.034	0.8257 ± 0.025	**0.8208 ± 0.025**	0.7455 ± 0.044
re8	[2,3]	GCN	0.4231 ± 0.064	0.6915 ± 0.078	0.7499 ± 0.042	0.763 ± 0.051	0.7487 ± 0.048	0.6752 ± 0.056
re8	[2]	GAT	0.5124 ± 0.073	0.7339 ± 0.023	0.7958 ± 0.034	0.8166 ± 0.026	0.8129 ± 0.026	0.7343 ± 0.036
re8	[2]	GCN	0.436 ± 0.087	0.637 ± 0.061	0.7014 ± 0.061	0.7022 ± 0.047	0.707 ± 0.043	0.6367 ± 0.06
re8	[3]	GAT	**0.5358 ± 0.078**	**0.7561 ± 0.034**	**0.8112 ± 0.03**	**0.8304 ± 0.03**	0.8185 ± 0.03	**0.7504 ± 0.04**
re8	[3]	GCN	0.475 ± 0.102	0.6734 ± 0.087	0.7397 ± 0.05	0.7116 ± 0.103	0.7205 ± 0.043	0.664 ± 0.077
review_polarity	[2,3]	GAT	0.4716 ± 0.042	**0.4816 ± 0.09**	0.4923 ± 0.086	**0.5503 ± 0.025**	0.5666 ± 0.027	**0.5125 ± 0.054**
review_polarity	[2,3]	GCN	0.4437 ± 0.067	0.4393 ± 0.093	0.4731 ± 0.108	0.4404 ± 0.112	0.4507 ± 0.137	0.4494 ± 0.103
review_polarity	[2]	GAT	**0.4783 ± 0.051**	0.4792 ± 0.079	**0.5226 ± 0.025**	0.5112 ± 0.096	0.5656 ± 0.029	0.5114 ± 0.056
review_polarity	[2]	GCN	0.4433 ± 0.09	0.4375 ± 0.111	0.4291 ± 0.097	0.4728 ± 0.127	0.4094 ± 0.118	0.4384 ± 0.109
review_polarity	[3]	GAT	0.4134 ± 0.053	0.4763 ± 0.083	0.5092 ± 0.065	0.5324 ± 0.075	**0.5689 ± 0.025**	0.5001 ± 0.06
review_polarity	[3]	GCN	0.4047 ± 0.06	0.3982 ± 0.095	0.4254 ± 0.109	0.4463 ± 0.129	0.3933 ± 0.127	0.4136 ± 0.104
webkb_parsed	[2,3]	GAT	0.1601 ± 0.052	0.3243 ± 0.023	0.3704 ± 0.035	0.4144 ± 0.011	0.4454 ± 0.015	0.3429 ± 0.027
webkb_parsed	[2,3]	GCN	0.1156 ± 0.035	0.2535 ± 0.026	0.2699 ± 0.081	0.2222 ± 0.08	0.3181 ± 0.093	0.2358 ± 0.063
webkb_parsed	[2]	GAT	**0.1694 ± 0.052**	0.3142 ± 0.021	**0.374 ± 0.02**	0.4136 ± 0.012	0.4406 ± 0.018	0.3424 ± 0.024
webkb_parsed	[2]	GCN	0.1344 ± 0.057	0.2625 ± 0.041	0.2427 ± 0.075	0.2738 ± 0.096	0.2213 ± 0.091	0.2269 ± 0.072
webkb_parsed	[3]	GAT	0.1596 ± 0.052	**0.3287 ± 0.03**	0.3722 ± 0.026	**0.416 ± 0.012**	**0.446 ± 0.017**	**0.3445 ± 0.027**
webkb_parsed	[3]	GCN	0.1485 ± 0.039	0.2512 ± 0.042	0.2691 ± 0.077	0.2937 ± 0.069	0.2368 ± 0.105	0.2398 ± 0.067

Table 5.1: All GAT and GCN macro F1-Socre results. The numbers represent the average and standard deviation of all ten iterations for each dataset, separated by the number of labeled data, keyphrases, model, and dataset.

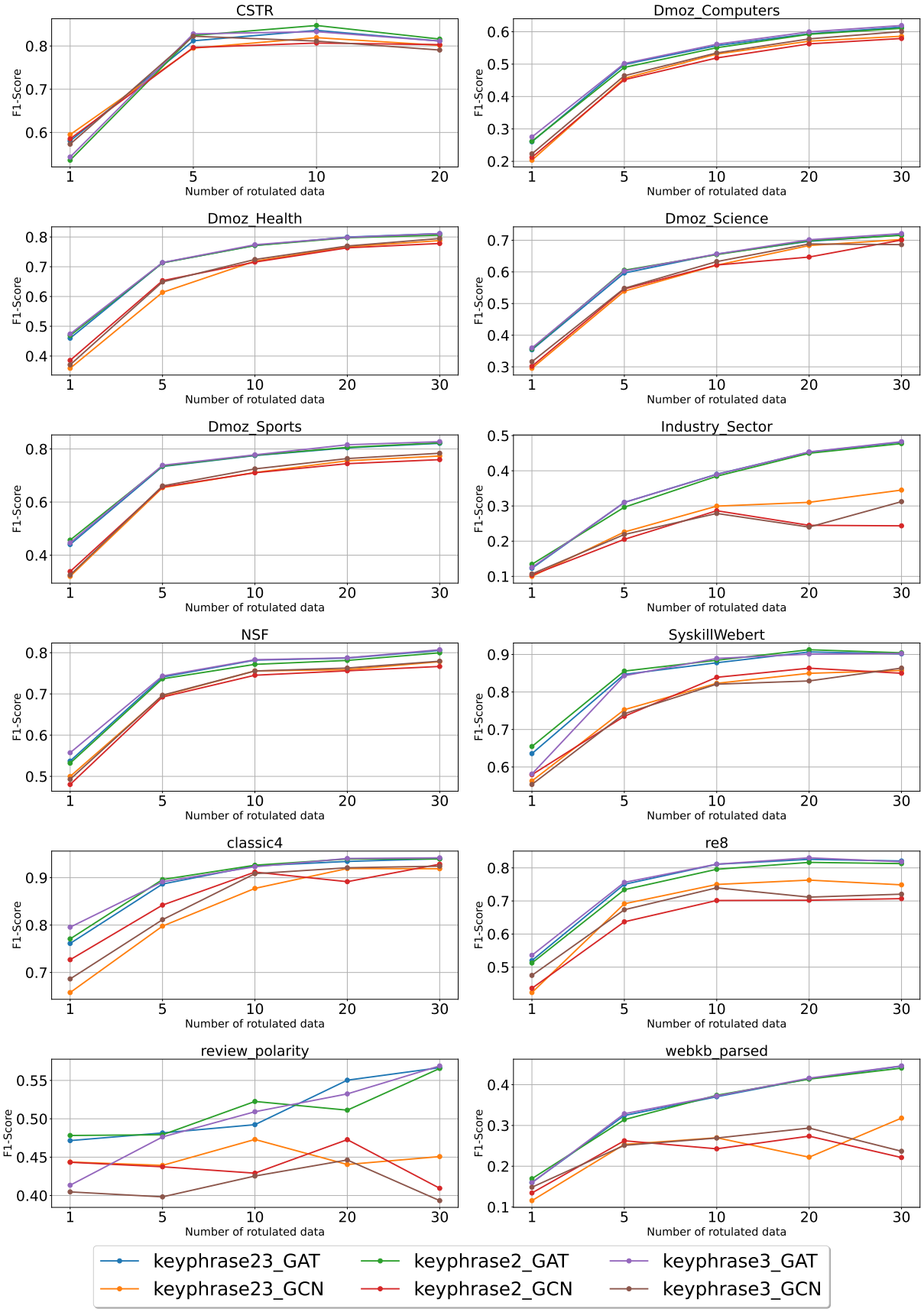


Figure 5.1: All GAT and GCN models F1-scores results, over the increase of labeled node numbers.

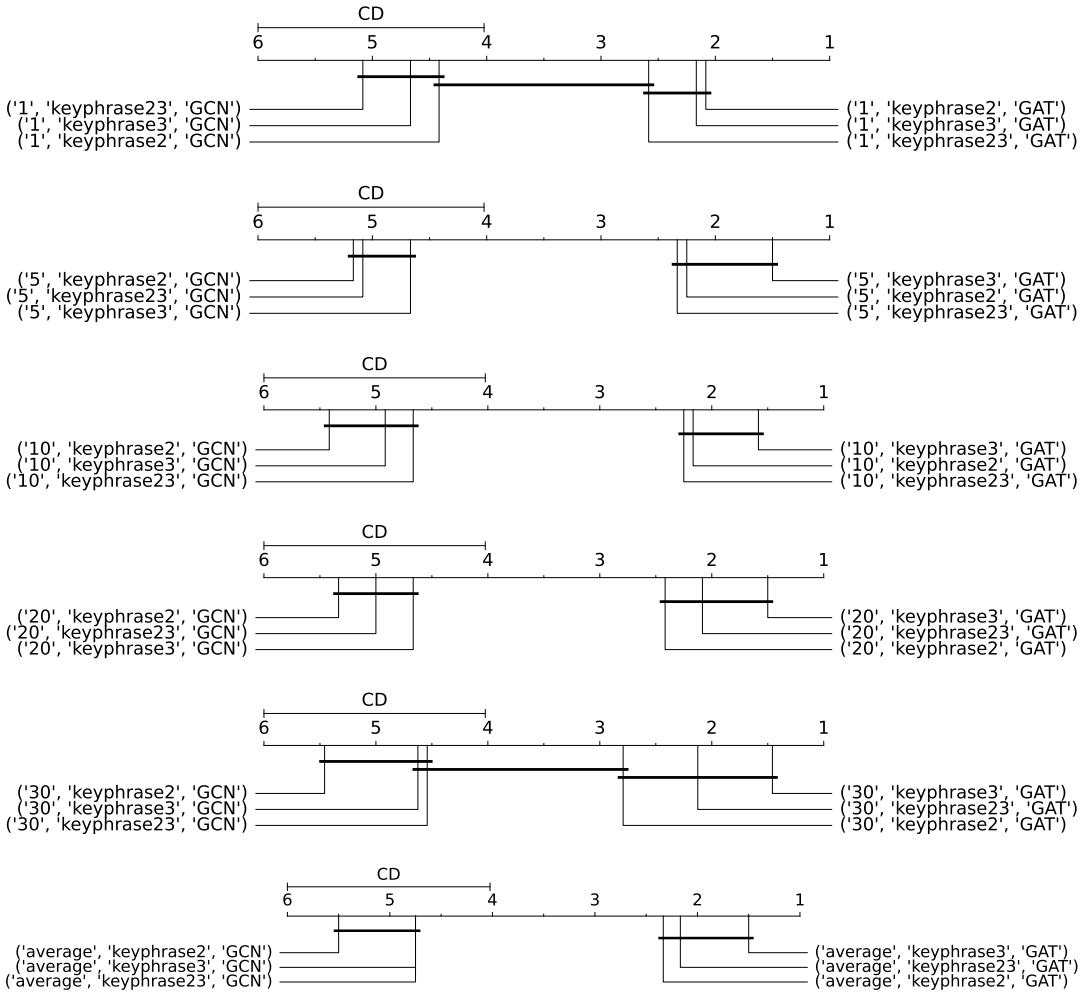


Figure 5.2: Traditional GAT and GCN critical difference plots.

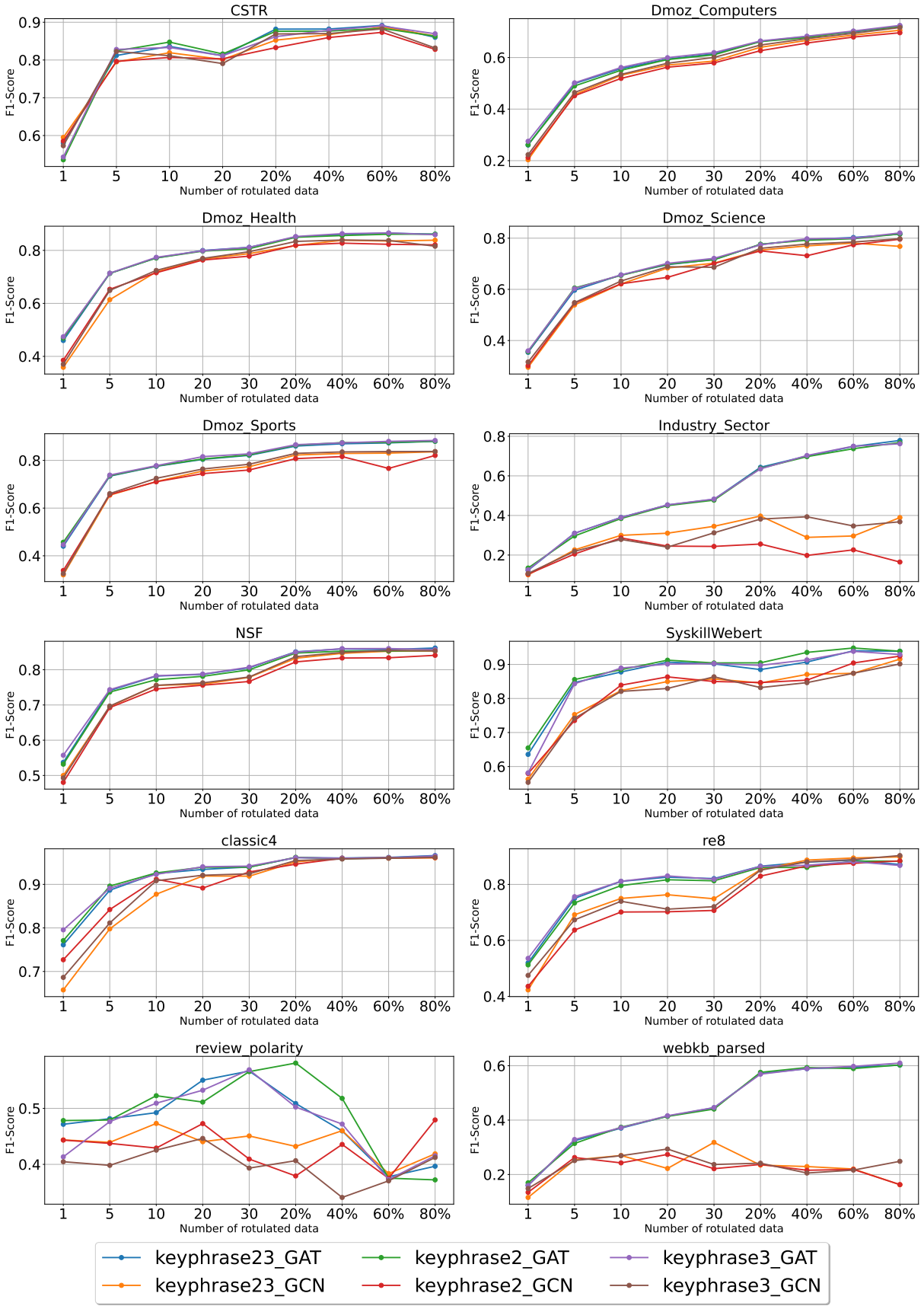


Figure 5.3: GAT and GCN models F1-scores results over the increase of labeled node numbers but also considering 20%, 40%, 60%, and 80% labeled instances.

Dataset	Keyphrases	GNN Model	Number of Labeled Data				Average
			20%	40%	60%	80%	Performance
CSTR	[2,3]	GAT	**0.8818 ± 0.016**	**0.882 ± 0.006**	**0.8917 ± 0.006**	0.8601 ± 0.014	**0.8789 ± 0.011**
CSTR	[2,3]	GCN	0.8525 ± 0.027	0.8672 ± 0.016	0.8874 ± 0.011	0.8637 ± 0.023	0.8677 ± 0.019
CSTR	[2]	GAT	0.876 ± 0.013	0.8766 ± 0.011	0.8838 ± 0.009	0.8632 ± 0.015	0.8749 ± 0.012
CSTR	[2]	GCN	0.8328 ± 0.033	0.8596 ± 0.029	0.8732 ± 0.022	0.8281 ± 0.055	0.8484 ± 0.035
CSTR	[3]	GAT	0.862 ± 0.03	0.8788 ± 0.007	0.8901 ± 0.006	**0.8695 ± 0.018**	0.8751 ± 0.015
CSTR	[3]	GCN	0.869 ± 0.025	0.87 ± 0.01	0.8824 ± 0.013	0.8319 ± 0.034	0.8633 ± 0.02
Dmoz_Computers	[2,3]	GAT	0.6627 ± 0.004	0.6815 ± 0.004	0.6991 ± 0.004	0.721 ± 0.003	0.6911 ± 0.004
Dmoz_Computers	[2,3]	GCN	0.6398 ± 0.005	0.6664 ± 0.005	0.6869 ± 0.004	0.7059 ± 0.004	0.6748 ± 0.004
Dmoz_Computers	[2]	GAT	0.6621 ± 0.003	0.6773 ± 0.004	0.7017 ± 0.006	0.7204 ± 0.004	0.6904 ± 0.004
Dmoz_Computers	[2]	GCN	0.6266 ± 0.006	0.6563 ± 0.007	0.6795 ± 0.007	0.6961 ± 0.008	0.6646 ± 0.007
Dmoz_Computers	[3]	GAT	**0.6644 ± 0.002**	**0.6828 ± 0.002**	**0.7029 ± 0.005**	**0.7247 ± 0.006**	**0.6937 ± 0.004**
Dmoz_Computers	[3]	GCN	0.6485 ± 0.004	0.6733 ± 0.006	0.6939 ± 0.002	0.7164 ± 0.007	0.683 ± 0.005
Dmoz_Health	[2,3]	GAT	0.8505 ± 0.003	0.8606 ± 0.003	0.8651 ± 0.003	0.861 ± 0.003	0.8593 ± 0.003
Dmoz_Health	[2,3]	GCN	0.8184 ± 0.012	0.8383 ± 0.005	0.8352 ± 0.015	0.8384 ± 0.006	0.8326 ± 0.009
Dmoz_Health	[2]	GAT	0.8494 ± 0.003	0.856 ± 0.003	0.861 ± 0.002	**0.8613 ± 0.004**	**0.8569 ± 0.003**
Dmoz_Health	[2]	GCN	0.8193 ± 0.004	0.8273 ± 0.003	0.8229 ± 0.011	0.8209 ± 0.013	0.8226 ± 0.008
Dmoz_Health	[3]	GAT	**0.8523 ± 0.002**	**0.8631 ± 0.002**	**0.8659 ± 0.002**	0.8591 ± 0.005	**0.8601 ± 0.003**
Dmoz_Health	[3]	GCN	0.8338 ± 0.004	0.8391 ± 0.005	0.8374 ± 0.007	0.8157 ± 0.064	0.8315 ± 0.02
Dmoz_Science	[2,3]	GAT	0.7754 ± 0.003	0.7949 ± 0.003	**0.8027 ± 0.003**	0.8158 ± 0.003	0.7972 ± 0.003
Dmoz_Science	[2,3]	GCN	0.7533 ± 0.004	0.7699 ± 0.007	0.7807 ± 0.006	0.768 ± 0.112	0.768 ± 0.032
Dmoz_Science	[2]	GAT	**0.7764 ± 0.005**	0.7919 ± 0.003	0.7978 ± 0.003	0.8166 ± 0.004	0.7957 ± 0.004
Dmoz_Science	[2]	GCN	0.7504 ± 0.004	0.7315 ± 0.117	0.7742 ± 0.003	0.796 ± 0.007	0.763 ± 0.033
Dmoz_Science	[3]	GAT	0.7739 ± 0.003	**0.7976 ± 0.002**	0.7998 ± 0.004	**0.8205 ± 0.003**	**0.798 ± 0.003**
Dmoz_Science	[3]	GCN	0.7604 ± 0.004	0.777 ± 0.005	0.785 ± 0.004	0.7979 ± 0.005	0.7801 ± 0.004
Dmoz_Sports	[2,3]	GAT	0.86 ± 0.002	0.8697 ± 0.001	0.8733 ± 0.001	0.88 ± 0.002	0.8708 ± 0.002
Dmoz_Sports	[2,3]	GCN	0.8224 ± 0.004	0.8291 ± 0.005	0.8305 ± 0.006	0.8361 ± 0.007	0.8295 ± 0.005
Dmoz_Sports	[2]	GAT	0.8636 ± 0.001	**0.8746 ± 0.002**	0.8752 ± 0.001	0.8804 ± 0.002	0.8735 ± 0.002
Dmoz_Sports	[2]	GCN	0.8073 ± 0.004	0.816 ± 0.006	0.7663 ± 0.146	0.8208 ± 0.008	0.8026 ± 0.041
Dmoz_Sports	[3]	GAT	**0.8658 ± 0.001**	0.8736 ± 0.001	**0.8798 ± 0.001**	**0.8836 ± 0.002**	**0.8757 ± 0.001**
Dmoz_Sports	[3]	GCN	0.8295 ± 0.003	0.8356 ± 0.005	0.8371 ± 0.004	0.8375 ± 0.008	0.8349 ± 0.005
Industry_Sector	[2,3]	GAT	**0.6433 ± 0.005**	0.7001 ± 0.006	0.7486 ± 0.02	**0.7789 ± 0.005**	**0.7177 ± 0.009**
Industry_Sector	[2,3]	GCN	0.397 ± 0.108	0.2893 ± 0.141	0.2964 ± 0.128	0.3894 ± 0.11	0.343 ± 0.122
Industry_Sector	[2]	GAT	0.6383 ± 0.007	0.6971 ± 0.007	0.7373 ± 0.007	0.7684 ± 0.011	0.7103 ± 0.008
Industry_Sector	[2]	GCN	0.2556 ± 0.132	0.1981 ± 0.146	0.2262 ± 0.106	0.1647 ± 0.112	0.2112 ± 0.124
Industry_Sector	[3]	GAT	0.6353 ± 0.005	**0.7029 ± 0.009**	**0.75 ± 0.011**	0.7621 ± 0.011	**0.7126 ± 0.009**
Industry_Sector	[3]	GCN	0.3817 ± 0.136	0.3933 ± 0.08	0.3469 ± 0.129	0.368 ± 0.132	0.3725 ± 0.119
NSF	[2,3]	GAT	**0.8509 ± 0.002**	0.859 ± 0.002	0.8571 ± 0.001	**0.8617 ± 0.004**	0.8572 ± 0.002
NSF	[2,3]	GCN	0.8325 ± 0.002	0.8466 ± 0.001	0.8525 ± 0.003	0.8534 ± 0.008	0.8462 ± 0.003
NSF	[2]	GAT	0.8473 ± 0.002	0.8527 ± 0.003	0.853 ± 0.002	0.8545 ± 0.004	0.8519 ± 0.003
NSF	[2]	GCN	0.8223 ± 0.003	0.8332 ± 0.004	0.8339 ± 0.008	0.8409 ± 0.004	0.8326 ± 0.004
NSF	[3]	GAT	0.8497 ± 0.002	**0.8598 ± 0.002**	**0.8601 ± 0.002**	0.8571 ± 0.004	**0.8567 ± 0.002**
NSF	[3]	GCN	0.8371 ± 0.003	0.8491 ± 0.003	0.8549 ± 0.005	0.8535 ± 0.008	0.8486 ± 0.005
SyskillWebert	[2,3]	GAT	0.8849 ± 0.014	0.9071 ± 0.012	0.9402 ± 0.009	0.9388 ± 0.009	0.9177 ± 0.011
SyskillWebert	[2,3]	GCN	0.8456 ± 0.013	0.8707 ± 0.02	0.874 ± 0.046	0.9157 ± 0.034	0.8765 ± 0.028
SyskillWebert	[2]	GAT	**0.9048 ± 0.011**	**0.9355 ± 0.011**	**0.9483 ± 0.008**	**0.9388 ± 0.015**	**0.9318 ± 0.011**
SyskillWebert	[2]	GCN	0.8468 ± 0.034	0.8539 ± 0.045	0.9043 ± 0.026	0.9245 ± 0.014	0.8824 ± 0.029
SyskillWebert	[3]	GAT	0.8974 ± 0.012	0.9134 ± 0.007	0.9385 ± 0.006	0.9293 ± 0.017	0.9196 ± 0.01
SyskillWebert	[3]	GCN	0.832 ± 0.021	0.8465 ± 0.038	0.8741 ± 0.026	0.9015 ± 0.038	0.8635 ± 0.031
classic4	[2,3]	GAT	0.9614 ± 0.002	**0.9605 ± 0.001**	**0.9619 ± 0.002**	**0.9662 ± 0.001**	**0.9625 ± 0.002**
classic4	[2,3]	GCN	0.9517 ± 0.004	0.9601 ± 0.004	0.9601 ± 0.003	0.9608 ± 0.007	0.9582 ± 0.004
classic4	[2]	GAT	**0.9622 ± 0.002**	0.9588 ± 0.002	0.9608 ± 0.002	0.9631 ± 0.004	0.9612 ± 0.002
classic4	[2]	GCN	0.9464 ± 0.006	0.9599 ± 0.003	0.961 ± 0.004	0.9616 ± 0.003	0.9572 ± 0.004
classic4	[3]	GAT	0.9614 ± 0.002	0.96 ± 0.003	0.9611 ± 0.001	0.9652 ± 0.002	**0.9619 ± 0.002**
classic4	[3]	GCN	0.9547 ± 0.002	0.9582 ± 0.003	0.9601 ± 0.003	0.9621 ± 0.005	0.9588 ± 0.003
re8	[2,3]	GAT	**0.865 ± 0.018**	0.8795 ± 0.011	0.8867 ± 0.008	0.8712 ± 0.014	0.8756 ± 0.013
re8	[2,3]	GCN	0.8529 ± 0.02	**0.8869 ± 0.013**	**0.8945 ± 0.009**	0.8983 ± 0.018	0.8831 ± 0.015
re8	[2]	GAT	0.8605 ± 0.009	0.8605 ± 0.03	0.883 ± 0.014	0.8832 ± 0.016	0.8718 ± 0.017
re8	[2]	GCN	0.8296 ± 0.017	0.8663 ± 0.007	0.8753 ± 0.009	0.884 ± 0.017	0.8638 ± 0.013
re8	[3]	GAT	0.8644 ± 0.016	0.8683 ± 0.037	0.8813 ± 0.008	0.8678 ± 0.018	**0.8704 ± 0.02**
re8	[3]	GCN	0.8509 ± 0.016	0.8804 ± 0.02	0.8887 ± 0.011	**0.9032 ± 0.013**	0.8808 ± 0.015
review_polarity	[2,3]	GAT	0.5087 ± 0.159	0.4598 ± 0.168	0.3779 ± 0.117	0.3967 ± 0.127	**0.4358 ± 0.143**
review_polarity	[2,3]	GCN	0.4321 ± 0.159	0.4601 ± 0.161	**0.3834 ± 0.098**	0.4188 ± 0.139	0.4236 ± 0.139
review_polarity	[2]	GAT	**0.5809 ± 0.146**	**0.5178 ± 0.168**	0.3753 ± 0.099	0.3723 ± 0.105	**0.4616 ± 0.129**
review_polarity	[2]	GCN	0.3793 ± 0.089	0.4356 ± 0.144	0.3751 ± 0.112	**0.4793 ± 0.154**	0.4173 ± 0.125
review_polarity	[3]	GAT	0.5023 ± 0.156	0.472 ± 0.176	0.3729 ± 0.124	0.4153 ± 0.133	0.4406 ± 0.147
review_polarity	[3]	GCN	0.4066 ± 0.126	0.3409 ± 0.016	0.3706 ± 0.111	0.4123 ± 0.119	0.3826 ± 0.093
webkb_parsed	[2,3]	GAT	0.5712 ± 0.014	0.5891 ± 0.011	0.5946 ± 0.008	0.6021 ± 0.009	0.5892 ± 0.01
webkb_parsed	[2,3]	GCN	0.2345 ± 0.064	0.2292 ± 0.049	0.2202 ± 0.061	0.1624 ± 0.056	0.2116 ± 0.058
webkb_parsed	[2]	GAT	**0.5756 ± 0.008**	**0.5933 ± 0.005**	0.5895 ± 0.011	0.603 ± 0.013	0.5904 ± 0.009
webkb_parsed	[2]	GCN	0.2369 ± 0.098	0.2156 ± 0.077	0.2186 ± 0.052	0.1629 ± 0.08	0.2085 ± 0.077
webkb_parsed	[3]	GAT	0.569 ± 0.015	0.5893 ± 0.011	**0.5977 ± 0.008**	**0.6098 ± 0.011**	**0.5914 ± 0.011**
webkb_parsed	[3]	GCN	0.2415 ± 0.074	0.2054 ± 0.061	0.2161 ± 0.058	0.2485 ± 0.043	0.2279 ± 0.059

