Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Chen Ling Date

Harnessing networked, textual data with graph and language modeling

By

Chen Ling
Doctor of Philosophy Candidate

Computer Science and Informatics

Liang Zhao, Ph.D.
Advisor

Carl Yang, Ph.D.
Committee Member

Andreas Züfle, Ph.D.
Committee Member

Xujiang Zhao, Ph.D.
Committee Member

Accepted:

Kimberly Jacob Arriola, Ph.D.
Dean of the James T. Laney School of Graduate Studies

Date

Harnessing networked, textual data with graph and language modeling

By

Chen Ling
B.S., University of Vermont, VT, 2018
M.S., University of Delaware, DE, 2020

Advisor: Liang Zhao, Ph.D.

An abstract of
A Dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy Candidate
in Computer Science and Informatics
2024

Abstract

Harnessing networked, textual data with graph and language modeling
By Chen Ling

Networked, textual data are ubiquitous across various domains, playing a critical role in numerous applications from social network analysis to knowledge representation. These two types of data can naturally be combined, as text data can be structured like a graph, and graph data can have rich text embedded on nodes and edges. For instance, a document collection can be represented as a graph where nodes are documents and edges indicate relationships such as citations or references. Conversely, social networks often contain textual information in the form of user profiles, posts, and interactions embedded on nodes and edges. Existing works have primarily focused on either graph or textual data in isolation, often overlooking the potential synergy between the two. Combining both modalities to address unique research problems that require a holistic understanding of structural relationships and semantic content is important. Integrating these modalities can leverage the complementary strengths of graph data's structural insights and textual data's semantic richness, leading to more robust and comprehensive data mining methodologies.

Despite the advancements in graph data mining and language modeling, existing approaches that treat graph and textual data separately can introduce significant limitations. While many graph neural networks do integrate semantic information from node and edge attributes, there remain graph data mining problems—*such as those addressing graph inverse problems or combinatorial optimization*—that predominantly rely on graph topology and information flow for making predictions or approximations. Conversely, language models that overlook the structural context provided by graphs may lack a framework for accurately interpreting and generating text, especially in tasks requiring a deep understanding of relationships and dependencies. This dichotomy motivates the need to bridge graph and textual data, leveraging their complementary strengths for more effective data mining.

There are three key challenges to addressing this integration. First, preserving both graph and textual information in a unified representation is challenging. This process involves creating embeddings that maintain the structural integrity of graphs while encapsulating the semantic richness of texts. Second, enhancing graph-based tasks with textual information is crucial. For instance, in source localization of information diffusion on information networks, incorporating node texts can provide additional context that improves the accuracy of identifying information diffusion sources. Third, utilizing graph structures to augment text-based tasks, such as knowledge-extensive question answering, is also essential. For example, knowledge graphs can provide a structured context that enhances the reasoning capabilities of language models. This dissertation proposal is dedicated to exploring these challenges, particularly focusing on applications of 1) integrating textual data for solving graph data mining problems like source localization of information diffusion and 2) employing graph-structured data to improve the reasoning capability of language models.

Specifically, this dissertation focuses on three primary areas: 1) learning latent embeddings that fuse both semantic and structural information of the observed network to facilitate downstream graph data mining tasks, e.g., deep graph generation, source localization, and influence maximization. 2) enhancing natural language understanding tasks by different means, such as quantifying the uncertainty of a large language model's response and employing an external knowledge base to enhance the commonsense reasoning task in a retrieval-augmented manner. 3) creating a framework that combines the semantic processing capabilities of large language models with the structural analysis strengths of graph neural networks to learn a unified representation for text-attributed graphs.

This dissertation's contributions include novel formulations and frameworks for each task, the creation of new datasets, and extensive experimental validation. This interdisciplinary approach advances the theoretical understanding of integrating graph data mining and language modeling and has practical implications for a wide range of applications in data science.

Harnessing networked, textual data with graph and language modeling

By

Chen Ling
B.S., University of Vermont, VT, 2018
M.S., University of Delaware, DE, 2020

Advisor: Liang Zhao, Ph.D.

A Dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy Candidate
in Computer Science and Informatics
2024

Acknowledgments

I would like to express my deepest gratitude to my advisor, Dr. Liang Zhao, for his invaluable guidance, unwavering support, and constant encouragement throughout my doctoral journey. His mentorship has been instrumental in shaping both my academic pursuits and personal growth. I am also profoundly grateful to my dissertation committee members, Dr. Carl Yang, Dr. Andreas Züfle, and Dr. Xujiang Zhao, for their insightful feedback, constructive criticism, and steadfast support. Their expertise and perspectives have significantly enhanced the quality of my research.

My sincere appreciation extends to my collaborators, both within and beyond my research group. I would like to particularly acknowledge Guangji Bai, Zheng Zhang, Yifei Zhang, Bo Pan, Yuntong Hu, Dr. Tanmoy Chowdhury, and Dr. Xuchao Zhang for their contributions, stimulating discussions, and camaraderie. Your collaboration has enriched my research experience and broadened my intellectual horizons.

I owe a debt of gratitude to my family, especially my parents for their unconditional love, patience, and support throughout this challenging yet rewarding journey. Your belief in me has been a constant source of strength and motivation.