Table 5.2: All GAT and GCN macro F1-Socre results using 20%, 40%, 60%, and 80% of labeled data. The numbers represent the average and standard deviation of all ten iterations for each dataset, separated by the number of labeled data, keyphrases, model, and dataset.

Dataset	Naive Bayes	Multinomial Naive Bayes	J48: Descision Tree	SMO: Support Vector Machine	IBk: k-Nearest Neighbors	GAT: (1-30)	GCN: (1-30)	GAT: (20%-80%)	GCN: (20%-80%)
CSTR	0.810	**0.870**	0.649	0.778	0.834	0.847	0.823	**0.892**	0.887
Dmoz_Computers	0.606	**0.701**	0.545	0.664	0.638	0.619	0.601	**0.725**	0.716
Dmoz_Health	0.732	**0.821**	0.736	0.807	0.779	0.811	0.795	**0.866**	0.839
Dmoz_Science	0.625	**0.740**	0.574	0.677	0.645	0.721	0.702	**0.821**	0.798
Dmoz_Sports	0.764	0.840	0.840	**0.858**	0.804	0.827	0.784	**0.884**	0.837
Industry_Sector	0.471	0.750	0.547	0.707	**0.871**	0.483	0.345	**0.779**	0.397
NSF	0.715	**0.835**	0.697	0.821	0.784	0.807	0.779	**0.862**	0.855
SyskillWebert	0.689	0.892	**0.960**	0.760	0.954	0.913	0.864	**0.948**	0.924
classic4	0.900	**0.962**	0.899	0.949	0.943	0.942	0.928	**0.966**	0.962
re8	0.679	**0.901**	0.819	0.824	0.888	0.830	0.763	0.887	**0.903**
review_polarity	0.669	0.801	0.683	**0.837**	0.705	0.569	0.473	**0.581**	0.479
webkb_parsed	0.386	0.557	0.557	0.523	**0.600**	0.446	0.318	**0.61**	0.249

Table 5.3: Listing of the best macro F1-score observed from the inductive models benchmarked at [Rossi et al., 2013] and the trained GAT and GCN models, aggregating models from 1 to 30 labeled data and 20% to 80% labeled data.

5.2 Coarsening

Considering the experiments conducted with coarsening, we performed a similar evaluation to subsection 5.1 but with an extra comparison analysis focused on coarsening characteristics. Table 5.4 represents the overall macro F1-Score evaluation of all models, Figure 5.4 shows the gradual evolution charts considering the progressive number of labeled instances ($n_{labeled} = [1, 5, 10, 20, 30]$), and Figure 5.6 represents the critical difference diagram for the coarsened graphs.

Here GAT also mostly achieved a higher F1-Score than GCN models, but the statistical difference between GAT and GCN decreased, as can be seen on the CD diagrams of Figure 5.6 and the CD groupings. Considering keyphrase sizes, keyphrase = (2,3) achieved constant better results with coarsened graphs, showing that flexible-sized keyphrases may facilitate coarsening node grouping.

Additionally, Figure 5.5 compares models trained on coarsened and non-coarsened graphs over the evolution of labeled instance numbers (i.e.,$n_{labeled}$). The green colors represent coarsened GAT keyphrases models, blue colors non-coarsened GAT keyphrases, yellow colors coarsened GCN keyphrases, and red colors represent non-coarsened GCN keyphrases. Here, we can more easily observe how coarsened graphs impact GAT and GCN compared to non-coarsened graphs. In most datasets, this impact is a minor decrease and, in some cases, a more significant one.

The results of Figure 5.5 help visualize how coarsening impacted GAT and GCN models' performance. But to further investigate this impact, Table 5.5 shows the combined average F1-score of coarsened and traditional models for each dataset and model. Also, we calculate the percentage difference between model scores: $(\frac{Coarsened\_F1}{Traditional\_F1} \times 100)$, that

can also be perceived as the *preservation score* of how much performance a coarsened model maintained when compared to its traditional counterpart.

We can observe that coarsening reduced graph performance, but this reduction impact varies from dataset to dataset, especially considering graph sizes and node distribution. Table 5.5 shows that the total average F1-Score percentage difference, or *preservation score*, was 82.85% for GAT and 84.3% for GCN, meaning that on average, coarsened GAT and GCN graphs performed 82.85% and 84.3% the capacity of a traditional graph. *NSF* presented the highest *preservation score* of 96.0% and 97.31% while *SyskillWebert* the lowest of 53.46% and 50.96% for GAT and GCN respectively. In no case do coarsened models outperform traditional models.

Figure 5.7 shows the critical difference diagram on all twelve datasets, averaging keyphrases performances and performing a statistical comparison between traditional and coarsened GAT and GCN models. The only model we can statistically affirm is more performative than the others is the traditional GAT. This analysis also shows no statistical difference between coarsening GAT and traditional GCN, meaning a coarsened GAT model can outperform non-coarsened GCN, as seen in cases of *NSF* (F1-score of 0.7010 for coarsened GAT and 0.6945 for traditional GCN) or *Industry_Sector* (F1-score of 0.2770 for coarsened GAT and 0.2348 for traditional GCN) on Table 5.5.

Lastly, Table 5.6 represents an overall graph size reduction analysis on the twelve datasets, showing the numbers of nodes and edges of coarsened and non-coarsened graphs and their reduction percentages. Table 5.6 last line shows the average reduction between all datasets, achieving a 50% average reduction on node numbers and a 44% on edge numbers. This means a possible halving of computational costs such as disk space or processing power.

Overall, considering the 50% and 44% average node and edge reduction of Table 5.6, the coarsening performance reduction presented in Table 5.5 can still be a viable alternative and a good trade-off for reduced computational costs. This is especially true when using larger and well-balanced datasets (with lower deviation on class percentages), such are the cases of the best performing coarsened models like the Dmoz_Collections, Classic4, NSF, and CSTR as these characteristics can be seen in Table 4.1.

Dataset	Keyphrases	GNN Model	Number of Labeled Data					Average
			1	5	10	20	30	Performance
CSTR	[2,3]	GAT	0.4493 ± 0.11	0.6589 ± 0.112	**0.6922 ± 0.1**	0.6419 ± 0.098	-	0.6106 ± 0.105
CSTR	[2,3]	GCN	0.4511 ± 0.153	0.6657 ± 0.107	0.6477 ± 0.116	0.6301 ± 0.095	-	0.5986 ± 0.118
CSTR	[2]	GAT	0.4137 ± 0.099	**0.7058 ± 0.062**	0.6556 ± 0.106	**0.6808 ± 0.064**	-	**0.614 ± 0.083**
CSTR	[2]	GCN	0.4208 ± 0.1	0.6463 ± 0.058	0.6582 ± 0.089	0.6141 ± 0.089	-	0.5849 ± 0.084
CSTR	[3]	GAT	**0.4894 ± 0.114**	0.6722 ± 0.083	0.604 ± 0.068	0.6274 ± 0.094	-	0.5982 ± 0.09
CSTR	[3]	GCN	0.4168 ± 0.151	0.6341 ± 0.101	0.611 ± 0.04	0.6018 ± 0.077	-	0.5659 ± 0.092
Dmoz_Computers	[2,3]	GAT	0.2341 ± 0.029	**0.4633 ± 0.009**	0.5156 ± 0.011	0.5458 ± 0.007	0.5658 ± 0.008	0.4649 ± 0.013
Dmoz_Computers	[2,3]	GCN	0.1974 ± 0.031	0.43 ± 0.026	0.5018 ± 0.012	0.5311 ± 0.01	0.5446 ± 0.01	0.441 ± 0.018
Dmoz_Computers	[2]	GAT	0.2218 ± 0.025	0.4368 ± 0.01	0.4825 ± 0.012	0.5169 ± 0.01	0.5346 ± 0.008	0.4385 ± 0.013
Dmoz_Computers	[2]	GCN	0.1792 ± 0.024	0.4031 ± 0.013	0.4599 ± 0.014	0.4958 ± 0.011	0.5139 ± 0.007	0.4104 ± 0.014
Dmoz_Computers	[3]	GAT	**0.2447 ± 0.035**	0.4619 ± 0.01	**0.5163 ± 0.015**	**0.5464 ± 0.009**	**0.5702 ± 0.009**	**0.4679 ± 0.016**
Dmoz_Computers	[3]	GCN	0.2005 ± 0.037	0.4427 ± 0.015	0.5085 ± 0.014	0.5374 ± 0.01	0.5542 ± 0.009	0.4487 ± 0.017
Dmoz_Health	[2,3]	GAT	0.4039 ± 0.048	0.6438 ± 0.02	**0.7066 ± 0.017**	0.726 ± 0.009	**0.7412 ± 0.011**	0.6443 ± 0.021
Dmoz_Health	[2,3]	GCN	0.3299 ± 0.068	0.599 ± 0.033	0.6734 ± 0.011	0.7063 ± 0.011	0.7187 ± 0.017	0.6055 ± 0.028
Dmoz_Health	[2]	GAT	0.3887 ± 0.045	0.6083 ± 0.014	0.6627 ± 0.017	0.6842 ± 0.015	0.6966 ± 0.009	0.6081 ± 0.02
Dmoz_Health	[2]	GCN	0.3247 ± 0.076	0.5635 ± 0.027	0.6368 ± 0.014	0.6675 ± 0.018	0.688 ± 0.006	0.5761 ± 0.028
Dmoz_Health	[3]	GAT	**0.435 ± 0.051**	**0.6476 ± 0.027**	0.7006 ± 0.014	**0.728 ± 0.012**	0.7396 ± 0.009	**0.6502 ± 0.023**
Dmoz_Health	[3]	GCN	0.3345 ± 0.084	0.6005 ± 0.027	0.6683 ± 0.017	0.7149 ± 0.016	0.7269 ± 0.007	0.609 ± 0.03
Dmoz_Science	[2,3]	GAT	0.3264 ± 0.032	**0.5606 ± 0.025**	0.6061 ± 0.014	0.6496 ± 0.012	0.6706 ± 0.01	**0.5627 ± 0.019**
Dmoz_Science	[2,3]	GCN	0.2404 ± 0.03	0.5168 ± 0.022	0.5769 ± 0.02	0.6383 ± 0.012	0.6573 ± 0.009	0.526 ± 0.018
Dmoz_Science	[2]	GAT	**0.3285 ± 0.036**	0.5487 ± 0.021	0.5859 ± 0.023	0.6314 ± 0.014	0.651 ± 0.012	0.5491 ± 0.021
Dmoz_Science	[2]	GCN	0.239 ± 0.041	0.5066 ± 0.018	0.5626 ± 0.011	0.6176 ± 0.014	0.6357 ± 0.009	0.5123 ± 0.019
Dmoz_Science	[3]	GAT	0.3133 ± 0.035	0.5579 ± 0.02	**0.6097 ± 0.018**	**0.6511 ± 0.015**	**0.6726 ± 0.008**	0.5609 ± 0.019
Dmoz_Science	[3]	GCN	0.2825 ± 0.053	0.5226 ± 0.04	0.5875 ± 0.021	0.6356 ± 0.015	0.6573 ± 0.008	0.5371 ± 0.027
Dmoz_Sports	[2,3]	GAT	0.3771 ± 0.033	**0.6572 ± 0.011**	**0.6943 ± 0.006**	**0.7224 ± 0.003**	**0.7391 ± 0.005**	**0.638 ± 0.012**
Dmoz_Sports	[2,3]	GCN	0.3143 ± 0.021	0.5995 ± 0.011	0.6636 ± 0.01	0.6929 ± 0.005	0.712 ± 0.004	0.5965 ± 0.011
Dmoz_Sports	[2]	GAT	0.3245 ± 0.022	0.5789 ± 0.01	0.6101 ± 0.014	0.6328 ± 0.009	0.6506 ± 0.004	0.5594 ± 0.012
Dmoz_Sports	[2]	GCN	0.2863 ± 0.038	0.5197 ± 0.022	0.5676 ± 0.013	0.5548 ± 0.13	0.6185 ± 0.011	0.5094 ± 0.043
Dmoz_Sports	[3]	GAT	**0.384 ± 0.033**	0.6504 ± 0.01	0.6864 ± 0.01	0.7141 ± 0.007	0.7312 ± 0.005	0.6332 ± 0.013
Dmoz_Sports	[3]	GCN	0.3325 ± 0.035	0.5954 ± 0.011	0.6483 ± 0.009	0.6862 ± 0.007	0.7036 ± 0.007	0.5932 ± 0.014
Industry_Sector	[2,3]	GAT	0.0735 ± 0.026	**0.219 ± 0.029**	**0.3102 ± 0.028**	0.396 ± 0.012	**0.4236 ± 0.013**	**0.2845 ± 0.022**
Industry_Sector	[2,3]	GCN	0.0589 ± 0.028	0.1501 ± 0.013	0.1953 ± 0.069	0.1803 ± 0.112	0.163 ± 0.121	0.1495 ± 0.069
Industry_Sector	[2]	GAT	0.0729 ± 0.012	0.1982 ± 0.036	0.2865 ± 0.025	0.3697 ± 0.035	0.3844 ± 0.045	0.2624 ± 0.031
Industry_Sector	[2]	GCN	0.0568 ± 0.01	0.1491 ± 0.03	0.2268 ± 0.032	0.1239 ± 0.082	0.1634 ± 0.114	0.144 ± 0.054
Industry_Sector	[3]	GAT	**0.0805 ± 0.026**	0.2155 ± 0.03	0.3087 ± 0.02	**0.3972 ± 0.017**	0.4195 ± 0.018	0.2843 ± 0.022
Industry_Sector	[3]	GCN	0.0637 ± 0.018	0.1605 ± 0.027	0.2036 ± 0.06	0.1654 ± 0.099	0.2259 ± 0.116	0.1638 ± 0.064
NSF	[2,3]	GAT	0.5356 ± 0.047	**0.7289 ± 0.018**	**0.7657 ± 0.008**	**0.7694 ± 0.013**	**0.7858 ± 0.008**	**0.7171 ± 0.019**
NSF	[2,3]	GCN	0.5164 ± 0.027	0.6977 ± 0.029	0.7486 ± 0.008	0.7487 ± 0.013	0.7667 ± 0.011	0.6956 ± 0.018
NSF	[2]	GAT	0.493 ± 0.056	0.6909 ± 0.016	0.7198 ± 0.011	0.7302 ± 0.009	0.7453 ± 0.012	0.6758 ± 0.021
NSF	[2]	GCN	0.4562 ± 0.047	0.657 ± 0.026	0.6935 ± 0.01	0.7078 ± 0.012	0.7221 ± 0.017	0.6473 ± 0.022
NSF	[3]	GAT	**0.5365 ± 0.067**	0.7207 ± 0.018	0.7504 ± 0.014	0.7643 ± 0.012	0.7788 ± 0.008	0.7102 ± 0.024
NSF	[3]	GCN	0.4796 ± 0.066	0.7015 ± 0.014	0.7348 ± 0.014	0.7448 ± 0.015	0.7616 ± 0.014	0.6844 ± 0.025
SyskillWebert	[2,3]	GAT	**0.4249 ± 0.042**	0.4921 ± 0.027	0.4916 ± 0.03	0.4734 ± 0.076	0.3643 ± 0.105	**0.4493 ± 0.056**
SyskillWebert	[2,3]	GCN	0.3838 ± 0.06	0.4835 ± 0.05	0.4556 ± 0.029	0.3853 ± 0.077	0.269 ± 0.024	0.3954 ± 0.048
SyskillWebert	[2]	GAT	0.4024 ± 0.057	**0.507 ± 0.027**	0.4936 ± 0.021	0.4386 ± 0.033	**0.3673 ± 0.08**	0.4418 ± 0.044
SyskillWebert	[2]	GCN	0.372 ± 0.046	0.4565 ± 0.023	0.4535 ± 0.027	0.386 ± 0.019	0.2772 ± 0.071	0.389 ± 0.037
SyskillWebert	[3]	GAT	0.3953 ± 0.052	0.4952 ± 0.046	**0.5156 ± 0.052**	**0.4843 ± 0.086**	0.3358 ± 0.038	0.4453 ± 0.055
SyskillWebert	[3]	GCN	0.4009 ± 0.058	0.464 ± 0.029	0.4526 ± 0.017	0.3674 ± 0.036	0.2651 ± 0.017	0.39 ± 0.031
classic4	[2,3]	GAT	**0.742 ± 0.105**	0.8126 ± 0.075	**0.8732 ± 0.04**	0.8541 ± 0.073	0.8678 ± 0.062	**0.8299 ± 0.071**
classic4	[2,3]	GCN	0.5689 ± 0.111	0.7552 ± 0.117	0.8646 ± 0.031	0.8785 ± 0.026	**0.8733 ± 0.028**	0.7881 ± 0.063
classic4	[2]	GAT	0.6417 ± 0.061	0.7239 ± 0.097	0.7607 ± 0.063	0.8078 ± 0.048	0.776 ± 0.091	0.742 ± 0.072
classic4	[2]	GCN	0.4968 ± 0.168	0.708 ± 0.078	0.8223 ± 0.057	0.8312 ± 0.062	0.7929 ± 0.069	0.7302 ± 0.087
classic4	[3]	GAT	0.7046 ± 0.087	**0.8421 ± 0.055**	0.8441 ± 0.061	0.8616 ± 0.055	0.8389 ± 0.058	0.8183 ± 0.063
classic4	[3]	GCN	0.5687 ± 0.168	0.7868 ± 0.076	0.8564 ± 0.038	**0.879 ± 0.033**	0.8667 ± 0.02	0.7915 ± 0.067
re8	[2,3]	GAT	0.5406 ± 0.087	**0.748 ± 0.038**	0.7894 ± 0.032	**0.8105 ± 0.023**	0.7922 ± 0.049	**0.7361 ± 0.046**
re8	[2,3]	GCN	0.4177 ± 0.086	0.6813 ± 0.039	0.7055 ± 0.05	0.7514 ± 0.064	0.7261 ± 0.053	0.6564 ± 0.058
re8	[2]	GAT	0.4802 ± 0.084	0.6619 ± 0.037	0.7145 ± 0.047	0.7561 ± 0.026	0.722 ± 0.028	0.667 ± 0.044
re8	[2]	GCN	0.3321 ± 0.083	0.5651 ± 0.072	0.6213 ± 0.05	0.6401 ± 0.058	0.6756 ± 0.057	0.5668 ± 0.064
re8	[3]	GAT	**0.5422 ± 0.077**	0.7419 ± 0.029	**0.7929 ± 0.032**	0.8035 ± 0.023	**0.7937 ± 0.037**	0.7348 ± 0.04
re8	[3]	GCN	0.4558 ± 0.087	0.6544 ± 0.056	0.7087 ± 0.051	0.734 ± 0.074	0.7266 ± 0.062	0.6559 ± 0.066
review_polarity	[2,3]	GAT	0.3476 ± 0.024	0.3525 ± 0.057	0.3332 ± 0.003	0.3556 ± 0.05	0.3805 ± 0.057	0.3539 ± 0.038
review_polarity	[2,3]	GCN	0.3484 ± 0.029	**0.3933 ± 0.069**	0.3699 ± 0.054	0.3603 ± 0.041	**0.4188 ± 0.089**	**0.3781 ± 0.056**
review_polarity	[2]	GAT	0.3615 ± 0.06	0.3453 ± 0.022	0.3352 ± 0.005	0.3321 ± 0.004	0.3503 ± 0.022	0.3449 ± 0.022
review_polarity	[2]	GCN	**0.3733 ± 0.059**	0.3757 ± 0.052	**0.3748 ± 0.071**	0.3679 ± 0.048	0.3867 ± 0.074	0.3757 ± 0.061
review_polarity	[3]	GAT	0.3331 ± 0.0	0.34 ± 0.012	0.334 ± 0.004	0.3333 ± 0.005	0.3706 ± 0.05	0.3422 ± 0.014
review_polarity	[3]	GCN	0.3331 ± 0.0	0.3596 ± 0.034	0.3504 ± 0.021	**0.3787 ± 0.063**	0.3778 ± 0.059	0.3599 ± 0.035
webkb_parsed	[2,3]	GAT	**0.107 ± 0.038**	0.2338 ± 0.034	0.2466 ± 0.027	**0.2889 ± 0.032**	**0.3008 ± 0.028**	**0.2354 ± 0.032**
webkb_parsed	[2,3]	GCN	0.0924 ± 0.041	0.1867 ± 0.039	0.2055 ± 0.045	0.2015 ± 0.06	0.2153 ± 0.055	0.1803 ± 0.048
webkb_parsed	[2]	GAT	0.1047 ± 0.035	0.2317 ± 0.026	**0.2588 ± 0.032**	0.2836 ± 0.028	0.2912 ± 0.012	0.234 ± 0.027
webkb_parsed	[2]	GCN	0.0829 ± 0.026	0.1859 ± 0.037	0.2301 ± 0.025	0.1687 ± 0.074	0.2613 ± 0.052	0.1858 ± 0.043
webkb_parsed	[3]	GAT	0.1063 ± 0.039	**0.2358 ± 0.036**	0.2496 ± 0.032	0.2828 ± 0.03	0.2974 ± 0.03	0.2344 ± 0.034
webkb_parsed	[3]	GCN	0.0778 ± 0.039	0.1959 ± 0.039	0.1823 ± 0.043	0.2166 ± 0.073	0.1904 ± 0.071	0.1726 ± 0.053

Table 5.4: All coarsened GAT and GCN macro F1-Socre results. The numbers represent the average and standard deviation of all ten iterations for each dataset, separated by the number of labeled data, keyphrases, model, and dataset.