A special thanks goes to my cousin, Dr. Yue Zhang, whose love, guidance, and support have been invaluable throughout my academic journey. Your example has inspired me to persevere and excel from the first day of my academic journey.

Finally, I reserve my deepest love and appreciation for my wife, Dr. Xinwen Zhang. Your unwavering support, understanding, and love have been my anchor through the ups and downs of this journey. We have weathered difficulties and celebrated joys together, and I look forward to the continued happiness that lies ahead. This achievement is as much yours as it is mine.

To all those mentioned and the many others who have contributed to my growth and success, I extend my heartfelt thanks. This dissertation stands as a testament to your collective support and encouragement.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In the rapidly evolving domain of data science, the ability to effectively analyze and interpret complex information from diverse sources has become increasingly crucial. Two significant modalities that have gained substantial attention in recent years are graph data and text data. Both modalities can naturally be combined, as text data can be structured like a graph, and graph data can have rich text embedded on nodes and edges. For instance, academic publications can form a citation network where each document is a node, and the edges represent citation relationships, with rich textual abstracts or keywords associated with each node. Similarly, shown in Figure 1.1, social media platforms not only connect users through friendship or follower networks but also embed extensive textual information in posts and profiles. Despite their intertwined nature, existing works have primarily focused on either graph data or textual data in isolation, often overlooking the potential synergy between the two. Combining both modalities is essential to solving unique research problems that require a holistic understanding of structural relationships and semantic content. This integration can harness the complementary strengths of graph data's structural insights and textual data's semantic richness, paving the way for innovative and robust data mining methodologies.

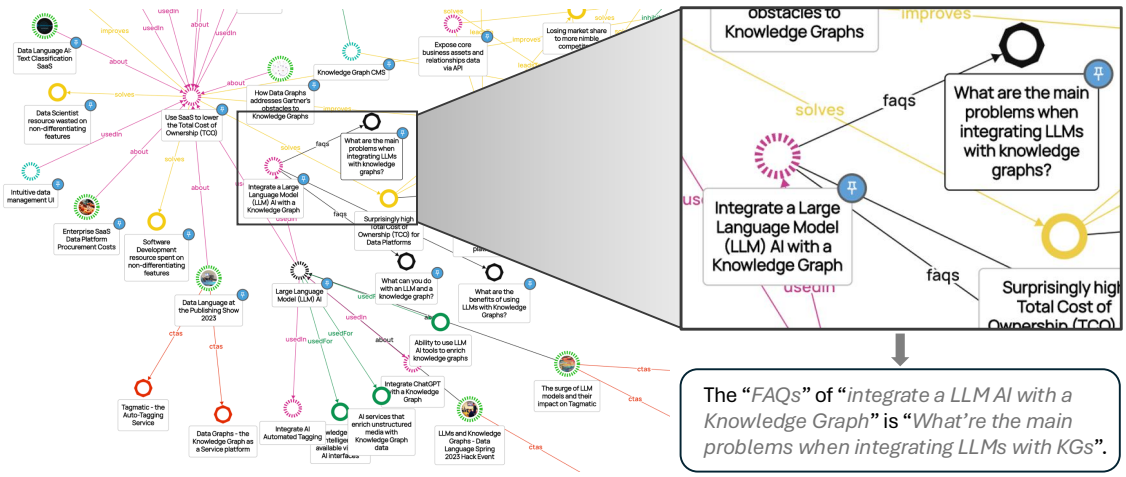


Figure 1.1: An example of networked, textual data, the local connection in an academic research graph can be interchangeably converted to natural language.

Traditionally, graph data mining has emphasized the structural aspects of data, such as the connections between nodes and the overall network topology. While this approach has been highly effective, certain methods may *underutilize or not explicitly leverage the rich textual information associated with nodes*, despite its potential to significantly enhance analysis. For example, in diffusion source localization within social networks, many conventional methods focus primarily on structural information, often not incorporating the textual content of communications associated with each node. However, graphs are often used to describe user mobility with textual data describing the semantics of the visited places [131, 130]. However, analyzing a node's movement/mobility history alongside its structural context can yield critical insights, complement traditional approaches, and enable a more comprehensive identification of rumor source nodes. By integrating textual features with structural patterns, researchers can gain a multidimensional understanding of the data, enriching the analysis and expanding the applicability of graph data mining in scenarios where textual content is crucial.

Similarly, the field of natural language processing (NLP) has increasingly recognized the value of incorporating structured knowledge into its frameworks. This

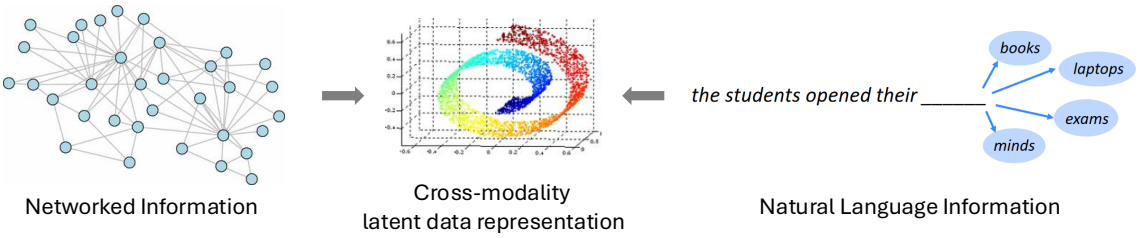


Figure 1.2: Integrating topology and semantic information to a unified representation.

recognition stems from the realization that language models, while powerful in understanding and generating text, can significantly benefit from external, structured contexts to make more informed decisions. Specifically, in tasks requiring factual accuracy, integrating external knowledge bases (e.g., knowledge graphs) presents a compelling solution. These graphs, constructed from key entities and their interrelations extracted from textual queries, offer a rich reservoir of structured knowledge that can guide language models in navigating complex reasoning pathways. By harnessing this structured knowledge, language models can transcend their inherent limitations, achieving a deeper comprehension of context and enhancing their decision-making capabilities in a way that mirrors human cognitive processes more closely.

As shown in Figure 1.2, integrating the modalities of graph data mining and language modeling into a cohesive analytical framework presents several technical challenges that necessitate innovative solutions. *The first challenge lies in preserving semantic richness in textual information while integrating it with structural graph data.* Traditional approaches often convert textual data into latent embeddings at an early stage, potentially stripping away nuanced meanings that could be crucial for decision-making. This highlights the need for maintaining the integrity of semantic information throughout the analysis process, ensuring that both textual and structural data are effectively utilized.

The second challenge involves enhancing graph-based tasks with textual information. For instance, solving graph problems such as source localization can be significantly improved by considering the textual content associated with nodes. This

integration can provide additional context and insights, leading to more accurate and robust solutions. Developing methodologies that can seamlessly incorporate textual data into graph mining tasks is essential for advancing this field.

The third challenge is leveraging graph structures to augment text-based tasks, such as question-answering. Knowledge graphs provide a structured context that can enhance the reasoning capabilities of language models, particularly in zero-shot scenarios where the model encounters unseen instances. Sophisticated methods are required to extract and represent this structured knowledge in a way that is understandable and useful for language models, enabling them to navigate complex reasoning tasks with improved accuracy and efficiency.

Addressing these challenges not only advances the theoretical understanding of integrating graph data mining and language modeling but also has practical implications for a wide range of applications in data science. This dissertation explores these challenges, focusing on the application of integrating textual data for solving graph problems like source localization and employing graph data to improve the reasoning capabilities of language models.

1.1 Research Issues

This thesis aims to explore the intersection of graph data mining and language modeling, with a specific focus on aligning both modalities to solve various data mining tasks, thereby harnessing the complementary strengths of these two modalities to enrich our understanding and processing of complex data.

1.1.1 Enhancing Graph Data Mining by Exploiting Semantic Information on Networks

The field of graph data mining has seen significant advancements through the consideration of semantic meaning in heterogeneous and complex networks. The first work introduces a novel framework (HGEN) for heterogeneous network generation that addresses the challenges of modeling local semantic distributions, preserving graph-structured distributions, and characterizing global graph patterns. This framework features a hierarchical heterogeneous walk generator and a graph assembler that constructs graphs by stratifying meta-path instances, offering a comprehensive approach to capturing semantic, structural, and global distributions in heterogeneous graphs.

The second work focuses on source localization in cross-networks, where information diffusion in one network depends on another interconnected network. The proposed method, CNSL, tackles the challenges of diffusion source distribution modeling, learning heterogeneous diffusion patterns, and handling both static and dynamic node features. CNSL employs Bayesian inference and disentangled encoders, enabling effective information diffusion source localization by considering the overall diffusion process across networks.

The third work addresses influence maximization in social networks, a problem that seeks to maximize influence spread by selecting optimal initial users. Traditional methods have reached a performance plateau, prompting the development of learning-based approaches. The proposed DeepIM framework generatively learns the latent representation of seed sets and adapts to various node-centrality constraints, offering a flexible, data-driven solution to the IM problem that overcomes the limitations of previous methods. Together, these works contribute to enhancing graph data mining by integrating semantic considerations into complex graph structures and processes.

1.1.2 Integrating Structured Knowledge and Quantifying Uncertainty in Natural Language Understanding

For natural language understanding, recent research has focused on integrating structured knowledge and addressing uncertainty to enhance the capabilities of AI systems. The first work explores open-ended commonsense reasoning, a challenging task that involves solving commonsense questions without predefined answer candidates or a limited answer scope. Traditional question-answering methods and retrieval-based approaches struggle in this open-ended setting due to the vast search space and the necessity for implicit multi-hop reasoning. To overcome these challenges, the paper proposes leveraging pre-trained language models to iteratively retrieve reasoning paths from external knowledge bases without task-specific supervision. These reasoning paths help pinpoint the most accurate answers, significantly improving performance on commonsense reasoning benchmarks compared to state-of-the-art methods.

The second work investigates the complexities of predictive uncertainty in the context of in-context learning, a powerful capability of Large Language Models (LLMs). While in-context learning has revolutionized various applications by utilizing a few task-relevant demonstrations, it also introduces concerns about the trustworthiness of LLM responses, particularly regarding hallucinations. This research delves into the dual sources of uncertainty in LLMs—aleatoric uncertainty from provided demonstrations and epistemic uncertainty from the model's configurations. A novel formulation and estimation method are proposed to quantify both types of uncertainty, offering an unsupervised, plug-and-play approach to understanding and mitigating the risks associated with LLM predictions in in-context learning scenarios. Together, these studies contribute to advancing natural language understanding by incorporating structured knowledge and addressing the nuances of uncertainty in AI models.