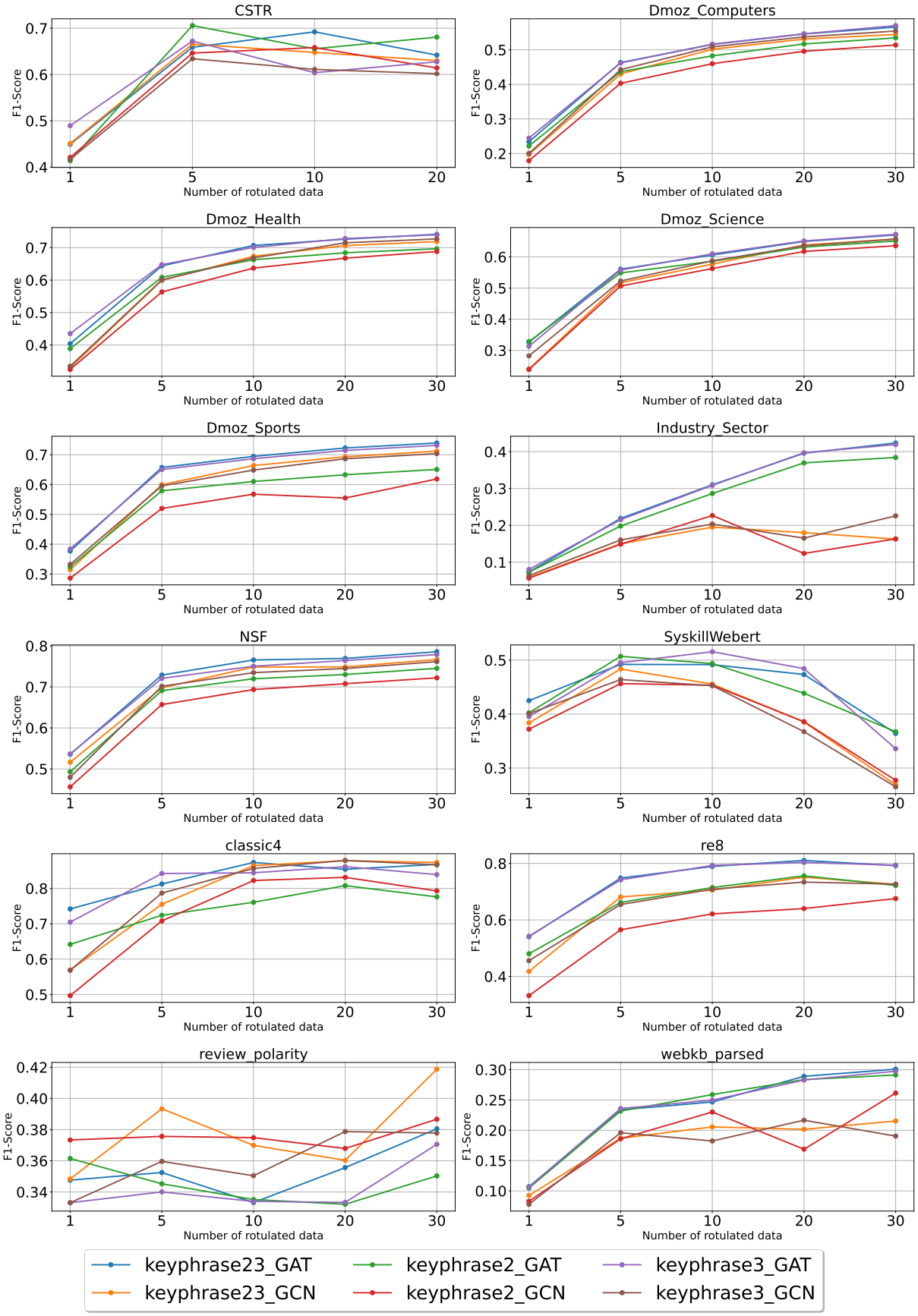


Figure 5.4: All coarsened GAT and GCN models F1-scores results, over the increase of labeled node numbers.

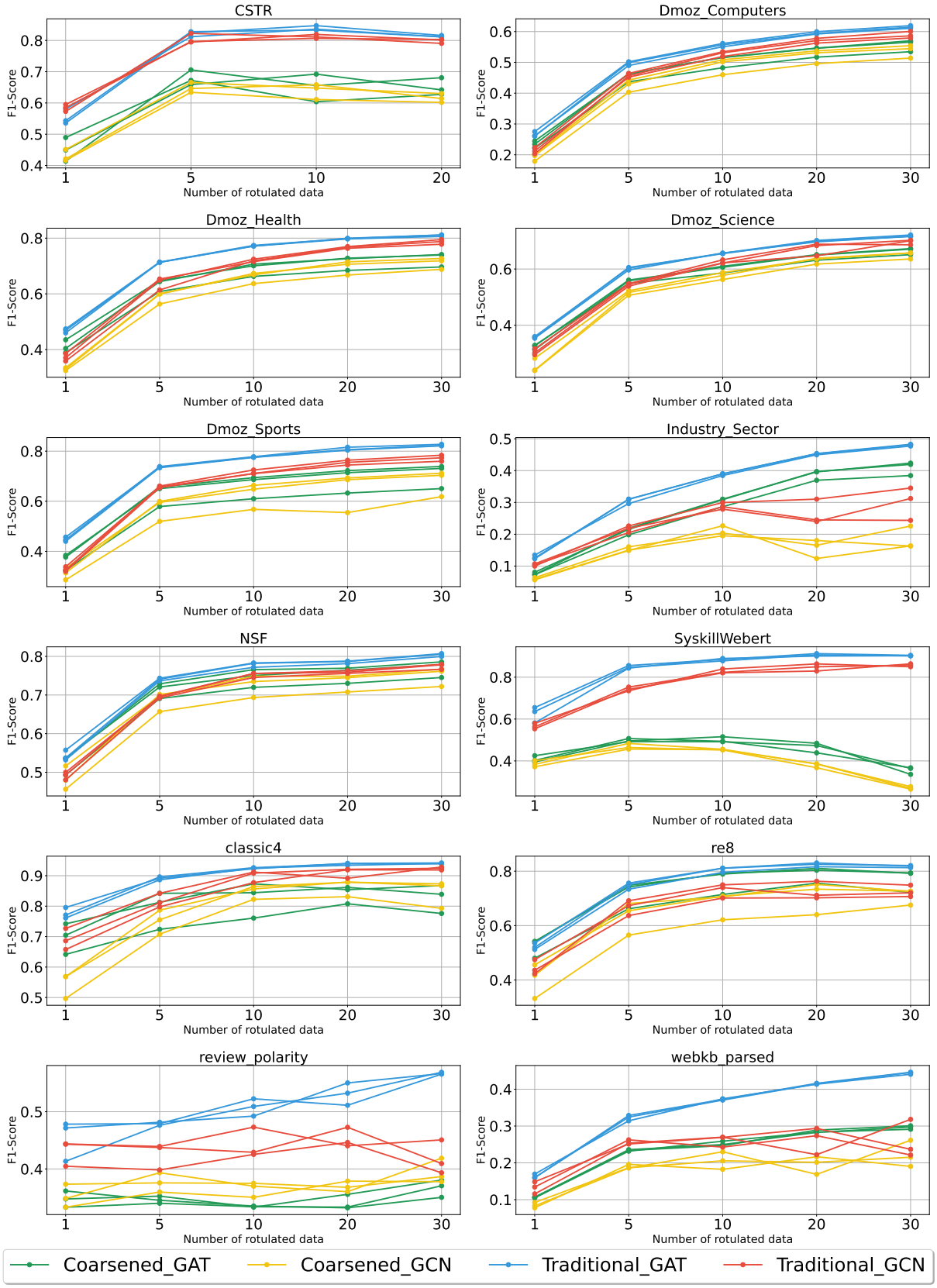


Figure 5.5: Comparative evolutionary graph between all coarsened GAT and GCN models and traditional GAT and GCN, previously presented in sub-section 5.1, F1-scores.

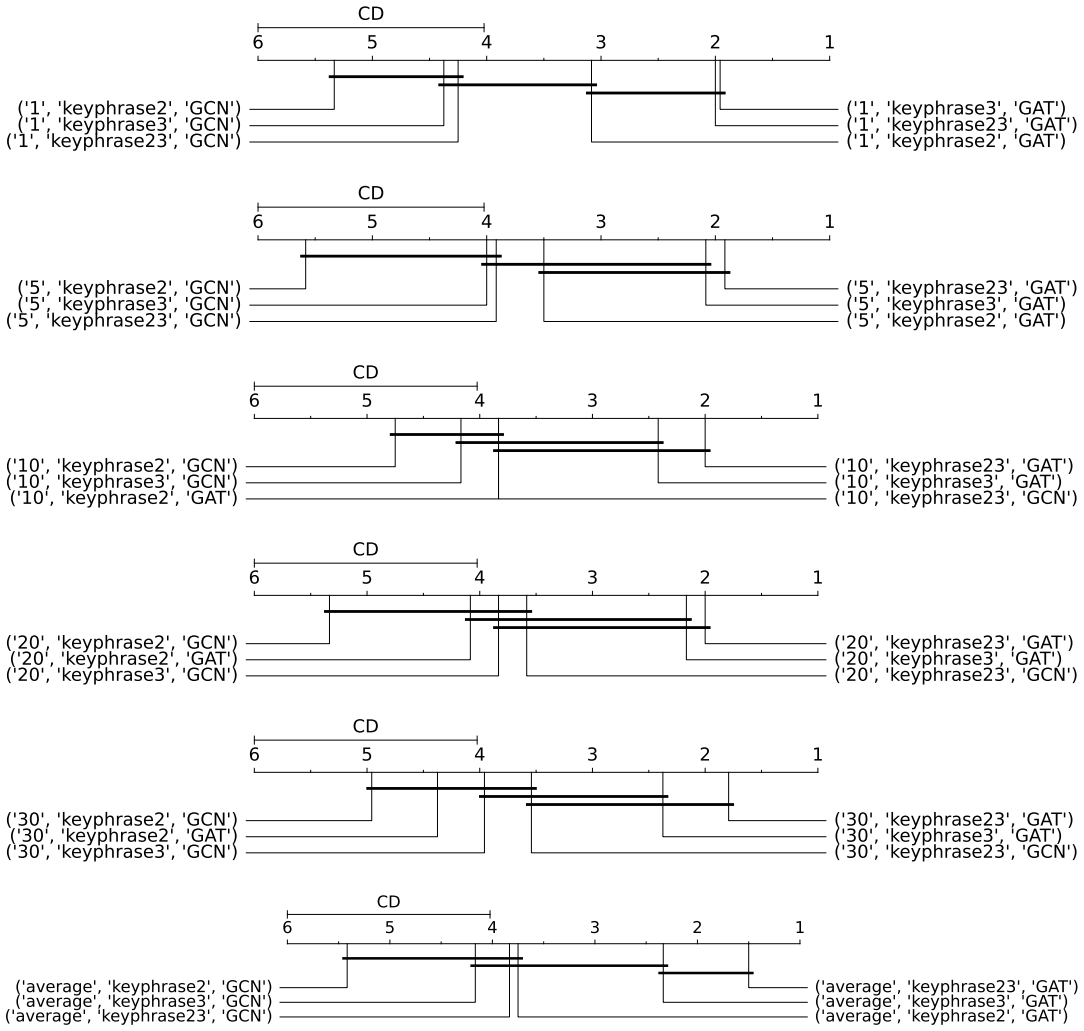


Figure 5.6: Coarsening critical difference plots.

Dataset	GAT Traditional	GCN Traditional	GAT Coarsening	GCN Coarsening	GAT F1-score preservation	GCN F1-score preservation
CSTR	0.7566	0.7500	0.6076	0.5831	80.31%	77.75%
Dmoz_Computers	0.5058	0.4715	0.4571	0.4333	90.37%	91.9%
Dmoz_Health	0.7123	0.6567	0.6342	0.5969	89.04%	90.89%
Dmoz_Science	0.6062	0.5686	0.5576	0.5251	91.98%	92.35%
Dmoz_Sports	0.7186	0.6455	0.6102	0.5664	84.92%	87.75%
Industry_Sector	0.3507	0.2348	0.2770	**0.1524**	78.98%	64.91%
NSF	0.7302	0.6945	0.7010	0.6758	**96.0%**	**97.31%**
SyskillWebert	0.8332	0.7682	0.4454	0.3915	**53.46%**	**50.96%**
classic4	**0.8941**	**0.8482**	**0.7967**	**0.7699**	89.11%	90.77%
re8	0.7434	0.6587	0.7126	0.6264	95.86%	95.1%
review_polarity	0.5080	0.4338	0.3470	0.3712	68.31%	85.57%
webkb_parsed	**0.3433**	**0.2342**	**0.2346**	0.1796	68.34%	76.69%
Total Average	**0.6419**	**0.5804**	**0.5318**	**0.4893**	**82.85%**	**84.3%**

Table 5.5: Comparison of average F1-score metric across datasets between coarsened and non-coarsened graph models, and coarsening F1-score performance preservation.

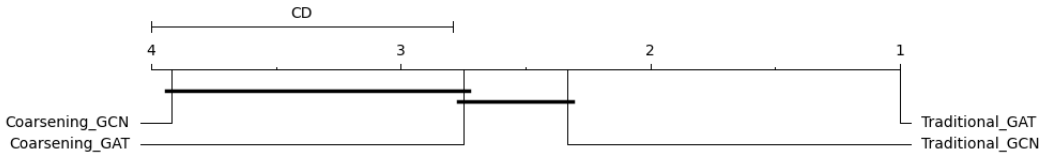


Figure 5.7: Critical difference diagram of traditional GAT, traditional GCN, coarsened GAT, and coarsened GCN models.

Dataset	Keyphrases	Total Nodes	Coarsened Nodes	Node Reduction	Total Edges	Coarsened Edges	Edge Reduction
classic4	[2]	32138	16068	49%	373512	123585	33%
classic4	[3]	36591	18295	49%	373447	153769	41%
classic4	[2,3]	37305	18652	49%	372682	159707	42%
CSTR	[2]	1613	806	49%	16517	4395	26%
CSTR	[3]	1721	860	49%	16666	4393	26%
CSTR	[2,3]	1691	845	49%	16529	4318	26%
Dmoz_Computers	[2]	48455	25130	51%	217578	149053	68%
Dmoz_Computers	[3]	54804	28766	52%	211293	144506	68%
Dmoz_Computers	[2,3]	54778	28629	52%	206577	136289	65%
Dmoz_Health	[2]	31612	16135	51%	160990	99999	62%
Dmoz_Health	[3]	36311	18762	51%	157460	100421	63%
Dmoz_Health	[2,3]	36149	18629	51%	154562	96546	62%
Dmoz_Science	[2]	32494	16896	52%	126931	84363	66%
Dmoz_Science	[3]	34537	17842	51%	123185	80207	65%
Dmoz_Science	[2,3]	34975	18055	51%	121028	76613	63%
Dmoz_Sports	[2]	57318	29476	51%	328103	213502	65%
Dmoz_Sports	[3]	71502	37921	53%	320034	226139	70%
Dmoz_Sports	[2,3]	71052	37552	52%	311959	212856	68%
Industry_Sector	[2]	40149	20074	49%	1079927	420075	38%
Industry_Sector	[3]	44801	22400	49%	1085381	599032	55%
Industry_Sector	[2,3]	44570	22284	49%	1075787	598776	55%
NSF	[2]	45145	23141	51%	173949	99104	56%
NSF	[3]	48894	24663	50%	164245	89000	54%
NSF	[2,3]	55310	28503	51%	161953	87904	54%
re8	[2]	27690	13937	50%	131113	51894	39%
re8	[3]	33430	17515	52%	128337	58811	45%
re8	[2,3]	33400	17340	51%	126986	56619	44%
review_polarity	[2]	10923	5461	49%	752196	171136	22%
review_polarity	[3]	11828	5914	50%	755651	173314	22%
review_polarity	[2,3]	11655	5827	49%	750585	172653	23%
SyskillWebert	[2]	1836	917	49%	32408	4705	14%
SyskillWebert	[3]	1889	944	49%	32699	4738	14%
SyskillWebert	[2,3]	1892	945	49%	32650	4854	14%
webkb_parsed	[2]	38651	19325	49%	1463927	230282	15%
webkb_parsed	[3]	45029	22514	49%	1495506	233328	15%
webkb_parsed	[2,3]	44441	22220	49%	1486632	240005	16%
Average		**33793**	**17312**	**50%**	**403861**	**149080**	**44%**

Table 5.6: Graph reductions in number and percentage across all datasets and keyphrases, by nodes and edges.

5.3 Large Language Models

Concluding the first step on LLM application heavily discussed on 4.6, which evaluates zero-shot and few-shot labeling prompts, by analyzing the results on Table 4.2 and ultimately selecting the list of the most accurate prompts for each text collection, we proceed to the process of labeling all proposed volumes of LLM data.

We employ Llamma to annotate three different volumes of LLM-labeled instances ($LLMn_{labeled}$): 10, 50, and 100, as discussed in section 4.6. Each of these volumes was used to train GAT and GCN models. Also, we combined each volume with 1, 5, 10, 20, and 30 human-labeled instances ($n_{labeled}$), creating LLM+human datasets that were also submitted for training. Similarly to the previous sections 5.1 and 5.2, we evaluated how

LLM-only data and the combination of LLM+human data, would affect models as the number of human-labeled instances increased.

Table 5.7 presents the overall macro F1-score, and Figure 5.8 presents the gradual evolution charts, iterating over the number of labeled documents (LLM_only,1,5,10,20,30) for $LLMn_{labeled} = 10$. Table 5.8 and Figure 5.9 present the same results for $LLMn_{labeled} = 50$, and Table 5.9 and Figure 5.10 for $LLMn_{labeled} = 100$. Figures 5.14, 5.15, and 5.16 represent the critical difference (CD) diagrams for all 10, 50, and 100 LLM-labeled models. Again, GAT models consistently achieved a higher F1-Score than GCN models, this time with a significant statistical difference (i.e., CD). Similarly to traditional models and opposing coarsening models, keyphrase = 3 achieved the best scores across keyphrase sizes.

Figure 5.11, Figure 5.12, and Figure 5.13 display the comparison between models trained on with LLM-labeled data and the traditional models trained only on human-labeled data. Each figure represents a comparison using $LLMn_{labeled} = [10, 50, 100]$, respectively, and grows by adding human-labeled instances to the models. By analyzing the $LLMn_{labeled}$ combinations, we observed a higher impact on the first iteration cases with only LLM-labeled data and combining LLM-labeled data to $n_{labeled} = 1$ human-labeled document. In these cases, LLM labeling consistently improved F1-Score metrics across all datasets by adding more training data. This was especially true with $LLMn_{labeled} = 50$ and $LLMn_{labeled} = 100$ whose results reached the top F1-scores.

However, when we add a few more instances of human-labeled documents, we notice LLM-labeling improvements decreasing. With $n_{labeled} = 5$ human-labeled documents for each class, LLM-labeled + human-labeled models were already outperformed by only human-labeled models, with only the datasets *Dmoz_Sports*, *Dmoz_Science* and *Industry_Sector* maintaining consistent performance gains. With $n_{labeled} = 10$ or more human-labeled documents, this situation aggravated, and traditional human-labeled models presented themselves as a better match for transductive graph learning on most datasets.

Regarding datasets, the main outliers were *Dmoz_sports* where LLM-only and LLM + human labeling achieved results remarkably close to only human-labeled data. This occurs due to the simplicity of the classification task on this dataset, *Sports* are clear and well-defined classes that facilitate the labeling of the Llama model as can be observed by its accuracy of 0.85 in Table 4.2. *SyskillWebert* achieved an inferior performance, possibly because of the dataset's lower size. *Review_polarity*, our only binary class dataset, also presented its standard, irregular behavior, but LMM+human GATs managed to regularly outperform traditional models besides the dataset's irregular behaviors.

In short, LLM-labeling impacts diminished as more human-labeled data was added. Also, LLM-labeled data was somewhat detrimental compared to traditionally human-

labeled data on the transductive classification task. These facts indicate that LLMs are not an excellent fit for transductive learning, where few very accurate data are more beneficial than a high quantity of moderate precision data. the use of generalists LLMs for transductive learning would be recommended only for semantically simpler classification tasks, like *Dmoz_Sports*, and in the cases of advanced, harder-to-interpret, or non-common textual structures such as websites, it would be recommended a fine-tuned LLM model or investing in accurate human labeling.