1.1.3 Representation Learning for Text-attributed Networks

Text-attributed graphs (TAGs), where nodes and edges are enriched with textual descriptions, are increasingly prevalent in real-world applications. The key challenge in representation learning on TAGs lies in effectively integrating textual semantics with the graph's topological structure. The paper presents a novel approach to address this challenge by focusing on textual-edge graphs (TEGs), which are particularly useful for capturing rich contextual information through text annotations on edges.

The proposed framework, LINK2DOC, is designed for link prediction tasks on TEGs, where existing methods often struggle to fully capture the complex interplay between edge semantics and graph topology. LINK2DOC innovatively summarizes the neighborhood information between node pairs as a human-readable document, thereby preserving both semantic content and structural connections. By leveraging a self-supervised learning model, LINK2DOC enhances the graph neural network's (GNN's) ability to understand and process text through language models. Extensive empirical evaluations across multiple real-world datasets demonstrate that LINK2DOC outperforms existing edge-aware GNNs and pre-trained language models, offering superior performance in link prediction, edge classification, and overall understanding of TEGs. This work is devoted to learning a unified representation of TAGs, particularly in effectively merging textual and topological information.

1.2 Contribution

The major proposed research contributions that have been addressed up to now can be stated as follows:

Enhancing Graph Data Mining by Exploiting Semantic Information on Networks:

- **Heterogeneous Graph Generation.** We not only formulate a new paradigm of heterogeneous graph generation but also identify and resolve its unique chal-

lenges in preserving various heterogeneous graph properties. The proposed framework can effectively learn the underlying distribution of heterogeneous graphs. It generates heterogeneous graphs with ensuring the preservation of various heterogeneous graph properties.

- **Information Diffusion Source Localization on Cross-networks.** We propose a unified framework for cross-network source localization that can jointly capture 1) both static and dynamic node features, and 2) the heterogeneous diffusion patterns of both networks. The approximation of diffusion sources is fully aware of various node features and the interplay of cross-network information diffusion patterns.

- **Deep Graph Representation Learning and Optimization for Influence Maximization (IM).** We formulate the learning-based IM problem as embedding the initial discrete optimization domain into continuous space for easing the optimization and identify its unique challenges arising from real applications. The proposed framework models the representation of the seed set in a latent space, and the representation is jointly trained with the model that learns the underlying graph diffusion process in an end-to-end manner.

Integrating Structured Knowledge and Quantifying Uncertainty in Natural Language Understanding

- **Open-ended Commonsense Reasoning.** The open-ended commonsense reasoning is formulated as a multi-hop reasoning task iteratively conducted on an external knowledge graph. We leverage the implicit knowledge stored in PLMs to guide the reasoning process on Knowledge Graphs, and the retrieved reasoning paths serve as additional explanations to justify the answer choice.

- **Uncertainty Quantification for In-Context Learning of Large Language Models.** We formulate the problem of uncertainty quantification of the

Large Language Model (LLM)'s response that extracts epistemic and aleatoric uncertainties from the predictive distribution of LLMs with in-context learning from the mutual information perspective.

Representation Learning of Text-attributed Graph

- **Problem**. We formulate the problem of link prediction on textual-edge graphs and highlight the unique challenges of learning representations on textual-edge graphs for link prediction.
- **Method**. We propose an integrated framework to jointly consider topology and semantic information in textual-edge graphs, which consists of 1) coherent document composition to summarize local topology information between node pairs in plain language; and 2) a self-supervised learning module to teach graph neural networks language processing ability.
- **Experiment**. We empirically compare our method against existing state-of-the-art models in four real-world datasets. Results have shown our proposed methods can elevate the performance of general GNNs and achieve competitive performance against edge-aware GNNs.

1.3 The Organization of Thesis

The remainder of the dissertation is as follows. Chapter 2 discusses a series of works for enhancing graph data mining by exploiting semantic information on networks, including the deep graph generation of heterogeneous networks, the source localization in cross-network information diffusion, and the influence maximization on social networks. Chapter 3 explores two works on integrating structured knowledge and quantifying uncertainty in natural language understanding, which consists of a novel open-ended commonsense reasoning problem that leverages the whole knowledge graph as

the answer scope, and an uncertainty quantification method that decomposes the predictive uncertainty of LLM's response into its aleatoric and epistemic components. After that, Chapter 4 discusses the work of TEG representation learning, which tries to discover the synergy between the LLMs and GNNs in learning graph data with rich semantic information on edges for link prediction. Finally, Chapter 5 concludes the dissertation with major contributions of each work.

Chapter 2

Enhancing Graph Data Mining by Exploiting Semantic Information on Networks

Incorporating semantic information into graph data mining can be crucial for enhancing the understanding of relationships and interactions within graph structures, leading to more accurate and robust models. For tasks such as heterogeneous graph generation, semantic meaning enables models to capture nuanced relationships between different types of nodes and edges, preserving both local and global graph properties that might otherwise be overlooked. In information diffusion source localization across networks, semantic awareness aids in disentangling static and dynamic node features, allowing for more precise modeling of the latent distribution of source nodes and the diffusion process across interconnected networks. Similarly, in influence maximization on social networks, integrating semantic information can improve the model's ability to learn diverse diffusion patterns, enabling more effective identification of influential nodes under varying constraints.

In this chapter, I will introduce three works that leverage semantic information

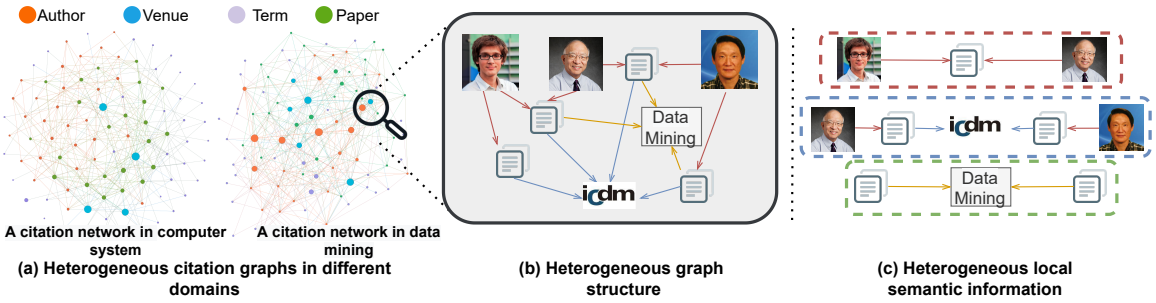


Figure 2.1: Examples of the heterogeneous graph in an academic field.

to enhance different graph data mining tasks. By incorporating semantic meaning, these works demonstrate that graph mining tasks can achieve greater accuracy, generalization, and adaptability, ultimately improving the quality of insights derived from complex, real-world networks.

2.1 Deep Generation of Heterogeneous Networks

Graphs, a ubiquitous data structure, model connections (edges) between objects (nodes). Research in graph analysis primarily falls into two categories: 1) *graph representation learning*, which encodes topological and semantic graph information into vector spaces [178], and 2) *graph generation*, which reconstructs graph data from low-dimensional representations of graph rules or distributions [58]. While most prior work has focused on homogeneous graphs (single node and edge types), heterogeneous graphs generalize this by incorporating multiple node and edge types, as seen in citation [213] and social networks [43]. For instance, a citation network (Figure 2.1 b) may include nodes like authors, papers, venues, and terms, with edges such as authorship or publication. A key feature of heterogeneous graphs is their local semantics, captured by *meta-paths*, which are sequences of node and edge types reflecting diverse relationships [154, 153]. For example, two authors might be connected via a meta-path by co-authoring a paper or by publishing in the same venue (Figure 2.1 b).

Recent advancements in graph neural networks have led to significant progress in heterogeneous graph representation learning and embedding [44, 169, 65, 189, 188, 190, 192], enabling downstream tasks such as meta-relation detection [44, 49], heterogeneous node embedding [169, 65], and link prediction [207, 189]. However, *heterogeneous graph generation* remains underexplored. Generating realistic heterogeneous graphs offers two key benefits: 1) It enhances understanding of latent graph distributions by capturing implicit properties, and 2) it supports applications like recommendation systems [147], knowledge graph reasoning [207], and node proximity search [154]. Despite its importance, only one prior work [60] has attempted this, using hand-crafted rules that fail to learn real data distributions. In contrast, deep generative models have excelled in homogeneous graph generation [58, 59, 16, 200, 150, 193], learning latent graph-structure distributions directly from data without manual rules, and effectively preserving structural properties.

However, existing deep generative models designed for homogeneous graphs cannot be trivially adapted to heterogeneous graphs due to the following challenges:

1) *Difficulties in preserving heterogeneous semantic information.* Models designed for homogeneous graphs often rely on random walks to learn topological distributions ([16, 19]) or directly model edge distributions ([86, 150]). However, heterogeneous graphs involve diverse meta-paths (Figure 2.1 c), which encode complex semantic patterns. Without specific consideration of meta-paths, adapting these models to heterogeneous graphs fails to capture such intricate semantics.

2) *Difficulties in preserving heterogeneous higher-order structural information.* Beyond meta-paths, higher-order structures like triangles or orbits emerge frequently in heterogeneous graphs (Figure 2.1 b). These structures, representing more intricate node relationships (e.g., an author writing multiple papers on the same topic), are difficult to model and preserve during graph generation.

3) *Difficulties in preserving heterogeneous global information.* Meta-paths also en-

code global patterns crucial for maintaining graph-wide properties, such as the ratio of node and edge types across domains (Figure 2.1 a). Preserving these global distributions, alongside topological and type-specific patterns, poses a significant challenge.