Dataset	Keyphrases	GNN Model	Number of Labeled Data						Average
			LLM Only	1	5	10	20	30	Performance
CSTR	[2,3]	GAT	0.7575 ± 0.07	0.7585 ± 0.091	0.7986 ± 0.03	0.7696 ± 0.035	0.7477 ± 0.041	-	0.7664 ± 0.053
CSTR	[2,3]	GCN	0.7428 ± 0.062	0.7485 ± 0.068	0.8073 ± 0.035	0.7347 ± 0.04	0.7212 ± 0.062	-	0.7509 ± 0.053
CSTR	[2]	GAT	**0.7625 ± 0.052**	**0.7711 ± 0.054**	0.7874 ± 0.043	**0.7856 ± 0.03**	0.7446 ± 0.049	-	**0.7702 ± 0.046**
CSTR	[2]	GCN	0.7429 ± 0.067	0.7446 ± 0.066	0.7742 ± 0.036	0.7595 ± 0.048	0.7099 ± 0.065	-	0.7462 ± 0.056
CSTR	[3]	GAT	0.7545 ± 0.063	0.7675 ± 0.056	**0.8075 ± 0.035**	0.7726 ± 0.035	**0.7478 ± 0.043**	-	0.77 ± 0.047
CSTR	[3]	GCN	0.7469 ± 0.09	0.7256 ± 0.063	0.7914 ± 0.044	0.7509 ± 0.061	0.7359 ± 0.062	-	0.7501 ± 0.064
Dmoz_Computers	[2,3]	GAT	0.4087 ± 0.015	0.4237 ± 0.019	0.4615 ± 0.008	0.5147 ± 0.022	0.5577 ± 0.013	0.5791 ± 0.012	0.4909 ± 0.015
Dmoz_Computers	[2,3]	GCN	0.3896 ± 0.015	0.4022 ± 0.018	0.4493 ± 0.012	0.4953 ± 0.023	0.537 ± 0.015	0.5548 ± 0.012	0.4714 ± 0.016
Dmoz_Computers	[2]	GAT	0.4068 ± 0.01	0.4174 ± 0.017	0.464 ± 0.01	0.5069 ± 0.018	0.5551 ± 0.011	0.5786 ± 0.012	0.4881 ± 0.013
Dmoz_Computers	[2]	GCN	0.3771 ± 0.017	0.3959 ± 0.014	0.4445 ± 0.013	0.4822 ± 0.019	0.5245 ± 0.014	0.5463 ± 0.01	0.4617 ± 0.014
Dmoz_Computers	[3]	GAT	**0.4149 ± 0.014**	**0.4271 ± 0.018**	**0.4697 ± 0.01**	**0.5176 ± 0.024**	**0.5637 ± 0.01**	**0.5869 ± 0.015**	**0.4966 ± 0.015**
Dmoz_Computers	[3]	GCN	0.3989 ± 0.019	0.4106 ± 0.01	0.4562 ± 0.008	0.5018 ± 0.026	0.5426 ± 0.014	0.5623 ± 0.014	0.4787 ± 0.015
Dmoz_Health	[2,3]	GAT	0.6744 ± 0.024	0.6791 ± 0.02	0.7154 ± 0.023	0.7402 ± 0.015	0.7768 ± 0.014	**0.7933 ± 0.008**	0.7299 ± 0.017
Dmoz_Health	[2,3]	GCN	0.6482 ± 0.028	0.6522 ± 0.026	0.6875 ± 0.021	0.7215 ± 0.018	0.7553 ± 0.014	0.7756 ± 0.009	0.7067 ± 0.019
Dmoz_Health	[2]	GAT	0.6742 ± 0.024	**0.6827 ± 0.024**	0.7153 ± 0.019	0.7445 ± 0.019	0.7766 ± 0.013	0.7882 ± 0.007	0.7303 ± 0.018
Dmoz_Health	[2]	GCN	0.6378 ± 0.022	0.6546 ± 0.03	0.679 ± 0.021	0.7121 ± 0.012	0.7454 ± 0.017	0.7614 ± 0.008	0.6984 ± 0.018
Dmoz_Health	[3]	GAT	**0.6774 ± 0.023**	0.6824 ± 0.023	**0.719 ± 0.024**	**0.7465 ± 0.017**	**0.7788 ± 0.014**	0.7927 ± 0.009	**0.7328 ± 0.018**
Dmoz_Health	[3]	GCN	0.6503 ± 0.028	0.6297 ± 0.104	0.6871 ± 0.022	0.733 ± 0.017	0.7547 ± 0.012	0.7739 ± 0.011	0.7048 ± 0.032
Dmoz_Science	[2,3]	GAT	0.5819 ± 0.033	0.5938 ± 0.024	0.6139 ± 0.034	0.6579 ± 0.015	0.6841 ± 0.01	0.7038 ± 0.004	0.6392 ± 0.02
Dmoz_Science	[2,3]	GCN	0.5684 ± 0.033	0.5738 ± 0.028	0.5996 ± 0.028	0.6451 ± 0.018	0.6712 ± 0.015	0.6956 ± 0.007	0.6256 ± 0.021
Dmoz_Science	[2]	GAT	0.5851 ± 0.031	0.6 ± 0.027	0.6153 ± 0.025	0.6606 ± 0.015	0.6839 ± 0.011	0.7085 ± 0.005	0.6422 ± 0.019
Dmoz_Science	[2]	GCN	0.5556 ± 0.032	0.5763 ± 0.028	0.6006 ± 0.025	0.6462 ± 0.014	0.6708 ± 0.009	0.6869 ± 0.01	0.6227 ± 0.02
Dmoz_Science	[3]	GAT	**0.5893 ± 0.034**	**0.6051 ± 0.024**	**0.6157 ± 0.026**	**0.6614 ± 0.017**	**0.6871 ± 0.01**	**0.7102 ± 0.005**	**0.6448 ± 0.019**
Dmoz_Science	[3]	GCN	0.5616 ± 0.037	0.5672 ± 0.024	0.6069 ± 0.028	0.6506 ± 0.017	0.6746 ± 0.008	0.7035 ± 0.008	0.6274 ± 0.02
Dmoz_Sports	[2,3]	GAT	0.73 ± 0.019	0.7383 ± 0.019	0.7558 ± 0.008	0.7781 ± 0.007	0.7985 ± 0.007	0.815 ± 0.006	0.7693 ± 0.011
Dmoz_Sports	[2,3]	GCN	0.6751 ± 0.013	0.6902 ± 0.011	0.7144 ± 0.007	0.7324 ± 0.007	0.7551 ± 0.006	0.769 ± 0.006	0.7227 ± 0.008
Dmoz_Sports	[2]	GAT	**0.733 ± 0.02**	0.74 ± 0.019	0.7623 ± 0.01	0.7827 ± 0.01	0.8021 ± 0.006	0.8168 ± 0.005	0.7728 ± 0.012
Dmoz_Sports	[2]	GCN	0.6636 ± 0.015	0.6749 ± 0.014	0.6944 ± 0.009	0.7118 ± 0.005	0.741 ± 0.003	0.7546 ± 0.008	0.7067 ± 0.009
Dmoz_Sports	[3]	GAT	0.7325 ± 0.019	**0.7416 ± 0.017**	**0.7655 ± 0.007**	**0.7846 ± 0.011**	**0.8066 ± 0.007**	**0.8207 ± 0.005**	**0.7753 ± 0.011**
Dmoz_Sports	[3]	GCN	0.6864 ± 0.015	0.6894 ± 0.014	0.7162 ± 0.008	0.7366 ± 0.011	0.7663 ± 0.008	0.7768 ± 0.006	0.7286 ± 0.01
Industry_Sector	[2,3]	GAT	0.2723 ± 0.021	0.292 ± 0.02	0.3405 ± 0.012	0.3761 ± 0.017	0.4296 ± 0.017	**0.4511 ± 0.013**	0.3603 ± 0.017
Industry_Sector	[2,3]	GCN	0.2162 ± 0.012	0.2335 ± 0.018	0.2658 ± 0.019	0.2768 ± 0.044	0.2641 ± 0.078	0.2614 ± 0.109	0.253 ± 0.047
Industry_Sector	[2]	GAT	0.2692 ± 0.019	0.2883 ± 0.015	0.3359 ± 0.011	0.3768 ± 0.014	0.4254 ± 0.015	0.4481 ± 0.013	0.3573 ± 0.014
Industry_Sector	[2]	GCN	0.2075 ± 0.024	0.1951 ± 0.057	0.2434 ± 0.041	0.22 ± 0.094	0.2542 ± 0.088	0.2431 ± 0.11	0.2272 ± 0.069
Industry_Sector	[3]	GAT	**0.276 ± 0.018**	**0.2943 ± 0.018**	**0.3449 ± 0.011**	**0.38 ± 0.015**	**0.4335 ± 0.012**	0.4481 ± 0.016	**0.3628 ± 0.015**
Industry_Sector	[3]	GCN	0.2219 ± 0.02	0.2383 ± 0.021	0.2335 ± 0.079	0.2822 ± 0.021	0.2282 ± 0.122	0.2685 ± 0.097	0.2454 ± 0.06
NSF	[2,3]	GAT	0.5848 ± 0.037	0.6043 ± 0.035	0.6529 ± 0.035	0.6986 ± 0.021	0.7364 ± 0.015	0.77 ± 0.011	0.6745 ± 0.026
NSF	[2,3]	GCN	0.5868 ± 0.042	0.6022 ± 0.037	0.6417 ± 0.033	0.6763 ± 0.025	0.7048 ± 0.013	0.7448 ± 0.013	0.6594 ± 0.027
NSF	[2]	GAT	0.5851 ± 0.036	0.6035 ± 0.038	**0.654 ± 0.039**	0.6986 ± 0.024	0.7344 ± 0.017	0.7692 ± 0.011	0.6741 ± 0.027
NSF	[2]	GCN	0.5789 ± 0.031	0.5914 ± 0.032	0.6298 ± 0.029	0.6808 ± 0.024	0.7086 ± 0.016	0.7429 ± 0.011	0.6554 ± 0.024
NSF	[3]	GAT	0.5877 ± 0.039	0.6079 ± 0.028	0.6508 ± 0.034	**0.7001 ± 0.022**	**0.7382 ± 0.016**	**0.7733 ± 0.01**	**0.6763 ± 0.025**
NSF	[3]	GCN	**0.5888 ± 0.038**	**0.6119 ± 0.022**	0.6424 ± 0.031	0.6851 ± 0.02	0.7091 ± 0.012	0.751 ± 0.01	0.6647 ± 0.022
SyskillWebert	[2,3]	GAT	0.6212 ± 0.031	**0.6348 ± 0.029**	**0.6336 ± 0.016**	0.6646 ± 0.051	0.7085 ± 0.075	0.6866 ± 0.057	**0.6582 ± 0.043**
SyskillWebert	[2,3]	GCN	0.5785 ± 0.045	0.6182 ± 0.035	0.6142 ± 0.045	0.6508 ± 0.079	**0.7137 ± 0.063**	0.6682 ± 0.08	0.6406 ± 0.058
SyskillWebert	[2]	GAT	**0.6219 ± 0.031**	0.6245 ± 0.024	0.6185 ± 0.02	0.671 ± 0.054	0.7086 ± 0.076	0.6834 ± 0.061	0.6546 ± 0.044
SyskillWebert	[2]	GCN	0.6213 ± 0.056	0.6151 ± 0.044	0.6127 ± 0.03	0.6461 ± 0.055	0.7055 ± 0.096	**0.693 ± 0.07**	0.6489 ± 0.058
SyskillWebert	[3]	GAT	0.6156 ± 0.028	0.626 ± 0.03	0.6182 ± 0.022	0.669 ± 0.054	0.7074 ± 0.061	0.6916 ± 0.06	0.6546 ± 0.042
SyskillWebert	[3]	GCN	0.5968 ± 0.035	0.6031 ± 0.041	0.6332 ± 0.043	**0.6747 ± 0.07**	0.7038 ± 0.057	0.6922 ± 0.072	0.6506 ± 0.053
classic4	[2,3]	GAT	**0.7633 ± 0.107**	0.7772 ± 0.099	**0.8064 ± 0.056**	**0.8788 ± 0.062**	0.9209 ± 0.014	0.9166 ± 0.026	**0.8439 ± 0.061**
classic4	[2,3]	GCN	0.7529 ± 0.101	0.7428 ± 0.057	0.7794 ± 0.053	0.8351 ± 0.067	0.8809 ± 0.025	0.8894 ± 0.018	0.8134 ± 0.054
classic4	[2]	GAT	0.7565 ± 0.107	0.7776 ± 0.105	0.8002 ± 0.063	0.8761 ± 0.068	0.9194 ± 0.015	**0.9191 ± 0.021**	0.8415 ± 0.063
classic4	[2]	GCN	0.7469 ± 0.081	0.7531 ± 0.11	0.7615 ± 0.06	0.8344 ± 0.059	0.8791 ± 0.029	0.8752 ± 0.047	0.8084 ± 0.064
classic4	[3]	GAT	0.7599 ± 0.099	**0.7799 ± 0.107**	0.7978 ± 0.054	0.874 ± 0.077	**0.9223 ± 0.013**	0.9166 ± 0.023	0.8417 ± 0.062
classic4	[3]	GCN	0.7501 ± 0.07	0.7588 ± 0.085	0.7751 ± 0.057	0.8435 ± 0.077	0.8854 ± 0.026	0.8809 ± 0.047	0.8156 ± 0.06
re8	[2,3]	GAT	**0.6358 ± 0.065**	0.6574 ± 0.065	0.695 ± 0.036	0.719 ± 0.052	**0.7619 ± 0.041**	0.7948 ± 0.018	0.7107 ± 0.046
re8	[2,3]	GCN	0.6009 ± 0.058	0.6222 ± 0.064	0.6501 ± 0.065	0.662 ± 0.06	0.6963 ± 0.052	0.7512 ± 0.02	0.6638 ± 0.053
re8	[2]	GAT	0.6305 ± 0.071	0.6379 ± 0.057	0.6884 ± 0.048	0.7117 ± 0.046	0.7507 ± 0.049	0.7856 ± 0.015	0.7008 ± 0.048
re8	[2]	GCN	0.5683 ± 0.05	0.5797 ± 0.058	0.6075 ± 0.115	0.6591 ± 0.078	0.6645 ± 0.043	0.7162 ± 0.027	0.6325 ± 0.062
re8	[3]	GAT	0.6292 ± 0.073	**0.6587 ± 0.067**	**0.7006 ± 0.046**	**0.7264 ± 0.052**	0.7593 ± 0.044	**0.7977 ± 0.021**	**0.712 ± 0.051**
re8	[3]	GCN	0.6129 ± 0.064	0.6345 ± 0.054	0.6391 ± 0.047	0.6673 ± 0.063	0.7078 ± 0.052	0.7381 ± 0.034	0.6666 ± 0.053
review_polarity	[2,3]	GAT	**0.5242 ± 0.063**	0.5454 ± 0.022	0.5385 ± 0.031	0.5091 ± 0.092	0.5426 ± 0.076	**0.5786 ± 0.021**	0.5397 ± 0.051
review_polarity	[2,3]	GCN	0.3912 ± 0.098	0.4636 ± 0.114	0.4077 ± 0.104	0.4139 ± 0.104	0.4653 ± 0.133	0.4818 ± 0.142	0.4372 ± 0.116
review_polarity	[2]	GAT	0.5204 ± 0.072	**0.5486 ± 0.026**	0.5393 ± 0.033	**0.5459 ± 0.03**	**0.5652 ± 0.019**	0.5296 ± 0.105	**0.5415 ± 0.047**
review_polarity	[2]	GCN	0.4755 ± 0.108	0.4423 ± 0.116	0.4717 ± 0.103	0.4574 ± 0.133	0.3842 ± 0.087	0.3881 ± 0.116	0.4365 ± 0.11
review_polarity	[3]	GAT	0.4745 ± 0.099	0.5299 ± 0.071	**0.5411 ± 0.034**	0.5452 ± 0.033	0.5427 ± 0.076	0.556 ± 0.079	0.5316 ± 0.065
review_polarity	[3]	GCN	0.411 ± 0.106	0.4907 ± 0.118	0.4306 ± 0.123	0.4103 ± 0.11	0.3973 ± 0.134	0.4246 ± 0.148	0.4274 ± 0.123
webkb_parsed	[2,3]	GAT	0.2555 ± 0.016	**0.2678 ± 0.022**	0.306 ± 0.015	0.3444 ± 0.021	0.3836 ± 0.015	**0.4199 ± 0.015**	**0.3295 ± 0.017**
webkb_parsed	[2,3]	GCN	0.2129 ± 0.054	0.219 ± 0.05	0.245 ± 0.052	0.2328 ± 0.079	0.2253 ± 0.098	0.2584 ± 0.104	0.2322 ± 0.073
webkb_parsed	[2]	GAT	**0.2558 ± 0.02**	0.2611 ± 0.02	0.3064 ± 0.016	0.343 ± 0.023	**0.3859 ± 0.015**	0.4124 ± 0.013	0.3274 ± 0.018
webkb_parsed	[2]	GCN	0.2199 ± 0.027	0.2097 ± 0.044	0.2223 ± 0.054	0.2283 ± 0.085	0.2231 ± 0.101	0.2043 ± 0.111	0.2179 ± 0.07
webkb_parsed	[3]	GAT	0.2558 ± 0.024	0.2629 ± 0.019	**0.3069 ± 0.011**	**0.3454 ± 0.024**	0.3805 ± 0.016	0.4189 ± 0.01	0.3284 ± 0.017
webkb_parsed	[3]	GCN	0.2099 ± 0.056	0.215 ± 0.055	0.2477 ± 0.055	0.2248 ± 0.085	0.2783 ± 0.085	0.2017 ± 0.112	0.2296 ± 0.074

Table 5.7: All LLM-10 ($LLMn_{labeled} = 10$) GAT and GCN macro F1-Socre results. The numbers represent the mean and standard deviation of all ten iterations for each dataset, the number of labeled data, and keyphrases.