To address these challenges, we propose an end-to-end framework, Heterogeneous Graph Generation (HGEN). HGEN aims to generate novel heterogeneous graphs by preserving local semantic, higher-order structural, and global properties through meta-path distribution modeling. Key components include:

- **Problem.** We not only formulate a new paradigm of heterogeneous graph generation but also identify and resolve its unique challenges in preserving various heterogeneous graph properties.
- **Framework.** We propose an end-to-end framework for heterogeneous graph generation. The proposed framework can effectively learn the underlying distribution of heterogeneous graphs. It generates heterogeneous graphs while ensuring the preservation of various heterogeneous graph properties.
- **Evaluation.** We conduct extensive experiments on both synthetic and real-world heterogeneous graphs. Compared with state-of-the-art baselines, HGEN achieves competitive results in preserving most of the static graph properties. In addition, HGEN is shown to be capable of generating realistic heterogeneous graphs by preserving important meta-path information.

2.1.1 Related Work

Heterogeneous Graph Representation Learning. In recent years, graph neural network (GNN) has achieved massive success in extensive applications [85, 202] due to its capability of effectively learning relationships and interactions on non-Euclidean data. There exist plenty of attempts to adopt GNNs to learn with heterogeneous graphs, and almost all of them rely on employing meta-paths to model heterogeneous

structures [192]. Specifically, proximity-preserving methods [44, 49, 189, 190] aim to capture heterogeneous network topological information via meta-path-constrained random walks. On the other line of approach, [188, 169, 65] try to aggregate information from heterogeneous neighbors via multiple layers of learnable projection functions. Throughout the study of heterogeneous graphs [154, 192], *meta-path* serves as the fundamental building block owing to its nonpareil ability to carry both graph topological and rich semantic information.

Graph Generation. Graph generative models have evolved significantly due to their applications in link prediction [16, 150], protein structure analysis [34], and information diffusion in social networks [167]. Traditional methods (e.g., random graphs, and stochastic block models) struggle to capture complex dependencies and fail to preserve the statistical properties of real-world graphs. Recent advances focus on deep graph generation, categorized into sequential-based and one-shot-based approaches [58]. Sequential methods [200, 16, 152] generate nodes and edges autoregressively using LSTMs, but their reliance on fixed orderings limits flexibility and scalability. One-shot methods [150, 34, 16, 191, 193, 205, 109] generate graph topology and attributes simultaneously, but their high time complexity restricts their use on large graphs. Additionally, methods for multi-attributed graph generation [200, 59, 55] focus on homogeneous graphs, failing to capture the higher-order local semantics crucial in heterogeneous graphs, which arise from the combinations of different node and edge types.

2.1.2 Problem Formulation

A heterogeneous graph [148, 192] is a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with multiple types of objects and relations. $\mathcal{V}$ is the set of objects (i.e., nodes), where each node $v_i \in \mathcal{V}$ is associated with a node type $o = \phi(v_i)$. $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, where each edge $e_{ij} \in \mathcal{E}$ is

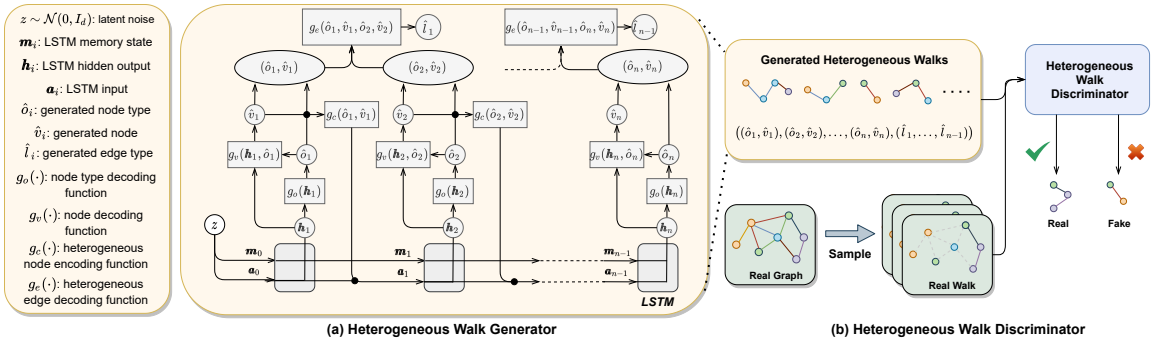


Figure 2.2: The illustration of the heterogeneous walks generation in HGEN.

associated with a relation type $l = \psi(e_{ij})$.

In the study of heterogeneous graphs, the concepts of meta-paths are widely considered as cornerstones and adopted to systematically capture numerous semantic relationships across multiple types of objects, which are defined as a path over the graph [155, 192]. Hence meta-paths are indispensable to be considered as basic units for heterogeneous graph generation. Concretely, a meta-path $\mathbf{o}$ is defined as a sequence of object types and edge types $\mathbf{o} = \big((o_1, o_2, ..., o_n), (l_1, l_2, ..., l_{n-1})\big) = o_1 \xrightarrow{l_1} o_2 \xrightarrow{l_2} ... \xrightarrow{l_{n-1}} o_n$, where each o_i and l_j are node type and edge type in the sequence, respectively. Each meta-path captures the rich semantic information between its two ends o_1 and o_n. In heterogeneous graphs, the local semantic information is carried on each of walks $\mathbf{v} = (v_0, v_1, ..., v_n)$ and its associated meta-path $\mathbf{o}$. We again take Figure 2.1 (c) as an example, there exist two meta-paths between papers: (*Paper, Author, Paper*) and (*Paper, Venue, Paper*). The utilization of different meta-paths allows the heterogeneous graph to contain rich topological and semantics among diverse objects.

With the preliminary notion of the heterogeneous graph, we formalize the heterogeneous graph generation problem as follows:

Problem 1 (**Heterogeneous Graph Generation**)**.** The goal of the heterogeneous graph generation is to learn a distribution $p_{\text{data}}(\mathcal{G})$ from the observed heterogeneous graphs such that a new graph $\hat{\mathcal{G}}$ can be obtained by sampling $\hat{\mathcal{G}} \sim p_{\text{data}}(\mathcal{G})$.

2.1.3 Heterogeneous Graph Generation

In this work, we propose a new heterogeneous graph generation framework, named HGEN, which leverages a *heterogeneous walk generator* to jointly learn the distribution of local walks and the associated meta-paths for capturing both heterogeneous topological and local semantic information. A novel *heterogeneous graph assembler* is also proposed to construct new heterogeneous graphs by capturing the global heterogeneous property, namely different meta-path ratios.

Heterogeneous Walk Generator

In the observed graph $\mathcal{G}$, a heterogeneous walk is defined as a tuple that consists of two components: a walk $\mathbf{v}$ and an associated meta-path $\mathbf{o}$. The proposed heterogeneous walk generator G is defined as a probabilistic sequential learning model to generate synthetic heterogeneous walks: $(\hat{\mathbf{v}}, \hat{\mathbf{o}}) = \big((\hat{v}_1, \hat{v}_2, ..., \hat{v}_n), ((\hat{o}_1, \hat{o}_2, ..., \hat{o}_n), (\hat{l}_1, \hat{l}_2, ..., \hat{l}_{n-1}))\big)$, where the $\hat{\mathbf{v}}$ and $\hat{\mathbf{o}}$ are denoted as the generated walk and associated meta-path, respectively. We use $\hat{v}_i$, $\hat{o}_i$, and $\hat{l}_i$ to denote each of the generated node, node type, and edge type in $(\hat{\mathbf{v}}, \hat{\mathbf{o}})$, respectively. Figure 2.2 (a) illustratively summarizes the whole generative process of each synthetic heterogeneous walk.

Heterogeneous Walk Generation. We model G as a sequential learning process based on a recurrent architecture, and each unit f_θ in the sequential model is parameterized by θ so that it can generate a node type $\hat{o}$ and a corresponding node $\hat{v}$ that belongs to this node type in a hierarchical manner. Precisely, the node type $\hat{o}$ is determined based on the previously generated sequence, and the node $\hat{v}$ is then coherently determined by the generated node type as well as the generated sequence. Both generated node type $\hat{o}$ and node $\hat{v}$ together provide information for the generation of the next node type and node instance.

Specifically, at each recurrent block (i.e., time step) t, f_θ produces two outputs

$(\boldsymbol{m}_t, \boldsymbol{h}_t)$, where the $\boldsymbol{m}_t$ is the current memory state and the $\boldsymbol{h}_t$ is a latent probabilistic distribution (i.e., hidden output of f_θ) denoting the information carried from previous time steps. We first sample the node type $\hat{o}_t \sim g_o(\boldsymbol{h}_t)$ based on the probability distribution $\boldsymbol{h}_t$, where the $g_o(\cdot)$ is a node type decoding function. We then sample the node $\hat{v}_t$ by a node decoding function $\hat{v}_t \sim g_v(\boldsymbol{h}_t, \hat{o}_t)$ that takes $\boldsymbol{h}_t$ and $\hat{o}_t$ as inputs. Lastly, the generated node type $\hat{o}_t$ and node $\boldsymbol{h}_t$ are fused by a heterogeneous node encoding function $g_c(\hat{o}_t, \hat{v}_t)$, which then serves as the input of next recurrent block.

Heterogeneous Node Sampling. To overcome the second challenge, we cannot uniformly sample $\hat{v}_t$ based on the node type $\hat{o}_t$ because such a way may cause the neglection of (1) *node structural* distribution and (2) *node semantic* distribution. To tackle this challenge, since latent node embedding could encode both topological and semantic information into the node, we propose to calculate a latent embedding $\tilde{v}_t$ of the next node v_t, then we select with a higher probability the closer embedding among all the embeddings that belong to node type $\hat{o}_t$ so that the next node v_t can be determined by the sampled embedding. More specifically, we first calculate the latent node embedding $\tilde{v}_t$ based on the sampled node type $\hat{o}_t$ by a simple linear transformation. We then calculated the distance between $\tilde{v}_t$ and other node embedding $\tilde{v}_i^{(\hat{o}_t)}$, meaning any node $\tilde{v}_i$ belonging to the sampled node type $\hat{o}_t$. Given a total number of k embeddings that belong to the type $\hat{o}_t$, the next node $\hat{v}_t$ can be sampled from a multinomial distribution:

$$\hat{v}_t \sim \text{Multi}(\tilde{v}_1^{(\hat{o}_t)}, \tilde{v}_2^{(\hat{o}_t)}, ..., \tilde{v}_k^{(\hat{o}_t)}; p_1, p_2, ..., p_k),$$

where each $p_i = -\|d(\tilde{v}_t, \tilde{v}_i^{(\hat{o}_t)})\|^2$ and $d(\cdot, \cdot)$ is a distance metric such as Euclidean distance. Note that the node embedding $\tilde{v}_i^{(\hat{o}_t)}$ can be obtained from a conventional heterogeneous node embedding technique such as [49].