Dataset	Keyphrases	GNN Model	Number of Labeled Data						Average
			LLM Only	1	5	10	20	30	Performance
CSTR	[2,3]	GAT	0.7584 ± 0.008	0.7584 ± 0.019	0.7561 ± 0.035	0.7433 ± 0.027	0.6904 ± 0.032	-	0.7413 ± 0.024
CSTR	[2,3]	GCN	**0.7692 ± 0.027**	0.7552 ± 0.039	**0.7693 ± 0.056**	**0.769 ± 0.061**	0.7103 ± 0.06	-	0.7546 ± 0.049
CSTR	[2]	GAT	0.7571 ± 0.015	**0.7597 ± 0.017**	0.7525 ± 0.024	0.7447 ± 0.027	0.6913 ± 0.031	-	0.7411 ± 0.023
CSTR	[2]	GCN	0.7674 ± 0.025	0.7596 ± 0.028	0.7489 ± 0.043	0.747 ± 0.049	0.7353 ± 0.056	-	0.7516 ± 0.04
CSTR	[3]	GAT	0.7596 ± 0.016	0.7562 ± 0.02	0.7513 ± 0.021	0.7384 ± 0.035	0.6925 ± 0.026	-	0.7396 ± 0.024
CSTR	[3]	GCN	0.7495 ± 0.043	0.7574 ± 0.032	0.7684 ± 0.041	0.7564 ± 0.038	**0.7563 ± 0.077**	-	**0.7576 ± 0.046**
Dmoz_Computers	[2,3]	GAT	0.4558 ± 0.006	0.4561 ± 0.01	0.4668 ± 0.007	0.4679 ± 0.01	0.4938 ± 0.014	0.5009 ± 0.018	0.4736 ± 0.011
Dmoz_Computers	[2,3]	GCN	0.4356 ± 0.008	0.4351 ± 0.007	0.447 ± 0.01	0.4507 ± 0.013	0.4721 ± 0.014	0.4823 ± 0.018	0.4538 ± 0.012
Dmoz_Computers	[2]	GAT	0.4528 ± 0.009	0.4554 ± 0.009	0.4649 ± 0.006	0.4652 ± 0.01	0.4906 ± 0.015	0.4992 ± 0.02	0.4713 ± 0.012
Dmoz_Computers	[2]	GCN	0.428 ± 0.008	0.4292 ± 0.007	0.4398 ± 0.007	0.4441 ± 0.013	0.4654 ± 0.012	0.4745 ± 0.014	0.4468 ± 0.01
Dmoz_Computers	[3]	GAT	**0.4577 ± 0.009**	**0.4599 ± 0.007**	**0.4705 ± 0.009**	**0.4712 ± 0.01**	**0.4946 ± 0.014**	**0.5054 ± 0.02**	**0.4765 ± 0.011**
Dmoz_Computers	[3]	GCN	0.4391 ± 0.007	0.4408 ± 0.006	0.454 ± 0.009	0.4584 ± 0.012	0.4774 ± 0.012	0.4871 ± 0.017	0.4594 ± 0.01
Dmoz_Health	[2,3]	GAT	0.7174 ± 0.02	**0.7215 ± 0.018**	0.7201 ± 0.014	0.731 ± 0.014	0.7463 ± 0.015	**0.7566 ± 0.014**	0.7321 ± 0.016
Dmoz_Health	[2,3]	GCN	0.7036 ± 0.018	0.7059 ± 0.014	0.7041 ± 0.013	0.7129 ± 0.015	0.7269 ± 0.017	0.7442 ± 0.018	0.7163 ± 0.016
Dmoz_Health	[2]	GAT	0.7158 ± 0.019	0.7206 ± 0.018	0.7177 ± 0.013	0.7283 ± 0.012	0.7442 ± 0.018	0.7522 ± 0.013	0.7298 ± 0.016
Dmoz_Health	[2]	GCN	0.7003 ± 0.018	0.7022 ± 0.017	0.6989 ± 0.017	0.7076 ± 0.011	0.7233 ± 0.016	0.7342 ± 0.015	0.7111 ± 0.016
Dmoz_Health	[3]	GAT	**0.7201 ± 0.019**	0.7197 ± 0.017	**0.7225 ± 0.012**	**0.7328 ± 0.013**	**0.7471 ± 0.016**	0.7555 ± 0.014	**0.7329 ± 0.015**
Dmoz_Health	[3]	GCN	0.7086 ± 0.021	0.7081 ± 0.017	0.7073 ± 0.017	0.7155 ± 0.01	0.7279 ± 0.018	0.7419 ± 0.014	0.7182 ± 0.016
Dmoz_Science	[2,3]	GAT	0.6516 ± 0.016	0.6509 ± 0.013	0.6455 ± 0.027	0.6453 ± 0.02	0.6595 ± 0.015	**0.6832 ± 0.013**	0.656 ± 0.017
Dmoz_Science	[2,3]	GCN	0.6424 ± 0.013	0.6385 ± 0.012	0.6395 ± 0.027	0.6366 ± 0.017	0.6476 ± 0.017	0.6671 ± 0.015	0.6453 ± 0.017
Dmoz_Science	[2]	GAT	0.6479 ± 0.014	0.6497 ± 0.013	0.6464 ± 0.026	0.6463 ± 0.019	**0.6631 ± 0.013**	0.6812 ± 0.013	0.6558 ± 0.017
Dmoz_Science	[2]	GCN	0.6385 ± 0.014	0.6366 ± 0.016	0.6343 ± 0.023	0.6325 ± 0.014	0.6446 ± 0.017	0.6672 ± 0.013	0.6423 ± 0.016
Dmoz_Science	[3]	GAT	**0.6532 ± 0.013**	**0.6533 ± 0.013**	**0.6468 ± 0.028**	**0.6478 ± 0.018**	0.6613 ± 0.014	0.6826 ± 0.016	**0.6575 ± 0.017**
Dmoz_Science	[3]	GCN	0.6428 ± 0.014	0.6447 ± 0.011	0.6055 ± 0.103	0.6386 ± 0.016	0.6543 ± 0.017	0.6723 ± 0.013	0.643 ± 0.029
Dmoz_Sports	[2,3]	GAT	0.7787 ± 0.005	0.7798 ± 0.005	0.7857 ± 0.004	0.7892 ± 0.003	0.7956 ± 0.006	0.8047 ± 0.006	0.789 ± 0.005
Dmoz_Sports	[2,3]	GCN	0.742 ± 0.006	0.7444 ± 0.004	0.7517 ± 0.006	0.7553 ± 0.003	0.7618 ± 0.006	0.7699 ± 0.006	0.7542 ± 0.005
Dmoz_Sports	[2]	GAT	0.7811 ± 0.003	0.7819 ± 0.004	0.7884 ± 0.005	0.7919 ± 0.004	0.7968 ± 0.005	0.8065 ± 0.006	0.7911 ± 0.005
Dmoz_Sports	[2]	GCN	0.7326 ± 0.004	0.7336 ± 0.005	0.7421 ± 0.006	0.745 ± 0.005	0.7495 ± 0.005	0.7572 ± 0.007	0.7433 ± 0.005
Dmoz_Sports	[3]	GAT	**0.7841 ± 0.005**	**0.7844 ± 0.004**	**0.7903 ± 0.006**	**0.7958 ± 0.003**	**0.7996 ± 0.006**	**0.8074 ± 0.006**	**0.7936 ± 0.005**
Dmoz_Sports	[3]	GCN	0.7503 ± 0.006	0.7472 ± 0.005	0.7522 ± 0.006	0.7582 ± 0.005	0.7679 ± 0.006	0.7742 ± 0.005	0.7583 ± 0.005
Industry_Sector	[2,3]	GAT	**0.3471 ± 0.011**	0.3477 ± 0.014	0.3564 ± 0.013	**0.3727 ± 0.012**	**0.3863 ± 0.013**	**0.407 ± 0.016**	**0.3695 ± 0.013**
Industry_Sector	[2,3]	GCN	0.2361 ± 0.047	0.2082 ± 0.086	0.1913 ± 0.08	0.1743 ± 0.099	0.2125 ± 0.101	0.2345 ± 0.08	0.2095 ± 0.082
Industry_Sector	[2]	GAT	0.344 ± 0.01	0.3465 ± 0.012	0.3467 ± 0.013	0.3662 ± 0.009	0.3814 ± 0.013	0.4032 ± 0.014	0.3647 ± 0.012
Industry_Sector	[2]	GCN	0.1878 ± 0.082	0.2181 ± 0.096	0.2067 ± 0.068	0.2016 ± 0.077	0.155 ± 0.081	0.1567 ± 0.083	0.1876 ± 0.081
Industry_Sector	[3]	GAT	0.347 ± 0.012	**0.3478 ± 0.01**	**0.3572 ± 0.01**	0.3703 ± 0.007	0.3844 ± 0.015	0.4037 ± 0.016	0.3684 ± 0.012
Industry_Sector	[3]	GCN	0.2072 ± 0.074	0.248 ± 0.066	0.1874 ± 0.071	0.2279 ± 0.094	0.1913 ± 0.105	0.2518 ± 0.085	0.2189 ± 0.082
NSF	[2,3]	GAT	**0.605 ± 0.025**	0.6043 ± 0.025	0.6361 ± 0.03	0.634 ± 0.024	0.6682 ± 0.039	0.6813 ± 0.038	0.6381 ± 0.03
NSF	[2,3]	GCN	0.5916 ± 0.023	0.5955 ± 0.023	0.6257 ± 0.032	0.6174 ± 0.023	0.6485 ± 0.033	0.6651 ± 0.033	0.624 ± 0.028
NSF	[2]	GAT	0.6031 ± 0.023	0.6068 ± 0.026	**0.6388 ± 0.028**	0.6296 ± 0.023	0.6703 ± 0.035	0.6803 ± 0.042	0.6381 ± 0.03
NSF	[2]	GCN	0.5922 ± 0.021	0.5928 ± 0.024	0.6076 ± 0.066	0.6184 ± 0.024	0.6492 ± 0.03	0.6579 ± 0.035	0.6197 ± 0.033
NSF	[3]	GAT	0.6015 ± 0.027	**0.6074 ± 0.025**	**0.6388 ± 0.031**	**0.6378 ± 0.025**	**0.6713 ± 0.037**	**0.6832 ± 0.04**	**0.64 ± 0.031**
NSF	[3]	GCN	0.5901 ± 0.026	0.6032 ± 0.017	0.6278 ± 0.026	0.6207 ± 0.021	0.6515 ± 0.033	0.6643 ± 0.037	0.6263 ± 0.027
SyskillWebert	[2,3]	GAT	0.6136 ± 0.014	0.6118 ± 0.013	0.6123 ± 0.016	0.6062 ± 0.018	0.6327 ± 0.028	0.6212 ± 0.047	0.6163 ± 0.023
SyskillWebert	[2,3]	GCN	0.6126 ± 0.02	0.6133 ± 0.019	0.6141 ± 0.022	0.6091 ± 0.021	0.6433 ± 0.05	0.6216 ± 0.098	0.619 ± 0.038
SyskillWebert	[2]	GAT	0.6103 ± 0.015	**0.6159 ± 0.014**	0.6127 ± 0.016	0.6112 ± 0.015	0.627 ± 0.037	0.6168 ± 0.039	0.6156 ± 0.023
SyskillWebert	[2]	GCN	**0.6283 ± 0.019**	0.595 ± 0.056	0.597 ± 0.08	0.6127 ± 0.02	0.6382 ± 0.024	**0.6374 ± 0.089**	0.6181 ± 0.048
SyskillWebert	[3]	GAT	0.6113 ± 0.013	0.6116 ± 0.015	0.618 ± 0.019	0.6051 ± 0.017	0.6207 ± 0.037	0.6278 ± 0.041	0.6157 ± 0.024
SyskillWebert	[3]	GCN	0.6103 ± 0.017	0.6094 ± 0.03	**0.6203 ± 0.032**	**0.6158 ± 0.028**	**0.6565 ± 0.056**	0.6195 ± 0.029	**0.622 ± 0.032**
classic4	[2,3]	GAT	0.7954 ± 0.074	**0.8058 ± 0.075**	**0.8287 ± 0.053**	0.8084 ± 0.076	0.8346 ± 0.053	**0.8759 ± 0.044**	0.8248 ± 0.062
classic4	[2,3]	GCN	0.7913 ± 0.066	0.7851 ± 0.069	0.7921 ± 0.049	0.7879 ± 0.075	0.8224 ± 0.059	0.853 ± 0.043	0.8053 ± 0.06
classic4	[2]	GAT	0.7958 ± 0.075	0.8036 ± 0.073	0.8256 ± 0.062	0.8124 ± 0.075	0.8348 ± 0.061	0.8694 ± 0.046	0.8236 ± 0.065
classic4	[2]	GCN	0.7877 ± 0.068	0.7724 ± 0.071	0.7909 ± 0.056	0.7878 ± 0.071	0.8182 ± 0.059	0.8453 ± 0.034	0.8004 ± 0.06
classic4	[3]	GAT	**0.7999 ± 0.077**	0.8041 ± 0.076	0.8249 ± 0.058	**0.8141 ± 0.073**	**0.8388 ± 0.054**	0.8711 ± 0.043	**0.8255 ± 0.064**
classic4	[3]	GCN	0.792 ± 0.063	0.7778 ± 0.066	0.7852 ± 0.054	0.7915 ± 0.076	0.8094 ± 0.047	0.8428 ± 0.049	0.7998 ± 0.059
re8	[2,3]	GAT	0.6887 ± 0.061	0.6899 ± 0.062	0.7094 ± 0.081	0.7226 ± 0.039	0.7312 ± 0.04	0.7568 ± 0.037	0.7164 ± 0.053
re8	[2,3]	GCN	0.6615 ± 0.06	0.6555 ± 0.052	0.6738 ± 0.078	0.689 ± 0.034	0.6971 ± 0.034	0.7045 ± 0.038	0.6802 ± 0.049
re8	[2]	GAT	0.6827 ± 0.065	0.6859 ± 0.066	0.7112 ± 0.083	0.7187 ± 0.04	0.7283 ± 0.042	0.752 ± 0.04	0.7132 ± 0.056
re8	[2]	GCN	0.6479 ± 0.057	0.6398 ± 0.055	0.6545 ± 0.068	0.6726 ± 0.049	0.6593 ± 0.046	0.6854 ± 0.048	0.6599 ± 0.054
re8	[3]	GAT	**0.6902 ± 0.064**	**0.6928 ± 0.06**	**0.7165 ± 0.083**	**0.7273 ± 0.039**	**0.7318 ± 0.042**	**0.7595 ± 0.039**	**0.7197 ± 0.055**
re8	[3]	GCN	0.6615 ± 0.061	0.6575 ± 0.044	0.6732 ± 0.074	0.6824 ± 0.051	0.6852 ± 0.049	0.7024 ± 0.047	0.677 ± 0.054
review_polarity	[2,3]	GAT	**0.5844 ± 0.008**	0.5583 ± 0.081	0.5682 ± 0.106	0.4964 ± 0.127	0.5466 ± 0.125	**0.5705 ± 0.085**	0.5541 ± 0.089
review_polarity	[2,3]	GCN	0.4864 ± 0.14	0.3634 ± 0.088	0.3961 ± 0.129	0.4058 ± 0.144	0.3724 ± 0.075	0.4552 ± 0.161	0.4132 ± 0.123
review_polarity	[2]	GAT	0.5794 ± 0.037	**0.5661 ± 0.083**	0.556 ± 0.119	**0.5368 ± 0.109**	**0.5656 ± 0.121**	0.5672 ± 0.125	**0.5618 ± 0.099**
review_polarity	[2]	GCN	0.3788 ± 0.089	0.3434 ± 0.026	0.3968 ± 0.117	0.4488 ± 0.135	0.3892 ± 0.118	0.5355 ± 0.116	0.4154 ± 0.1
review_polarity	[3]	GAT	0.5621 ± 0.082	0.5465 ± 0.095	**0.6176 ± 0.017**	0.5217 ± 0.13	0.5631 ± 0.118	0.5366 ± 0.114	0.5579 ± 0.093
review_polarity	[3]	GCN	0.4007 ± 0.128	0.4001 ± 0.138	0.3676 ± 0.075	0.4144 ± 0.118	0.4576 ± 0.142	0.3891 ± 0.122	0.4049 ± 0.121
webkb_parsed	[2,3]	GAT	0.3168 ± 0.01	0.3164 ± 0.009	0.3281 ± 0.011	0.3407 ± 0.014	0.356 ± 0.01	**0.3801 ± 0.014**	0.3397 ± 0.011
webkb_parsed	[2,3]	GCN	0.1965 ± 0.092	0.2176 ± 0.078	0.2097 ± 0.088	0.1979 ± 0.053	0.2367 ± 0.066	0.1928 ± 0.09	0.2085 ± 0.078
webkb_parsed	[2]	GAT	0.3148 ± 0.01	0.3177 ± 0.013	0.3285 ± 0.012	**0.3409 ± 0.013**	0.3564 ± 0.008	0.378 ± 0.016	0.3394 ± 0.012
webkb_parsed	[2]	GCN	0.2172 ± 0.08	0.1787 ± 0.068	0.2293 ± 0.078	0.2048 ± 0.074	0.1869 ± 0.087	0.154 ± 0.038	0.1951 ± 0.071
webkb_parsed	[3]	GAT	**0.3174 ± 0.011**	**0.3184 ± 0.01**	**0.3289 ± 0.011**	0.3385 ± 0.014	**0.3594 ± 0.011**	0.379 ± 0.018	**0.3403 ± 0.012**
webkb_parsed	[3]	GCN	0.1848 ± 0.061	0.2256 ± 0.064	0.2037 ± 0.08	0.2196 ± 0.095	0.1931 ± 0.064	0.1914 ± 0.094	0.203 ± 0.076

Table 5.8: All LLM-50 ($LLMn_{labeled} = 50$) GAT and GCN macro F1-Socre results. The numbers represent the mean and standard deviation of all ten iterations for each dataset, the number of labeled data, and keyphrases.

Dataset	Keyphrases	GNN Model	Number of Labeled Data						Average
			LLM Only	1	5	10	20	30	Performance
CSTR	[2,3]	GAT	0.7671 ± 0.016	0.7677 ± 0.018	0.7578 ± 0.017	0.7395 ± 0.013	0.6884 ± 0.036	-	0.7441 ± 0.02
CSTR	[2,3]	GCN	0.7977 ± 0.044	**0.7915 ± 0.014**	**0.7906 ± 0.035**	**0.7672 ± 0.027**	0.7261 ± 0.038	-	0.7746 ± 0.032
CSTR	[2]	GAT	0.7651 ± 0.018	0.7626 ± 0.016	0.7586 ± 0.011	0.7419 ± 0.015	0.6914 ± 0.036	-	0.7439 ± 0.019
CSTR	[2]	GCN	0.7916 ± 0.014	0.7531 ± 0.059	0.7365 ± 0.101	0.763 ± 0.026	0.7067 ± 0.046	-	0.7502 ± 0.049
CSTR	[3]	GAT	0.7699 ± 0.015	0.7716 ± 0.015	0.7589 ± 0.017	0.7372 ± 0.015	0.6909 ± 0.02	-	0.7457 ± 0.016
CSTR	[3]	GCN	**0.7991 ± 0.03**	0.7907 ± 0.02	0.7783 ± 0.041	0.7644 ± 0.029	**0.7492 ± 0.065**	-	**0.7763 ± 0.037**
Dmoz_Computers	[2,3]	GAT	0.4561 ± 0.006	0.4557 ± 0.007	0.4631 ± 0.007	0.4638 ± 0.006	0.4687 ± 0.007	0.4761 ± 0.017	0.4639 ± 0.008
Dmoz_Computers	[2,3]	GCN	0.4385 ± 0.006	0.4392 ± 0.006	0.447 ± 0.008	0.4463 ± 0.006	0.4524 ± 0.011	0.4643 ± 0.018	0.448 ± 0.009
Dmoz_Computers	[2]	GAT	0.453 ± 0.007	0.452 ± 0.005	0.4608 ± 0.007	0.4605 ± 0.007	0.4667 ± 0.01	0.4741 ± 0.016	0.4612 ± 0.009
Dmoz_Computers	[2]	GCN	0.4333 ± 0.008	0.4348 ± 0.008	0.4442 ± 0.007	0.4435 ± 0.007	0.4521 ± 0.011	0.4629 ± 0.018	0.4451 ± 0.01
Dmoz_Computers	[3]	GAT	**0.4577 ± 0.007**	**0.457 ± 0.008**	**0.4656 ± 0.006**	**0.4642 ± 0.008**	**0.4699 ± 0.007**	**0.4776 ± 0.018**	**0.4653 ± 0.009**
Dmoz_Computers	[3]	GCN	0.4472 ± 0.005	0.4443 ± 0.005	0.4517 ± 0.009	0.4525 ± 0.011	0.456 ± 0.008	0.4688 ± 0.017	0.4534 ± 0.009
Dmoz_Health	[2,3]	GAT	0.717 ± 0.019	0.7183 ± 0.019	0.7247 ± 0.017	0.7352 ± 0.014	**0.7397 ± 0.014**	0.7461 ± 0.006	0.7302 ± 0.015
Dmoz_Health	[2,3]	GCN	0.7056 ± 0.024	0.7085 ± 0.021	0.7135 ± 0.016	0.7234 ± 0.012	0.7279 ± 0.017	0.7358 ± 0.008	0.7191 ± 0.016
Dmoz_Health	[2]	GAT	0.7167 ± 0.019	0.7164 ± 0.02	0.7233 ± 0.017	0.7351 ± 0.015	0.7378 ± 0.013	0.746 ± 0.006	0.7292 ± 0.015
Dmoz_Health	[2]	GCN	0.7019 ± 0.017	0.7016 ± 0.021	0.7086 ± 0.022	0.7203 ± 0.012	0.7232 ± 0.014	0.7302 ± 0.006	0.7143 ± 0.015
Dmoz_Health	[3]	GAT	**0.7188 ± 0.018**	**0.7188 ± 0.017**	**0.7251 ± 0.015**	**0.7368 ± 0.015**	0.7394 ± 0.015	**0.7479 ± 0.006**	**0.7311 ± 0.014**
Dmoz_Health	[3]	GCN	0.7081 ± 0.021	0.7056 ± 0.016	0.7131 ± 0.014	0.7241 ± 0.013	0.7292 ± 0.012	0.7388 ± 0.011	0.7198 ± 0.014
Dmoz_Science	[2,3]	GAT	0.6432 ± 0.029	0.6434 ± 0.03	0.6507 ± 0.018	0.6638 ± 0.019	0.6707 ± 0.018	0.6709 ± 0.016	0.6571 ± 0.022
Dmoz_Science	[2,3]	GCN	0.6025 ± 0.082	0.6294 ± 0.028	0.6366 ± 0.019	0.6518 ± 0.018	0.6559 ± 0.016	0.6555 ± 0.017	0.6386 ± 0.03
Dmoz_Science	[2]	GAT	0.6407 ± 0.029	0.6423 ± 0.028	**0.6509 ± 0.016**	0.6642 ± 0.019	0.6687 ± 0.016	0.6697 ± 0.016	0.6561 ± 0.021
Dmoz_Science	[2]	GCN	0.6272 ± 0.027	0.628 ± 0.027	0.6354 ± 0.016	0.6489 ± 0.019	0.6496 ± 0.018	0.6516 ± 0.017	0.6401 ± 0.021
Dmoz_Science	[3]	GAT	**0.644 ± 0.028**	**0.6456 ± 0.028**	0.6508 ± 0.018	**0.6662 ± 0.02**	**0.6708 ± 0.016**	**0.6728 ± 0.016**	**0.6584 ± 0.021**
Dmoz_Science	[3]	GCN	0.6144 ± 0.067	0.634 ± 0.028	0.6391 ± 0.017	0.6551 ± 0.02	0.6614 ± 0.017	0.6597 ± 0.018	0.6439 ± 0.028
Dmoz_Sports	[2,3]	GAT	0.7983 ± 0.004	0.7987 ± 0.004	0.7979 ± 0.005	0.8001 ± 0.006	0.8041 ± 0.004	0.8076 ± 0.008	0.8011 ± 0.005
Dmoz_Sports	[2,3]	GCN	0.7653 ± 0.005	0.7685 ± 0.006	0.7654 ± 0.006	0.7654 ± 0.008	0.7702 ± 0.008	0.7718 ± 0.009	0.7678 ± 0.007
Dmoz_Sports	[2]	GAT	0.7999 ± 0.005	0.8009 ± 0.005	0.8003 ± 0.004	0.8019 ± 0.006	0.8054 ± 0.004	0.8093 ± 0.008	0.803 ± 0.005
Dmoz_Sports	[2]	GCN	0.7558 ± 0.004	0.7551 ± 0.006	0.7551 ± 0.004	0.7561 ± 0.006	0.7548 ± 0.004	0.7615 ± 0.006	0.7564 ± 0.005
Dmoz_Sports	[3]	GAT	**0.8011 ± 0.005**	**0.8013 ± 0.005**	**0.8026 ± 0.005**	**0.8028 ± 0.007**	**0.8076 ± 0.006**	**0.8111 ± 0.008**	**0.8044 ± 0.006**
Dmoz_Sports	[3]	GCN	0.773 ± 0.005	0.7711 ± 0.004	0.7721 ± 0.005	0.7714 ± 0.008	0.7771 ± 0.005	0.7805 ± 0.007	0.7742 ± 0.006
Industry_Sector	[2,3]	GAT	**0.3606 ± 0.009**	**0.3624 ± 0.008**	**0.3686 ± 0.01**	0.3645 ± 0.006	0.3756 ± 0.013	**0.3933 ± 0.012**	**0.3708 ± 0.01**
Industry_Sector	[2,3]	GCN	0.2482 ± 0.056	0.2402 ± 0.079	0.2318 ± 0.084	0.1428 ± 0.065	0.2134 ± 0.061	0.169 ± 0.089	0.2076 ± 0.072
Industry_Sector	[2]	GAT	0.3578 ± 0.007	0.3601 ± 0.008	0.3667 ± 0.01	0.362 ± 0.009	0.374 ± 0.013	0.3899 ± 0.011	0.3684 ± 0.01
Industry_Sector	[2]	GCN	0.1835 ± 0.083	0.2091 ± 0.088	0.1655 ± 0.101	0.1493 ± 0.069	0.1979 ± 0.092	0.1571 ± 0.086	0.1771 ± 0.086
Industry_Sector	[3]	GAT	0.3606 ± 0.008	0.3599 ± 0.007	0.3681 ± 0.012	**0.3661 ± 0.009**	**0.3765 ± 0.012**	0.3919 ± 0.012	0.3705 ± 0.01
Industry_Sector	[3]	GCN	0.2098 ± 0.074	0.2286 ± 0.071	0.1819 ± 0.087	0.2009 ± 0.08	0.2068 ± 0.082	0.2602 ± 0.062	0.2147 ± 0.076
NSF	[2,3]	GAT	0.617 ± 0.039	0.6171 ± 0.038	0.627 ± 0.038	0.6288 ± 0.046	0.638 ± 0.024	0.6567 ± 0.024	0.6308 ± 0.035
NSF	[2,3]	GCN	0.6014 ± 0.033	0.6034 ± 0.034	0.6153 ± 0.035	0.6152 ± 0.037	0.6251 ± 0.022	0.6356 ± 0.023	0.616 ± 0.031
NSF	[2]	GAT	0.6147 ± 0.038	0.6164 ± 0.042	0.6265 ± 0.039	0.6253 ± 0.045	0.637 ± 0.022	0.6519 ± 0.024	0.6286 ± 0.035
NSF	[2]	GCN	0.6023 ± 0.037	0.6028 ± 0.038	0.618 ± 0.033	0.6145 ± 0.041	0.6288 ± 0.022	0.6342 ± 0.024	0.6168 ± 0.032
NSF	[3]	GAT	**0.6176 ± 0.04**	**0.6198 ± 0.038**	**0.6299 ± 0.04**	**0.6305 ± 0.049**	**0.6409 ± 0.027**	**0.6573 ± 0.022**	**0.6327 ± 0.036**
NSF	[3]	GCN	0.6092 ± 0.036	0.6065 ± 0.035	0.618 ± 0.032	0.6171 ± 0.039	0.632 ± 0.024	0.6391 ± 0.024	0.6203 ± 0.032
SyskillWebert	[2,3]	GAT	0.602 ± 0.015	0.6055 ± 0.015	0.6026 ± 0.016	0.6029 ± 0.014	0.6116 ± 0.03	0.626 ± 0.048	0.6084 ± 0.023
SyskillWebert	[2,3]	GCN	0.6059 ± 0.016	**0.6152 ± 0.025**	0.6082 ± 0.029	**0.6179 ± 0.03**	0.6285 ± 0.04	0.6424 ± 0.067	0.6197 ± 0.035
SyskillWebert	[2]	GAT	0.6011 ± 0.012	0.6011 ± 0.016	0.6011 ± 0.015	0.609 ± 0.014	0.6154 ± 0.022	0.6288 ± 0.058	0.6094 ± 0.023
SyskillWebert	[2]	GCN	0.6063 ± 0.026	0.6143 ± 0.03	**0.6155 ± 0.027**	0.6096 ± 0.014	**0.6332 ± 0.03**	**0.6612 ± 0.08**	**0.6234 ± 0.035**
SyskillWebert	[3]	GAT	0.6016 ± 0.013	0.5978 ± 0.014	0.6057 ± 0.014	0.6046 ± 0.018	0.6237 ± 0.027	0.621 ± 0.052	0.6091 ± 0.023
SyskillWebert	[3]	GCN	**0.6065 ± 0.017**	0.6019 ± 0.02	0.6126 ± 0.03	0.6162 ± 0.027	0.6123 ± 0.021	0.6548 ± 0.058	0.6174 ± 0.029
classic4	[2,3]	GAT	0.7968 ± 0.08	0.7974 ± 0.082	0.794 ± 0.08	0.8188 ± 0.06	**0.8438 ± 0.055**	**0.8338 ± 0.039**	0.8141 ± 0.066
classic4	[2,3]	GCN	0.7847 ± 0.087	0.7808 ± 0.077	0.7697 ± 0.07	0.7988 ± 0.057	0.828 ± 0.051	0.8089 ± 0.04	0.7951 ± 0.064
classic4	[2]	GAT	0.7972 ± 0.082	0.8015 ± 0.078	**0.7946 ± 0.078**	0.8222 ± 0.059	0.8425 ± 0.057	0.833 ± 0.036	**0.8152 ± 0.065**
classic4	[2]	GCN	0.7728 ± 0.076	0.7763 ± 0.068	0.7741 ± 0.071	0.8045 ± 0.044	0.8231 ± 0.048	0.8156 ± 0.033	0.7944 ± 0.057
classic4	[3]	GAT	**0.7988 ± 0.08**	**0.8021 ± 0.077**	0.7928 ± 0.076	**0.8241 ± 0.06**	0.8418 ± 0.058	0.8259 ± 0.04	0.8143 ± 0.065
classic4	[3]	GCN	0.7852 ± 0.083	0.7845 ± 0.077	0.765 ± 0.066	0.7925 ± 0.056	0.8281 ± 0.053	0.8181 ± 0.039	0.7956 ± 0.062
re8	[2,3]	GAT	**0.7474 ± 0.07**	**0.7481 ± 0.07**	**0.7386 ± 0.061**	0.7574 ± 0.052	**0.7349 ± 0.047**	0.7058 ± 0.045	**0.7387 ± 0.058**
re8	[2,3]	GCN	0.7094 ± 0.076	0.7189 ± 0.07	0.7086 ± 0.061	0.7158 ± 0.057	0.6948 ± 0.043	0.6586 ± 0.044	0.701 ± 0.059
re8	[2]	GAT	0.7406 ± 0.07	0.7411 ± 0.071	0.727 ± 0.058	0.7526 ± 0.053	0.7319 ± 0.051	0.7006 ± 0.045	0.7323 ± 0.058
re8	[2]	GCN	0.7049 ± 0.073	0.7036 ± 0.069	0.6906 ± 0.058	0.6985 ± 0.045	0.6883 ± 0.046	0.6132 ± 0.138	0.6832 ± 0.071
re8	[3]	GAT	0.7464 ± 0.071	0.7459 ± 0.071	0.737 ± 0.064	**0.7581 ± 0.052**	0.7344 ± 0.049	**0.709 ± 0.043**	0.7385 ± 0.058
re8	[3]	GCN	0.7141 ± 0.056	0.7273 ± 0.073	0.7042 ± 0.061	0.737 ± 0.048	0.6976 ± 0.045	0.6489 ± 0.044	0.7049 ± 0.054
review_polarity	[2,3]	GAT	**0.5299 ± 0.149**	0.5957 ± 0.125	0.5091 ± 0.15	0.5569 ± 0.151	0.5704 ± 0.127	**0.6199 ± 0.103**	**0.5637 ± 0.134**
review_polarity	[2,3]	GCN	0.4305 ± 0.131	0.4511 ± 0.14	0.3989 ± 0.139	0.3757 ± 0.082	0.3638 ± 0.065	0.3428 ± 0.021	0.3938 ± 0.096
review_polarity	[2]	GAT	0.5295 ± 0.149	**0.6288 ± 0.104**	0.4932 ± 0.156	0.4784 ± 0.156	**0.5969 ± 0.119**	0.5349 ± 0.138	0.5436 ± 0.137
review_polarity	[2]	GCN	0.3333 ± 0.0	0.4568 ± 0.137	0.3667 ± 0.103	0.4342 ± 0.162	0.4032 ± 0.147	0.392 ± 0.114	0.3977 ± 0.111
review_polarity	[3]	GAT	0.5129 ± 0.157	0.5388 ± 0.15	**0.5705 ± 0.129**	**0.5879 ± 0.117**	0.5606 ± 0.105	0.5693 ± 0.151	0.5567 ± 0.135
review_polarity	[3]	GCN	0.4185 ± 0.118	0.3868 ± 0.123	0.3638 ± 0.053	0.3981 ± 0.116	0.3873 ± 0.105	0.3748 ± 0.114	0.3882 ± 0.105
webkb_parsed	[2,3]	GAT	0.3357 ± 0.008	0.3373 ± 0.008	0.3429 ± 0.01	0.342 ± 0.006	**0.3471 ± 0.009**	**0.3539 ± 0.011**	**0.3432 ± 0.009**
webkb_parsed	[2,3]	GCN	0.2211 ± 0.087	0.2103 ± 0.062	0.1957 ± 0.085	0.1842 ± 0.064	0.1778 ± 0.089	0.1913 ± 0.075	0.1967 ± 0.077
webkb_parsed	[2]	GAT	0.3365 ± 0.007	0.3369 ± 0.008	**0.3447 ± 0.012**	0.3421 ± 0.01	0.3452 ± 0.007	0.3515 ± 0.014	0.3428 ± 0.009
webkb_parsed	[2]	GCN	0.1986 ± 0.093	0.154 ± 0.057	0.1713 ± 0.079	0.1778 ± 0.077	0.1388 ± 0.066	0.1465 ± 0.041	0.1645 ± 0.069
webkb_parsed	[3]	GAT	**0.3375 ± 0.007**	**0.3378 ± 0.01**	0.3434 ± 0.01	**0.3422 ± 0.007**	0.3464 ± 0.007	0.3519 ± 0.011	**0.3432 ± 0.009**
webkb_parsed	[3]	GCN	0.1977 ± 0.065	0.1658 ± 0.097	0.1956 ± 0.068	0.189 ± 0.071	0.2001 ± 0.08	0.1472 ± 0.058	0.1826 ± 0.073

Table 5.9: All LLM-100 ($LLMn_{labeled} = 100$) GAT and GCN macro F1-Socre results. The numbers represent the mean and standard deviation of all ten iterations for each dataset, the number of labeled data, and keyphrases.

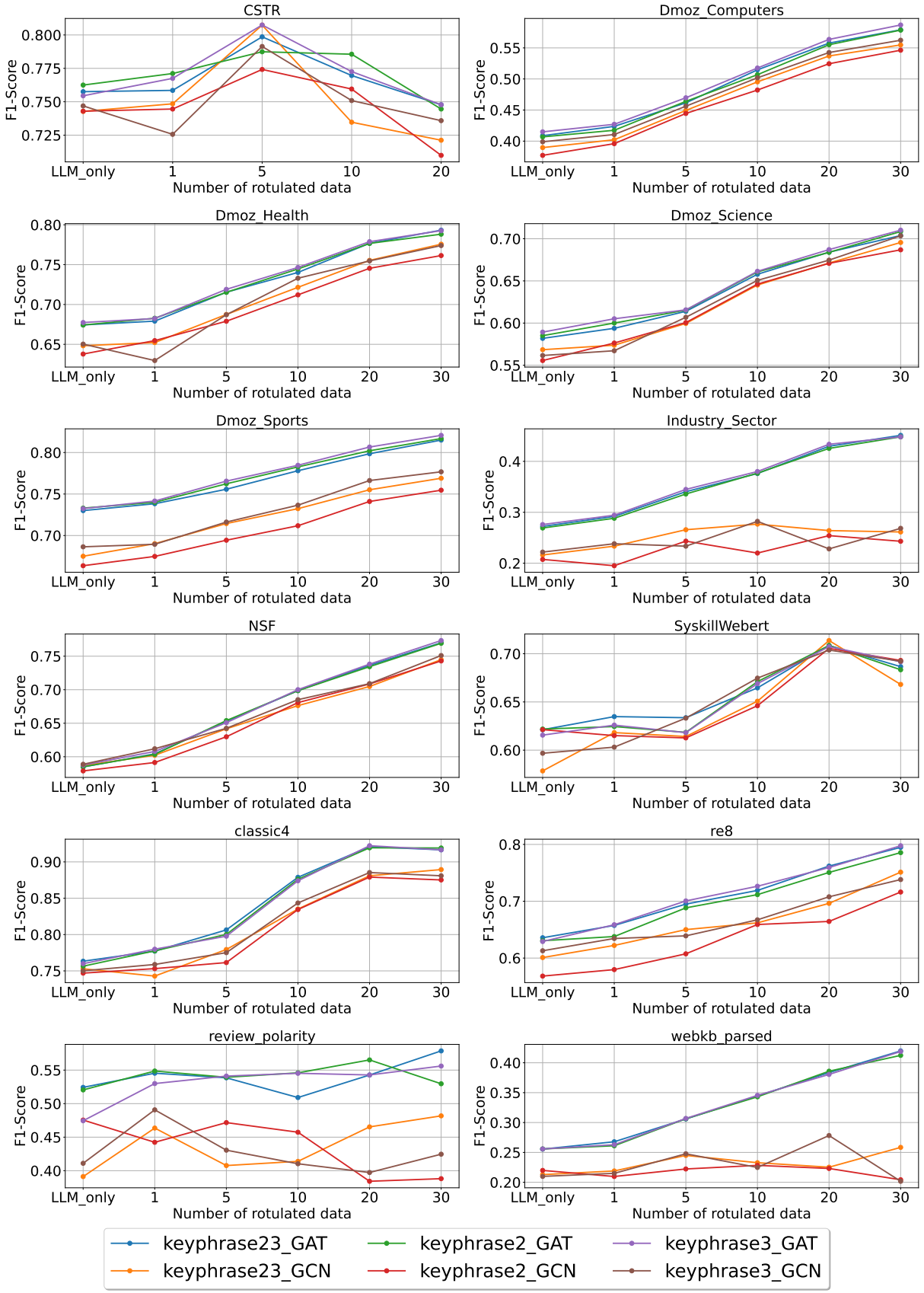


Figure 5.8: All GAT and GCN models trained with 10 LLM labeled instances ($LLMn_{labeled} = 10$) F1-scores results, over the evolution of labeled node numbers.

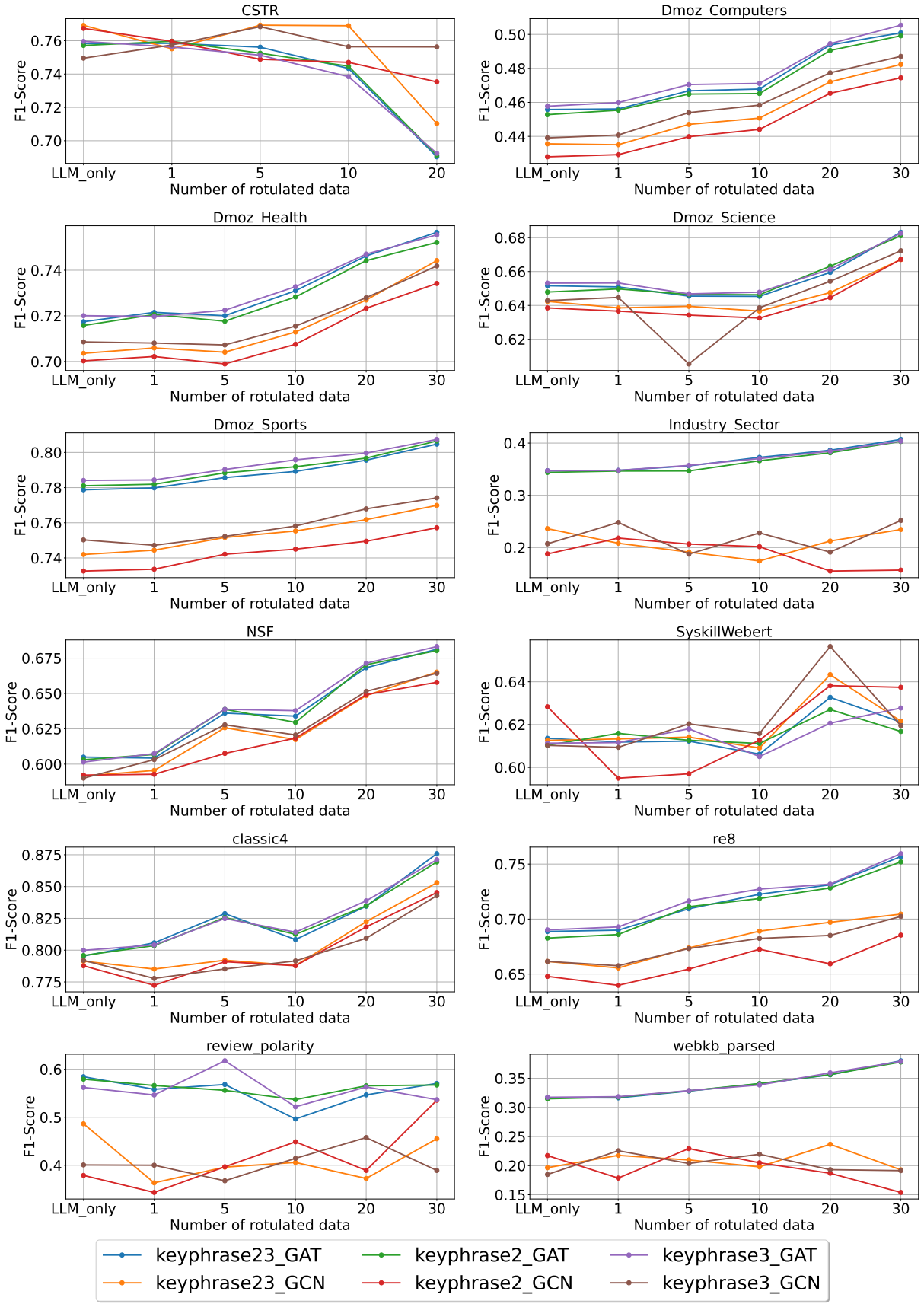


Figure 5.9: All GAT and GCN models trained with 50 LLM labeled instances ($LLMn_{labeled} = 50$) F1-scores results, over the evolution of labeled node numbers.

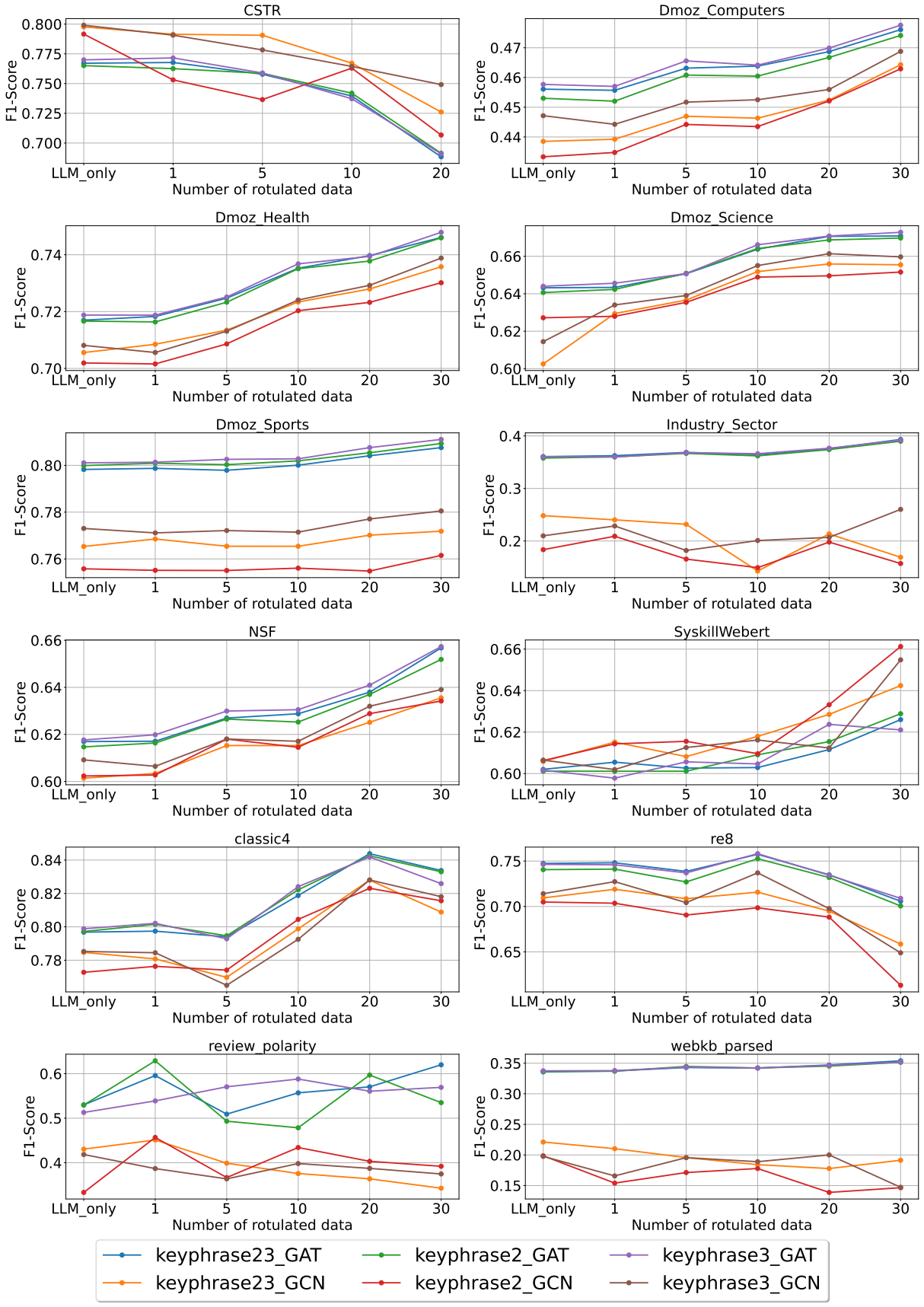


Figure 5.10: All GAT and GCN models trained with 100 LLM labeled instances ($LLMn_{labeled} = 100$) F1-scores results, over the evolution of labeled node numbers.

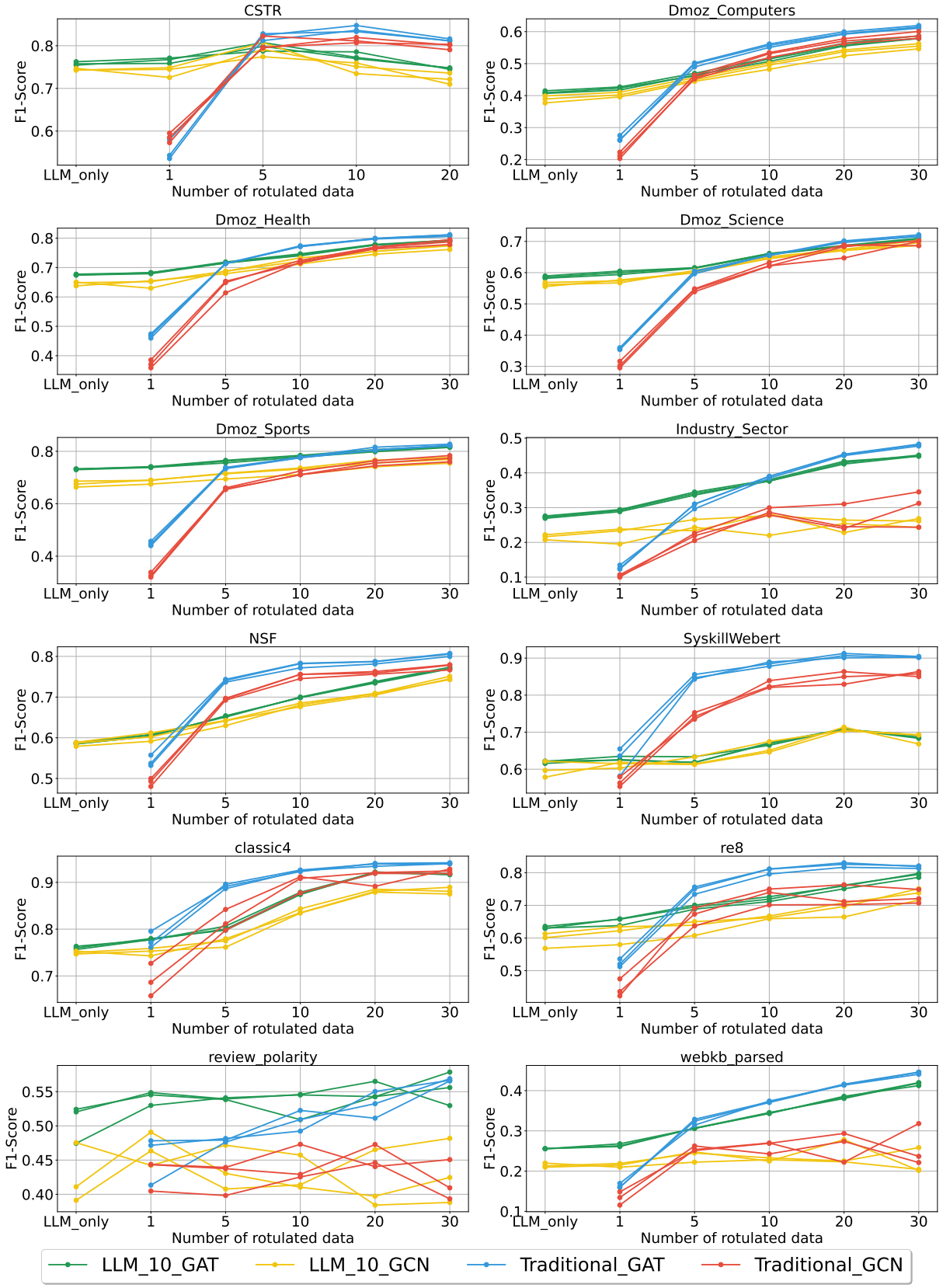


Figure 5.11: Comparative evolutionary graph between all GAT and GCN models trained with 10 LLM labeled instances ($LLMn_{labeled} = 10$) and traditionally labeled GAT and GCN models F1-scores results.

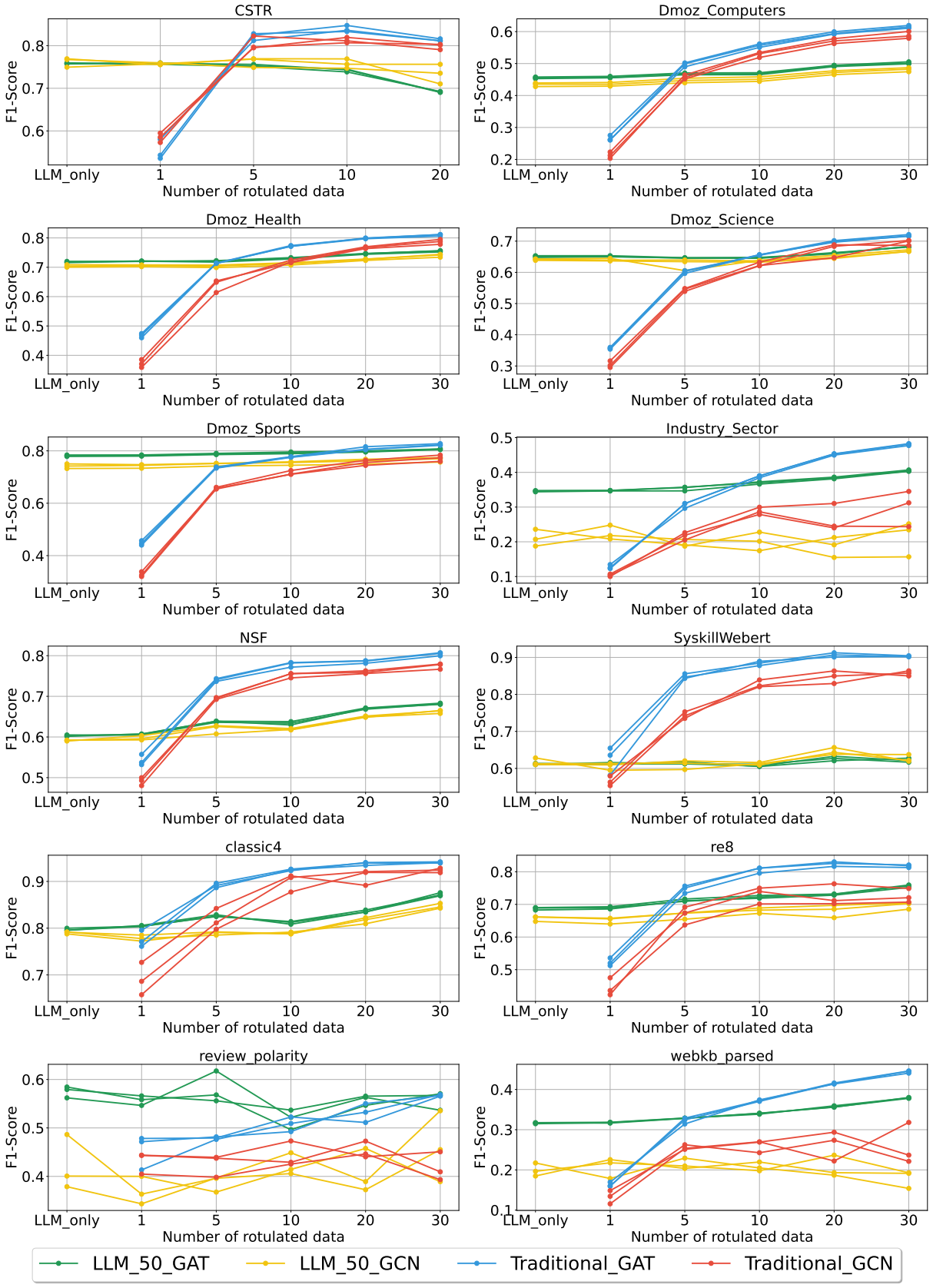


Figure 5.12: Comparative evolutionary graph between all GAT and GCN models trained with 50 LLM labeled instances ($LLMn_{labeled} = 50$) and traditionally labeled GAT and GCN models F1-scores results.

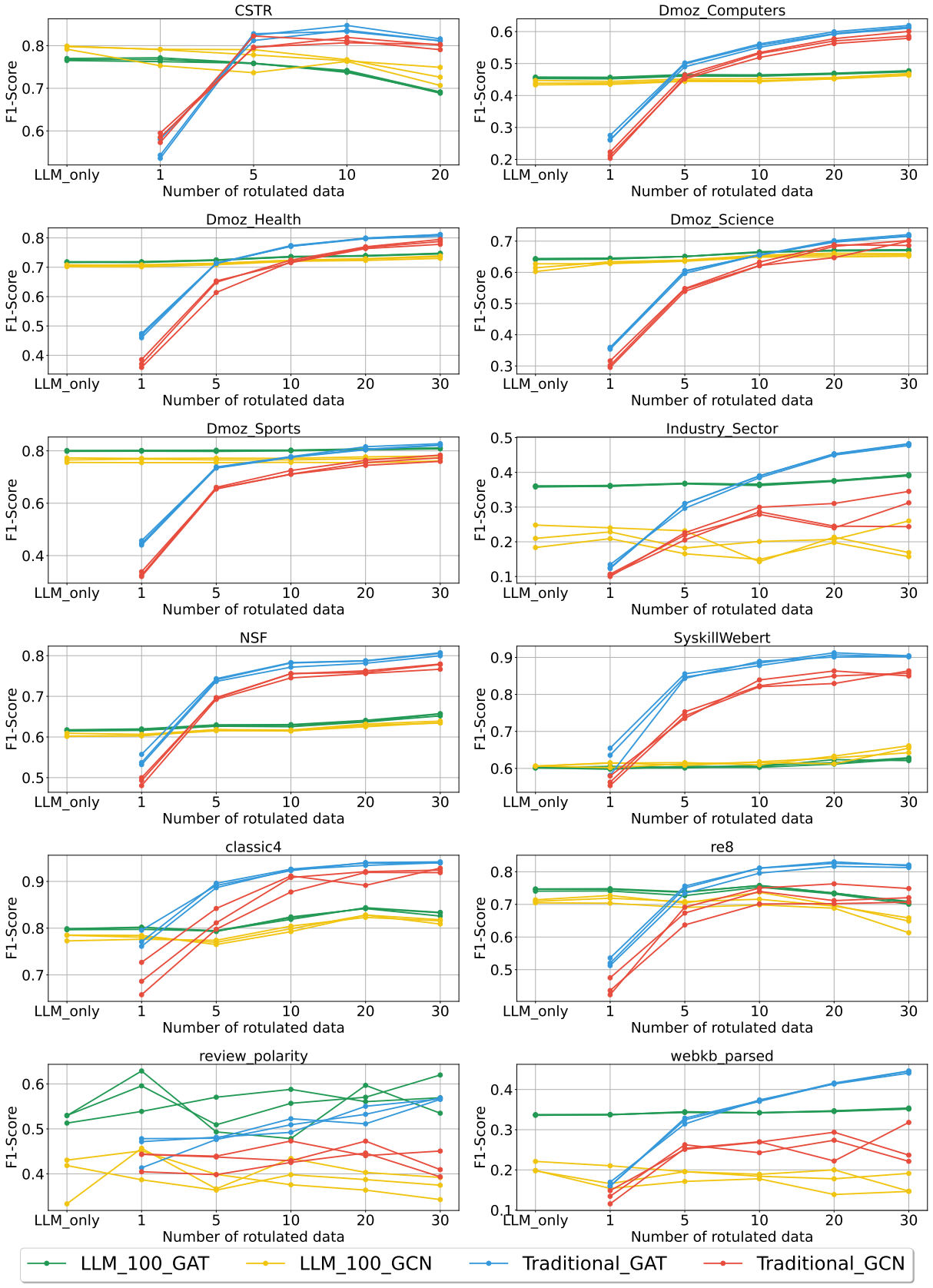


Figure 5.13: Comparative evolutionary graph between all GAT and GCN models trained with 100 LLM labeled instances ($LLMn_{labeled} = 100$) and traditionally labeled GAT and GCN models F1-scores results.

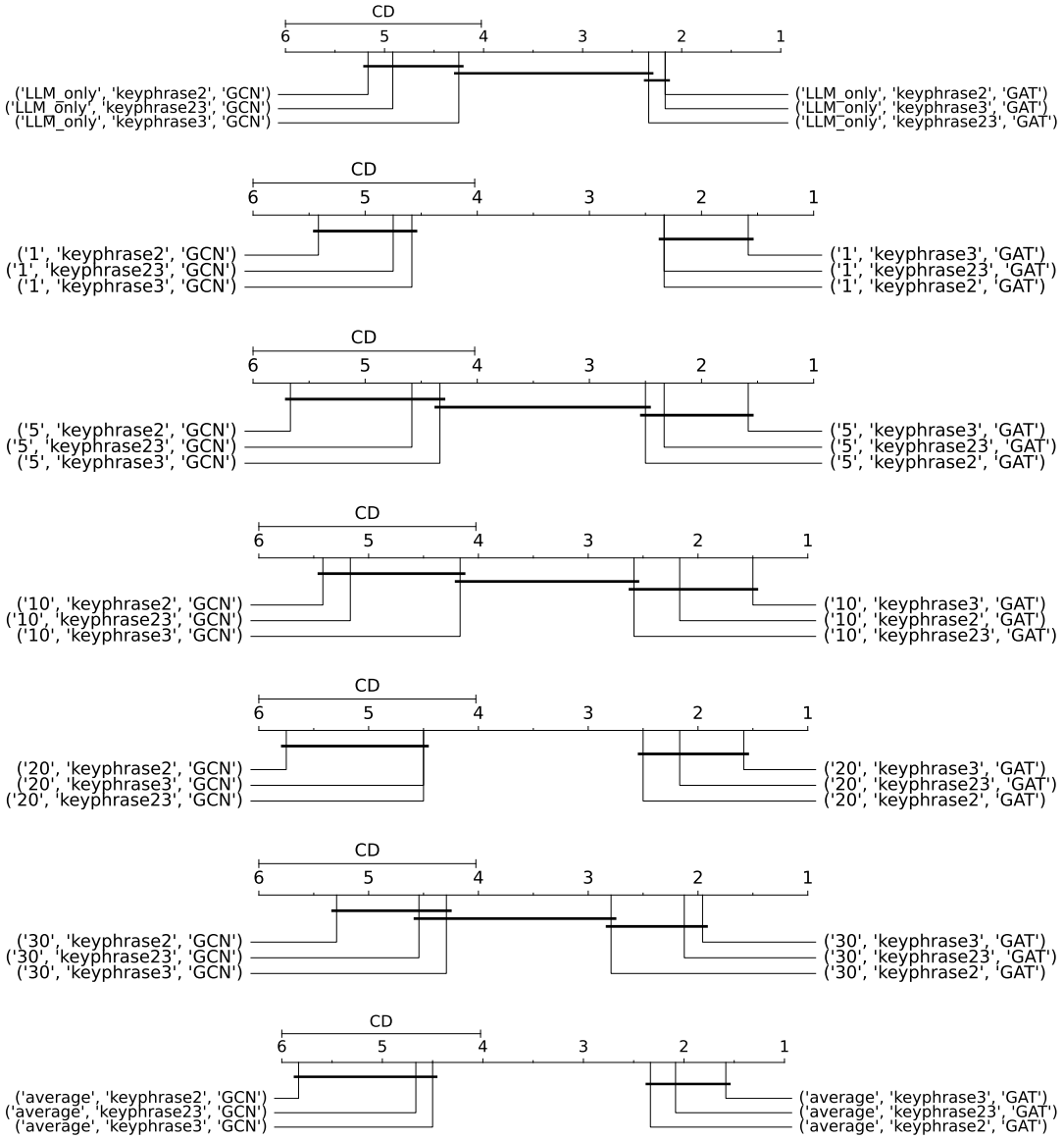


Figure 5.14: LLM 10 critical difference plots.

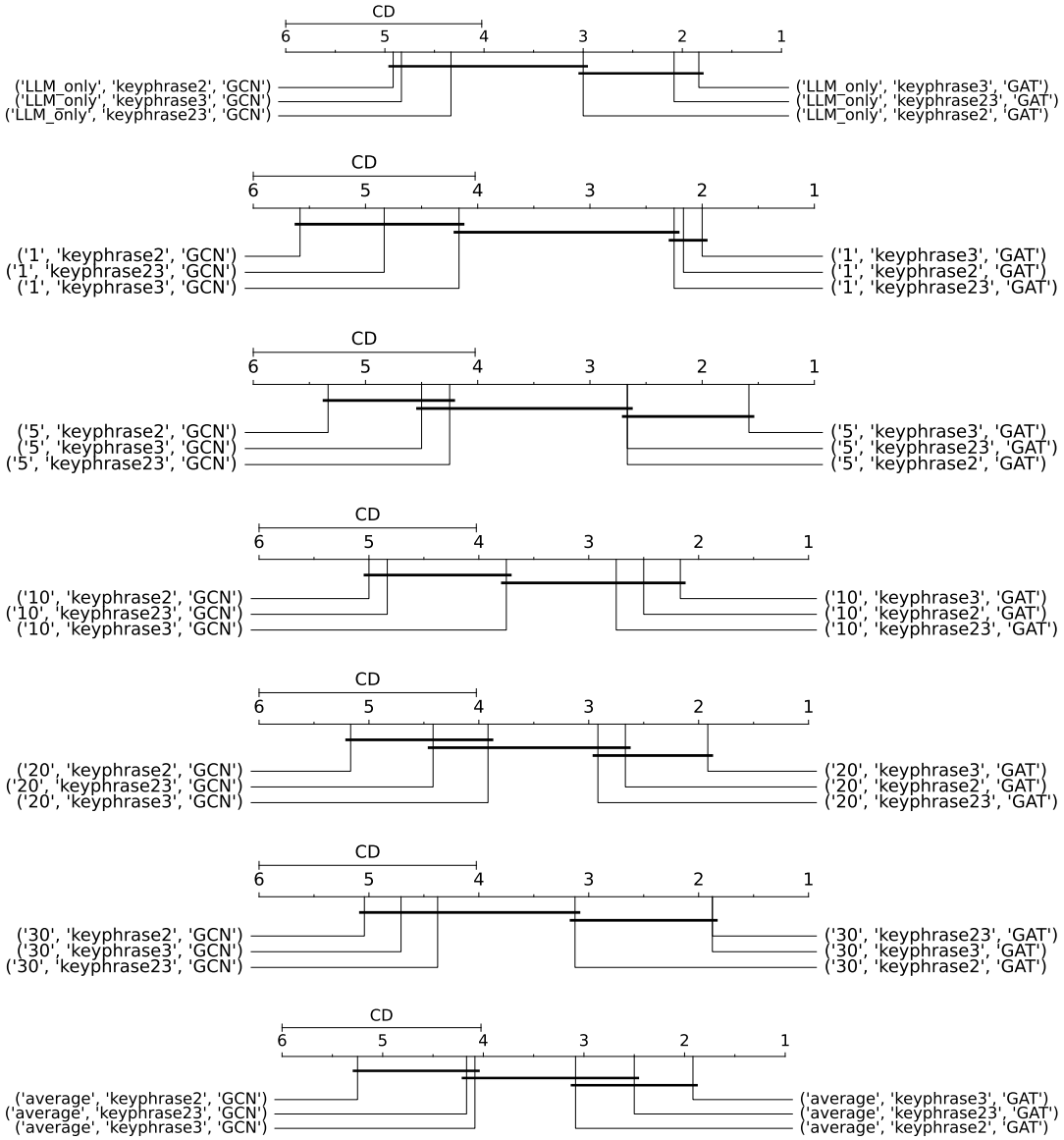


Figure 5.15: LLM 50 critical difference plots.

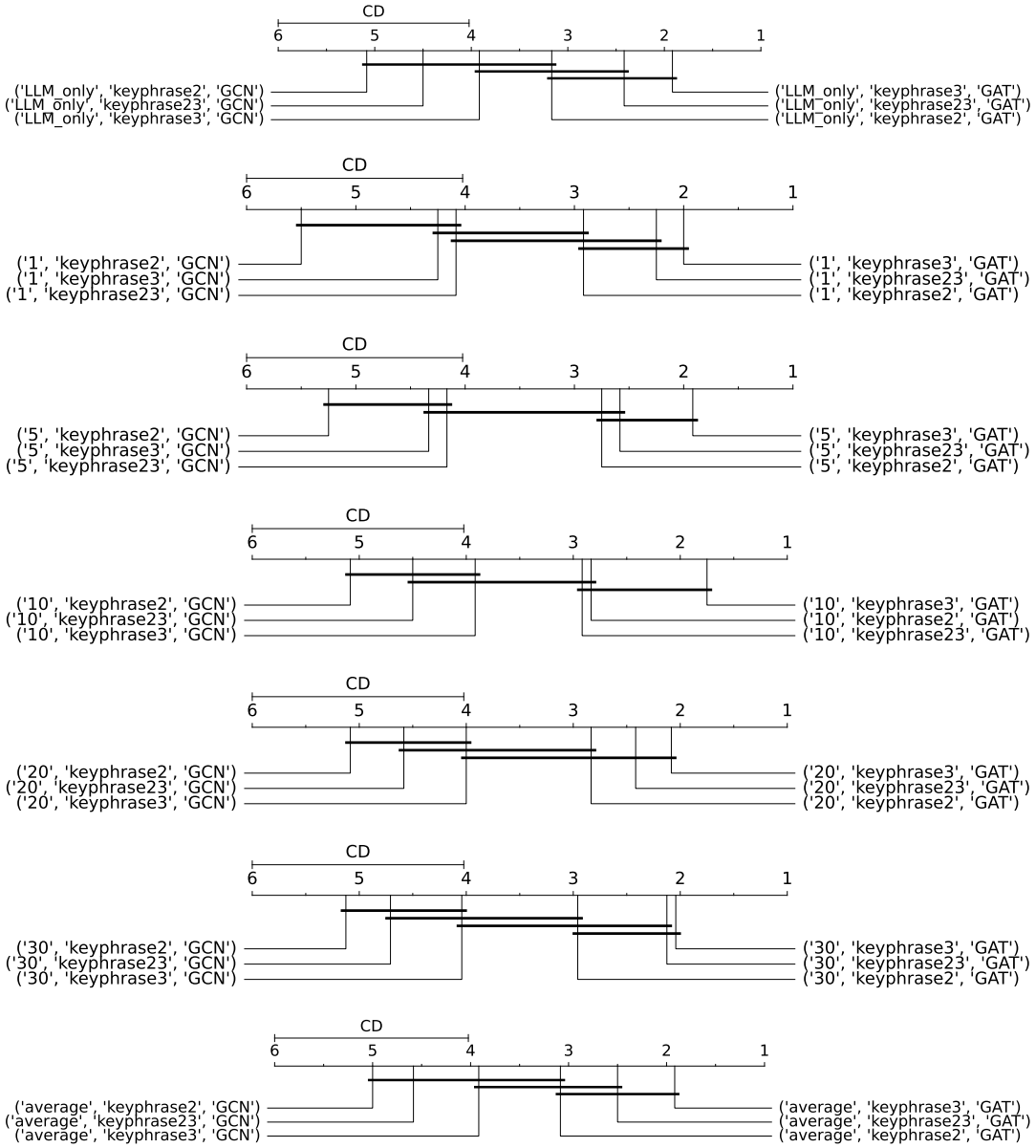


Figure 5.16: LLM 100 critical difference plots.

5.4 All models analysis

As a final analysis, Table 5.11 presents a complete comparison of all the models previously trained, aiming to find the best-performing models across all datasets and considering the use of $1, 5, 10, 20, 30$ human-labeled data. It compares keyphrase numbers: $[2], [2, 3], [3]$, GNN models: GAT and GCN, and also the techniques used on each model training: traditional modeling, coarsening, LLM-10, LLM-50 and LLM-100. Furthermore, Table 5.10 consolidates the results of this comparison, effectively counting the number of occurrences

each keyphrase, GNN Model, and technique were superior and trained the best models based on Table 5.11.

Keyphrase	**Number of best models**
[3]	**35**
[2]	14
[2,3]	10

GNN Model	**Number of best models**
GAT	**58**
GCN	1

Technique	**Number of best models**
Traditional Model	**37**
LLM-100	17
LLM-50	5
LLM-10	0
Coarsening	0

Table 5.10: Ranking tables, counting the number of occurrences each keyphrase, GNN Model, and technique were superior and trained the best models based on Table 5.11.

Considering keyphrase sizes, it is observable that $keyphrase = [3]$ achieved superior results in most cases, precisely 35 cases of Table 5.11, yet other keyphrase sizes also made appearances, especially when using LLM-labeled data, with $keyphrase = [2]$ achieving 14 best models, and $keyphrase = [2, 3]$ achieving 10.

For GNN models, as pointed out by all previous critical difference diagrams on Figures 5.2, 5.6, 5.14, 5.15, and 5.16, GAT achieved superior results almost unanimously, being the superior model on 58 cases, while GCN managed to surparss GAT only in 1 case, the case of CSTR dataset using one labled-dada for training.

Finally, regarding training techniques, traditional modeling was superior as expected, but LLM-labeling managed to support the initial training cases that had a lower amount of human-labeled data. Traditional modeling trained a total of 37 best models across all datasets and the number of labeled data. Although coarsening was able to preserve an average 83% of the traditional model's performance, no coarsening model was superior to its traditional counterpart, as pointed out by 5.2 and Table 5.5, and therefore, the coarsening technique trained no best model whatsoever. LLM-10 also were not capable of training a best model, however, LLM-50 and LLM-100 trained 5 and 17 best models respectively, with the vast majority being on the initial case with 1 and 5 human-labeled data.

Dataset	Number of Labeled Data	Best Performing Model			
		Keyphrases	GNN Model	Technique	Score:
CSTR	**1**	[2,3]	GCN	LLM-100	**0.7915 ± 0.014**
CSTR	**5**	[3]	GAT	Traditional Model	**0.828 ± 0.046**
CSTR	**10**	[2]	GAT	Traditional Model	**0.8474 ± 0.02**
CSTR	**20**	[2]	GAT	Traditional Model	**0.816 ± 0.031**
Dmoz_Computers	**1**	[3]	GAT	LLM-50	**0.4599 ± 0.007**
Dmoz_Computers	**5**	[3]	GAT	Traditional Model	**0.502 ± 0.011**
Dmoz_Computers	**10**	[3]	GAT	Traditional Model	**0.5614 ± 0.013**
Dmoz_Computers	**20**	[3]	GAT	Traditional Model	**0.5996 ± 0.008**
Dmoz_Computers	**30**	[3]	GAT	Traditional Model	**0.6193 ± 0.007**
Dmoz_Health	**1**	[2,3]	GAT	LLM-50	**0.7215 ± 0.018**
Dmoz_Health	**5**	[3]	GAT	LLM-100	**0.7251 ± 0.015**
Dmoz_Health	**10**	[3]	GAT	Traditional Model	**0.7739 ± 0.013**
Dmoz_Health	**20**	[2,3]	GAT	Traditional Model	**0.7996 ± 0.01**
Dmoz_Health	**30**	[2,3]	GAT	Traditional Model	**0.8114 ± 0.006**
Dmoz_Science	**1**	[3]	GAT	LLM-50	**0.6533 ± 0.013**
Dmoz_Science	**5**	[2]	GAT	LLM-100	**0.6509 ± 0.016**
Dmoz_Science	**10**	[3]	GAT	LLM-100	**0.6662 ± 0.02**
Dmoz_Science	**20**	[3]	GAT	Traditional Model	**0.7016 ± 0.01**
Dmoz_Science	**30**	[3]	GAT	Traditional Model	**0.7211 ± 0.008**
Dmoz_Sports	**1**	[3]	GAT	LLM-100	**0.8013 ± 0.005**
Dmoz_Sports	**5**	[3]	GAT	LLM-100	**0.8026 ± 0.005**
Dmoz_Sports	**10**	[3]	GAT	LLM-100	**0.8028 ± 0.007**
Dmoz_Sports	**20**	[3]	GAT	Traditional Model	**0.8158 ± 0.006**
Dmoz_Sports	**30**	[3]	GAT	Traditional Model	**0.8275 ± 0.004**
Industry_Sector	**1**	[2,3]	GAT	LLM-100	**0.3624 ± 0.008**
Industry_Sector	**5**	[2,3]	GAT	LLM-100	**0.3686 ± 0.01**
Industry_Sector	**10**	[2,3]	GAT	Traditional Model	**0.3901 ± 0.022**
Industry_Sector	**20**	[3]	GAT	Traditional Model	**0.4536 ± 0.015**
Industry_Sector	**30**	[3]	GAT	Traditional Model	**0.4827 ± 0.013**
NSF	**1**	[3]	GAT	LLM-100	**0.6198 ± 0.038**
NSF	**5**	[3]	GAT	Traditional Model	**0.7434 ± 0.016**
NSF	**10**	[3]	GAT	Traditional Model	**0.783 ± 0.009**
NSF	**20**	[3]	GAT	Traditional Model	**0.7876 ± 0.012**
NSF	**30**	[3]	GAT	Traditional Model	**0.8072 ± 0.006**
SyskillWebert	**1**	[2]	GAT	Traditional Model	**0.6547 ± 0.08**
SyskillWebert	**5**	[2]	GAT	Traditional Model	**0.8557 ± 0.033**
SyskillWebert	**10**	[3]	GAT	Traditional Model	**0.8894 ± 0.014**
SyskillWebert	**20**	[2]	GAT	Traditional Model	**0.9126 ± 0.019**
SyskillWebert	**30**	[2]	GAT	Traditional Model	**0.9044 ± 0.022**
classic4	**1**	[2,3]	GAT	LLM-50	**0.8058 ± 0.075**
classic4	**5**	[2]	GAT	Traditional Model	**0.896 ± 0.044**
classic4	**10**	[2]	GAT	Traditional Model	**0.9263 ± 0.012**
classic4	**20**	[3]	GAT	Traditional Model	**0.9405 ± 0.014**
classic4	**30**	[3]	GAT	Traditional Model	**0.9419 ± 0.013**
re8	**1**	[2,3]	GAT	LLM-100	**0.7481 ± 0.07**
re8	**5**	[3]	GAT	Traditional Model	**0.7561 ± 0.034**
re8	**10**	[3]	GAT	Traditional Model	**0.8112 ± 0.03**
re8	**20**	[3]	GAT	Traditional Model	**0.8304 ± 0.03**
re8	**30**	[2]	GAT	Traditional Model	**0.8208 ± 0.025**
review_polarity	**1**	[2]	GAT	LLM-100	**0.6288 ± 0.104**
review_polarity	**5**	[3]	GAT	LLM-50	**0.6176 ± 0.017**
review_polarity	**10**	[3]	GAT	LLM-100	**0.5879 ± 0.117**
review_polarity	**20**	[2]	GAT	LLM-100	**0.5969 ± 0.119**
review_polarity	**30**	[2,3]	GAT	LLM-100	**0.6199 ± 0.103**
webkb_parsed	**1**	[3]	GAT	LLM-100	**0.3378 ± 0.01**
webkb_parsed	**5**	[2]	GAT	LLM-100	**0.3447 ± 0.012**
webkb_parsed	**10**	[2]	GAT	Traditional Model	**0.374 ± 0.02**
webkb_parsed	**20**	[3]	GAT	Traditional Model	**0.416 ± 0.012**
webkb_parsed	**30**	[3]	GAT	Traditional Model	**0.446 ± 0.017**

Table 5.11: Table shows the best models trained across all datasets and number of human labeled data, considering all the models trained with all the previously presented techniques: Traditional GNN modeling, Coarsening, LLM-10, LLM-50, LLM-100, keyphrase numbers: $[2], [2,3], [3]$, and GNN models: GAT and GCN.

Chapter 6

Conclusion

Our study aimed to address the high labeling and computational costs often associated with natural language processing tasks, specifically in the domain of text classification. Hypothesizing that document-concept bipartite graph neural networks could be effectively leveraged for semi-supervised transductive learning, we explored methods to reduce dependency on human-labeled data and computational power. The research objectives were to evaluate the performance of Graph Attention Networks and Graph Convolutional Networks in a document-concept framework, apply graph coarsening techniques to optimize model efficiency, and assess the use of Large Language Models as an alternative for data labeling.

Our findings indicate that Graph Attention Networks (GAT) generally outperform Graph Convolutional Networks (GCN) across different datasets, as evidenced by higher F1-scores and greater robustness to variations in the number of labeled nodes. This supports the hypothesis that combining document-concept bipartite graphs and the GAT model's self-attention mechanism is particularly advantageous in capturing and differentiating meaningful relationships. Concepts (i.e., keyphrases) quantity also influenced results, with models trained on keyphrases containing three words often showing improved classification accuracy with richer semantic relationships, meaning that bigger keyphrases can improve performance.

In applying coarsening, the experiments demonstrated that this method could achieve a substantial reduction in graph size, averaging around 40-50% reduction in nodes and edges, without drastically compromising model performance. The coarsened models displayed a moderate decline in F1 scores, maintaining, on average, 83% of the original graph performance. The results suggest that coarsening is a viable approach for scaling graph-based models without compromising model accuracy, particularly for larger and more balanced datasets where minor performance reductions may be an acceptable trade-off for significant gains in efficiency.

The integration of LLM-labeled data with human-labeled instances provided a nuanced picture of LLM's effectiveness as a low-cost labeling method. Several efficient models were trained using LLM-only-labeled data, but most were easily outperformed by human-only models. Also, LLM data initially enhanced model performance when combined with human annotations, particularly in small quantities of human-labeled data. However, this benefit diminished as the quantity of human-labeled data increased, ultimately turning detrimental. This suggests that while LLMs can supplement early-stage labeling to improve performance, their utility decreases in the presence of more extensive human-labeled data, this is especially true on transductive models that do not require many labeled instances to perform, valuing a small quantity of very accurate data relative to many moderately accurate data. However, the accuracy of LLM-labeled data varied across datasets, indicating that LLMs may require dataset-specific calibration and can be very performative, closer to human labeling, on easier/well-defined classification tasks.

As per limitations, our study mainly focuses on GAT which demonstrated strong performance, it is computationally more intensive than GCN, and coarsening, our alternative to reducing computational costs, was not capable of maintaining GAT performance as well as when compared to GCN. Future research could explore optimizing GAT's efficiency or experimenting with GAT variants or other graph neural networks, such as GraphSage (Graph Sample and Aggregation) [Hamilton et al., 2017], GATv2 [Brody et al., 2021] or further exploring multi-head attention [Vaswani et al., 2017a] that could prove beneficial by parallelizing several attention heads each node. Additionally, while coarsening achieved efficiency gains to address its performance loss, refining coarsening techniques or experimenting with other coarsening methods could improve this trade-off.

Lastly, LLM-based labeling restrictive behavior as more human-labeled data was added demonstrates that current LLMs can complement human efforts, but they cannot fully substitute human labeling, especially on complex textual structures. In our implementations, we employed similarly designed prompts across all twelve datasets assessed, however, future research could focus on further optimizing LLM prompts to improve labeling accuracy across specific datasets. Most of all, we employ Llama 3.1 8B, a general-purpose LLM, as our labeler, as LLMs become more advanced and accessible, a more potent model, with a lager number of parameters (e.g., Llama 3.1 70B), and a fine-tuned text classification LLM could significantly enhance results, especially on harder and more challenging labeling tasks.

References

[Ainslie et al., 2023] Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. (2023). Gqa: Training generalized multi-query transformer models from multi-head checkpoints. 23

[Allan, 2002] Allan, J. (2002). *Topic detection and tracking: event-based information organization*, volume 12. Springer Science & Business Media. 25

[Bahdanau et al., 2016] Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate. 15

[Bakhvalov, 1966] Bakhvalov, N. (1966). On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Computational Mathematics and Mathematical Physics*, 6(5):101–135. 18

[Beliga, 2014] Beliga, S. (2014). Keyword extraction: a review of methods and approaches. *University of Rijeka, Department of Informatics, Rijeka*, 1(9):1–9. 9, 25

[Bennani-Smires et al., 2018] Bennani-Smires, K., Musat, C., Hossmann, A., Baeriswyl, M., and Jaggi, M. (2018). Simple unsupervised keyphrase extraction using sentence embeddings. 10, 26

[Bird et al., 2009] Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly, Beijing. 27

[Blei et al., 2003a] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003a). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022. 25

[Blei et al., 2003b] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003b). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022. 27

[Brandt, 1977] Brandt, A. (1977). Multi-level adaptive solutions to boundary-value problems. *Mathematics of computation*, 31(138):333–390. 18

[Brody et al., 2021] Brody, S., Alon, U., and Yahav, E. (2021). How attentive are graph attention networks? *CoRR*, abs/2105.14491. 28, 83

[Cai et al., 2021] Cai, C., Wang, D., and Wang, Y. (2021). Graph coarsening with neural networks. 17

[Cain, 2023] Cain, W. (2023). Prompting change: Exploring prompt engineering in large language model ai and its potential to transform education. *TechTrends*, 68. 22

[Chen et al., 2021] Chen, J., Saad, Y., and Zhang, Z. (2021). Graph coarsening: From scientific computing to machine learning. 18

[Chen et al., 2023a] Chen, Y., Wang, X., and Xu, G. (2023a). Gatgpt: A pre-trained large language model with graph attention network for spatiotemporal imputation. 30

[Chen et al., 2023b] Chen, Z., Mao, H., Wen, H., Han, H., Jin, W., Zhang, H., Liu, H., and Tang, J. (2023b). Label-free node classification on graphs with large language models (llms). 30

[Ciano et al., 2022] Ciano, G., Rossi, A., Bianchini, M., and Scarselli, F. (2022). On inductive–transductive learning with graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(2):758–769. 12

[Cunningham and Delany, 2021] Cunningham, P. and Delany, S. J. (2021). k-nearest neighbour classifiers - a tutorial. *ACM Computing Surveys*, 54(6):1–25. 35, 52

[de Paulo Faleiros et al., 2017] de Paulo Faleiros, T., Geraldeli Rossi, R., and de Andrade Lopes, A. (2017). Optimizing the class information divergence for transductive classification of texts using propagation in bipartite graphs. *Pattern Recognition Letters*, 87:127–138. Advances in Graph-based Pattern Recognition. 6, 7

[de Souza et al., 2024] de Souza, M. C., Gôlo, M. P. S., Jorge, A. M. G., de Amorim, E. C. F., Campos, R. N. T., Marcacini, R. M., and Rezende, S. O. (2024). Keywords attention for fake news detection using few positive labels. *Information Sciences*, 663:120300. 26

[Demšar, 2006] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30. 49

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. 22

[Ding et al., 2018] Ding, M., Tang, J., and Zhang, J. (2018). Semi-supervised learning on graphs with generative adversarial nets. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 913–922. ACM. 26

[dos Santos et al., 2024] dos Santos, N. R., Minatel, D., Valejo, A. D. B., and de A. Lopes, A. (2024). Bipartite graph coarsening for text classification using graph neural networks. In Vasconcelos, V., Domingues, I., and Paredes, S., editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 589–604, Cham. Springer Nature Switzerland. 29

[Dubey et al., 2024] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D.,

Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Tan, X. E., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Grattafiori, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Vaughan, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Franco, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Wyatt, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Ozgenel, F., Caggioni, F., Guzmán, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Thattai, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Damlaj, I., Molybog, I., Tufanov, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Prasad, K., Khandelwal, K., Zand, K.,

Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Huang, K., Chawla, K., Lakhotia, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Tsimpoukelli, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Laptev, N. P., Dong, N., Zhang, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Li, R., Hogan, R., Battey, R., Wang, R., Maheswari, R., Howes, R., Rinott, R., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Kohler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Albiero, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wang, X., Wu, X., Wang, X., Xia, X., Wu, X., Gao, X., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Hao, Y., Qian, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., and Zhao, Z. (2024). The llama 3 herd of models. 22, 42

[Eduardo Althoff et al., 2023] Eduardo Althoff, P., Demétrius Baria Valejo, A., and de Paulo Faleiros, T. (2023). Coarsening effects on k-partite network classification. *Applied Network Science*, 8(1):82. xi, 18, 21, 28, 29, 41

[Faleiros et al., 2017] Faleiros, T. d. P., Rossi, R. G., and Lopes, A. d. A. (2017). Optimizing the class information divergence for transductive classification of texts using propagation in bipartite graphs. *Pattern Recognit. Lett.*, 87:127–138. 6

[Fan et al., 2023] Fan, L., Li, L., Ma, Z., Lee, S., Yu, H., and Hemphill, L. (2023). A bibliometric review of large language models research from 2017 to 2023. 22

[Gammerman et al., 2013] Gammerman, A., Vovk, V., and Vapnik, V. (2013). Learning by transduction. 6

[Gori et al., 2005] Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, 2:729–734 vol. 2. 11, 26

[Goutte and Gaussier, 2005] Goutte, C. and Gaussier, E. (2005). A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. volume 3408, pages 345–359. 49

[Grootendorst, 2020] Grootendorst, M. (2020). Keybert: Minimal keyword extraction with bert. 10, 26, 35

[Gôlo and Marcacini, 2023] Gôlo, M. and Marcacini, R. (2023). Text representation through multimodal variational autoencoder for one-class learning. In *Anais Estendidos do XXIX Simpósio Brasileiro de Sistemas Multimídia e Web*, pages 19–22, Porto Alegre, RS, Brasil. SBC. 26

[Hamilton et al., 2017] Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Inductive representation learning on large graphs. *CoRR*, abs/1706.02216. 83

[Huang et al., 2021] Huang, Z., Zhang, S., Xi, C., Liu, T., and Zhou, M. (2021). Scaling up graph neural networks via graph coarsening. 29

[Janiesch et al., 2021] Janiesch, C., Zschech, P., and Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, 31(3):685–695. 1

[Khemani et al., 2024] Khemani, B., Patil, S., Kotecha, K., and Tanwar, S. (2024). A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. *Journal of Big Data*, 11. xi, 11, 12, 26

[Kipf and Welling, 2016a] Kipf, T. N. and Welling, M. (2016a). Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907. 12, 13

[Kipf and Welling, 2016b] Kipf, T. N. and Welling, M. (2016b). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*. 36

[Kong et al., 2013] Kong, X., Ng, M. K., and Zhou, Z.-H. (2013). Transductive multilabel learning via label set propagation. *IEEE Transactions on Knowledge and Data Engineering*, 25(3):704–719. 6

[Kumar et al., 2022] Kumar, M., Sharma, A., and Kumar, S. (2022). A unified framework for optimization-based graph coarsening. 17

[Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324. 12

[Lee et al., 2023] Lee, D.-H., Pujara, J., Sewak, M., White, R. W., and Jauhar, S. K. (2023). Making large language models better data creators. 29

[Li et al., 2022] Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P. S., and He, L. (2022). A survey on text classification: From traditional to deep learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(2):1–41. 2

[Li et al., 2023] Li, W., Xue, J., Zhang, X., Chen, H., Chen, Z., Huang, F., and Cai, Y. (2023). Word-graph2vec: An efficient word embedding approach on word co-occurrence graph using random walk technique. 3

[Lin et al., 2017] Lin, Z., Feng, M., dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. 15

[Linmei et al., 2019] Linmei, H., Yang, T., Shi, C., Ji, H., and Li, X. (2019). Heterogeneous graph attention networks for semi-supervised short text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4821–4830, Hong Kong, China. Association for Computational Linguistics. 27

[Liu, 2023] Liu, Y. (2023). The importance of human-labeled data in the era of llms. 29

[Liu et al., 2018] Liu, Y., Safavi, T., Dighe, A., and Koutra, D. (2018). Graph summarization methods and applications: A survey. *ACM computing surveys (CSUR)*, 51(3):1–34. 18

[Meng et al., 2017] Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., and Chi, Y. (2017). Deep keyphrase generation. In Barzilay, R. and Kan, M.-Y., editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592, Vancouver, Canada. Association for Computational Linguistics. 26

[Mihalcea and Tarau, 2004] Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411. 10, 25

[Naveed et al., 2023] Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., and Mian, A. (2023). A comprehensive overview of large language models. 21

[Neogi et al., 2020] Neogi, P. P. G., Das, A. K., Goswami, S., and Mustafi, J. (2020). Topic modeling for text classification. In *Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018*, pages 395–407, India. Springer. 26

[Niepert et al., 2016] Niepert, M., Ahmed, M., and Kutzkov, K. (2016). Learning convolutional neural networks for graphs. 27

[Nomoto, 2022] Nomoto, T. (2022). Keyword extraction: A modern perspective. *SN Computer Science*, 4. 25

[OpenAI et al., 2024] OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo,

Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, J. H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorny, Pokrass, M., Pong, V. H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M. B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. (2024). Gpt-4 technical report. 22

[Otter et al., 2019] Otter, D. W., Medina, J. R., and Kalita, J. K. (2019). A survey of the usages of deep learning in natural language processing. 1

[Page et al., 1998] Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project. 27

[Papagiannopoulou and Tsoumakas, 2019] Papagiannopoulou, E. and Tsoumakas, G. (2019). A review of keyphrase extraction. 9, 25

[Piccinno and Ferragina, 2014] Piccinno, F. and Ferragina, P. (2014). From tagme to wat: A new entity annotator. In *Proceedings of the First International Workshop on Entity Recognition & Disambiguation*, ERD '14, page 55–62, New York, NY, USA. Association for Computing Machinery. 27

[Platt, 1998] Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research Technical Report. 51

[Quinlan, 2014] Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier. 51

[Raffel et al., 2023] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2023). Exploring the limits of transfer learning with a unified text-to-text transformer. 22

[Rahman et al., 2018] Rahman, S., Khan, S., and Porikli, F. (2018). A unified approach for conventional zero-shot, generalized zero-shot, and few-shot learning. *IEEE Transactions on Image Processing*, 27(11):5652–5667. 22

[Ramachandran et al., 2019] Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., and Shlens, J. (2019). Stand-alone self-attention in vision models. 15

[Ren et al., 2024] Ren, X., Tang, J., Yin, D., Chawla, N., and Huang, C. (2024). A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 6616–6626. ACM. 30

[RK Rao and Devi, 2017] RK Rao, P. and Devi, S. L. (2017). Patent document summarization using conceptual graphs. *International Journal on Natural Language Computing (IJNLC) Vol*, 6. 9

[Rossi et al., 2013] Rossi, R. G., Marcacini, R. M., and Rezende, S. O. (2013). Benchmarking text collections for classification and clustering tasks. Technical Report 395, Institute of Mathematics and Computer Sciences - University of Sao Paulo. xiii, 33, 51, 58

[Saifuddin et al., 2021] Saifuddin, K. M., Islam, M. I., and Akbas, E. (2021). Drug abuse detection in twitter-sphere: Graph-based approach. pages 4136–4145. 8

[Salton and Yang, 1973] Salton, G. and Yang, C.-S. (1973). On the specification of term values in automatic indexing. *Journal of documentation*, 29(4):351–372. 10, 25

[Sasaki, 2007] Sasaki, Y. (2007). The truth of the f-measure. *Teach Tutor Mater*. 48

[Scarselli et al., 2008] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80. 11

[Sebastiani, 2002] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47. 1

[Seol et al., 2023] Seol, D., Choi, J., Kim, C., and Hong, S. (2023). Alleviating class-imbalance data of semiconductor equipment anomaly detection study. *Electronics*, 12:585. xi, 49

[Sharma and Li, 2019] Sharma, P. and Li, Y. (2019). Self-supervised contextual keyword and keyphrase retrieval with self-labelling. 10, 26

[Shazeer, 2020] Shazeer, N. (2020). Glu variants improve transformer. 23

[Siddiqi and Sharan, 2015] Siddiqi, S. and Sharan, A. (2015). Keyword and keyphrase extraction techniques: A literature review. *International Journal of Computer Applications*, 109:18–23. 9, 25

[Su et al., 2023] Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. (2023). Roformer: Enhanced transformer with rotary position embedding. 23

[Sun et al., 2022] Sun, Y., Zhu, D., Du, H., and Tian, Z. (2022). Mhnf: Multi-hop heterogeneous neighborhood information fusion graph representation learning. 27

[Touvron et al., 2023a] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023a). Llama: Open and efficient foundation language models. 23

[Touvron et al., 2023b] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023b). Llama 2: Open foundation and fine-tuned chat models. 23

[Tripodi and Pelillo, 2017] Tripodi, R. and Pelillo, M. (2017). 6 - transductive learning games for word sense disambiguation. In Sharp, B., Sèdes, F., and Lubaszewski, W., editors, *Cognitive Approach to Natural Language Processing*, pages 109–128. Elsevier. 6

[Valejo et al., 2021a] Valejo, A., Althoff, P., Faleiros, T., Chuerubim, M., Yan, J., Liu, W., and Zhao, L. (2021a). Coarsening algorithm via semi-synchronous label propagation for bipartite networks. In *Anais da X Brazilian Conference on Intelligent Systems*, Porto Alegre, RS, Brasil. SBC. 19, 28

[Valejo et al., 2020] Valejo, A., Ferreira, V., Fabbri, R., Oliveira, M. C. F. d., and Lopes, A. d. A. (2020). A critical survey of the multilevel method in complex networks. *ACM Comput. Surv.*, 53(2). 20

[Valejo et al., 2021b] Valejo, A. D. B., de Oliveira dos Santos, W., Naldi, M. C., and Zhao, L. (2021b). A review and comparative analysis of coarsening algorithms on bipartite networks. *The European Physical Journal Special Topics*, 230:2801–2811. 28

[Vaswani et al., 2017a] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017a). Attention is all you need. *CoRR*, abs/1706.03762. 15, 23, 24, 83

[Vaswani et al., 2017b] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017b). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. 27

[Veličković et al., 2017] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2017). Graph attention networks. *6th International Conference on Learning Representations*. 2, 15, 16, 26

[Vikramkumar et al., 2014] Vikramkumar, B, V., and Trilochan (2014). Bayes and naive bayes classifier. 51

[Waikhom and Patgiri, 2021] Waikhom, L. and Patgiri, R. (2021). Graph neural networks: Methods, applications, and opportunities. *CoRR*, abs/2108.10733. 12

[Walshaw, 2004] Walshaw, C. (2004). Multilevel refinement for combinatorial optimisation problems. *Annals of Operations Research*, 131:325–372. 18

[Wang et al., 2021] Wang, Y., Li, X., Zhou, X., and Ouyang, J. (2021). Extracting topics with simultaneous word co-occurrence and semantic correlation graphs: Neural topic modeling for short texts. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 18–27, Punta Cana, Dominican Republic. Association for Computational Linguistics. 3

[Weiss et al., 2012] Weiss, S. M., Indurkhya, N., and Zhang, T. (2012). *Fundamentals of Predictive Text Mining*. Springer. 1

[Wu et al., 2019] Wu, F., Zhang, T., de Souza Jr. au2, A. H., Fifty, C., Yu, T., and Weinberger, K. Q. (2019). Simplifying graph convolutional networks. 27

[Wu et al., 2023] Wu, L., Chen, Y., Shen, K., Guo, X., Gao, H., Li, S., Pei, J., Long, B., et al. (2023). Graph neural networks for natural language processing: A survey. *Foundations and Trends® in Machine Learning*, 16(2):119–328. 36

[Xie et al., 2021] Xie, Q., Huang, J., Du, P., Peng, M., and Nie, J.-Y. (2021). Inductive topic variational graph auto-encoder for text classification. In *proceedings of the 2021 conference of the North American chapter of the Association for Computational Linguistics: human language technologies*, pages 4218–4227, Online. ACL. 26

[Xu et al., 2017] Xu, S., Li, Y., and Zheng, W. (2017). Bayesian multinomial naïve bayes classifier to text classification. pages 347–352. 51

[Yao et al., 2018] Yao, L., Mao, C., and Luo, Y. (2018). Graph convolutional networks for text classification. 2, 27

[Zhang and Sennrich, 2019] Zhang, B. and Sennrich, R. (2019). Root mean square layer normalization. 23

[Zhang and Zhang, 2020] Zhang, H. and Zhang, J. (2020). Text graph transformer for document classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8322–8327, Online. Association for Computational Linguistics. 2, 27

[Zhang and Meng, 2019] Zhang, J. and Meng, L. (2019). Gresnet: Graph residual network for reviving deep gnns from suspended animation. 27

[Zhang et al., 2011] Zhang, L., Li, C., Liu, J., and Wang, H. (2011). Graph-based text similarity measurement by exploiting wikipedia as background knowledge. *World Academy of Science, Engineering and Technology*, 59:1548–1553. 9

[Zhang et al., 2016] Zhang, Q., Wang, Y., Gong, Y., and Huang, X.-J. (2016). Keyphrase extraction using deep recurrent neural networks on twitter. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 836–845. 26

[Zhang et al., 2021] Zhang, S., Jafari, O., and Nagarkar, P. (2021). A survey on machine learning techniques for auto labeling of video, audio, and text data. 2

[Zhao et al., 2023] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., and Wen, J.-R. (2023). A survey of large language models. 22

[Zhu and Koniusz, 2021] Zhu, H. and Koniusz, P. (2021). Simple spectral graph convolution. In *International Conference on Learning Representations*. 27