In order to generate a variable-length heterogeneous walk, we incorporate a end-

of-sequence token as an additional node type so that the heterogeneous walk generator stops when the sampled node type is the token at any steps. Therefore, the proposed generator is able to produce variable-length heterogeneous walks. Finally, the edge type l_t can be predicted by a simple edge decoding function $g_e(\hat{o}_t, \hat{v}_t, \hat{o}_{t-1}, \hat{v}_{t-1})$ that takes its two end nodes $\hat{v}_{t-1}$ and $\hat{v}_t$ as well as their node types $\hat{o}_{t-1}$ and $\hat{o}_t$ as inputs. In all, we summarize the overall generative process as follows:

$$
\begin{aligned}
&\boldsymbol{a}_0 = 0,\ \boldsymbol{m}_0 = f_0(\boldsymbol{z}),\ \boldsymbol{z} \sim \mathcal{N}(0,1) \\
&\boldsymbol{a}_1 = g_c(\hat{o}_1, \hat{v}_1),\ \hat{v}_1 \sim g_v(\boldsymbol{h}_1, \hat{o}_1),\ \hat{o}_1 \sim g_o(\boldsymbol{h}_1), (\boldsymbol{m}_1, \boldsymbol{h}_1) = f_\theta(\boldsymbol{m}_0, \boldsymbol{a}_0) \\
&\boldsymbol{a}_2 = g_c(\hat{o}_2, \hat{v}_2),\ \hat{v}_2 \sim g_v(\boldsymbol{h}_2, \hat{o}_2),\ \hat{o}_2 \sim g_o(\boldsymbol{h}_2), (\boldsymbol{m}_2, \boldsymbol{h}_2) = f_\theta(\boldsymbol{m}_1, \boldsymbol{a}_1) \\
&\hat{l}_1 = g_e(\hat{o}_2, \hat{v}_2, \hat{o}_1, \hat{v}_1) \\
&\dots \\
&\hat{v}_n \sim g_v(\boldsymbol{h}_n, \hat{o}_n),\ \hat{o}_n \sim g_o(\boldsymbol{h}_n),\ (\boldsymbol{m}_n, \boldsymbol{h}_n) = f_\theta(\boldsymbol{m}_{n-1}, \boldsymbol{a}_{n-1}) \\
&\hat{l}_{n-1} = g_e(\hat{o}_n, \hat{v}_n, \hat{o}_{n-1}, \hat{v}_{n-1})
\end{aligned}
$$

In this work, we utilize LSTM as the recurrent architecture, and f_θ becomes a single LSTM unit. To initialize the whole generative process, G takes a random noise $\boldsymbol{z}$ as input, which is drawn from a standard Gaussian distribution. Additionally, for the node type decoding function $g_o(\cdot)$, we apply the Gumbel-softmax trick [67] in $g_o(\cdot)$ to make the whole sampling differentiable. Finally, in most of the real-world scenarios, the edge type l_t can be determined by the types of its two end nodes $\hat{o}_t$ and $\hat{o}_{t-1}$ if there does not exist multi-typed relations between two node types. In this case, the heterogeneous walk generator can be simplified only to generate node sequences and associated node types.

Heterogeneous Generator Training and Utilization

We utilize a heterogeneous discriminator D to distinguish between real and generated heterogeneous walks, where the real heterogeneous walks are uniformly sampled from the observed graph. We then propose a heterogeneous graph assembler to construct new graphs based on the sampled heterogeneous walks.

We first introduce the overall objective function of the Wasserstein heterogeneous GAN [8], which is written as:

$$\begin{aligned}\mathcal{L}_{\text{HGEN}} = \max \mathbb{E}_{(\mathbf{o},\mathbf{v})\sim p(\mathcal{G})}[D_o(\mathbf{o}) + D_v(\mathbf{v})] \\ - \mathbb{E}_{z\sim p(z)}[D_o(\hat{\mathbf{o}}) + D_v(\hat{\mathbf{v}})], s.t.\ G(z) = (\hat{\mathbf{o}}, \hat{\mathbf{v}}),\end{aligned} \tag{2.1}$$

where $\mathbf{v}$ and $\mathbf{o}$ are the random walk and associated meta-path, respectively, directly sampled from the observed $\mathcal{G}$. They are the real data for training our heterogeneous walk generator G. Specifically, given an observed $\mathcal{G}$, we utilize random-walk-based method to uniformly sample a set of random walks $\{\mathbf{v}_1, \mathbf{v}_2, ...\}$, where each $\mathbf{v}_i$ is a node sequence s.t. $\mathbf{v}_i = (v_1, v_2, ..., v_n)$. In addition, we extract the meta-path information $\mathbf{o}_i = \big((o_1, o_2, ..., o_n), (l_1, l_2, l_{n-1})\big)$ from each $\mathbf{v}_i$.

The heterogeneous discriminator D in Eq. (2.1) is designed as a parallel recurrent architecture in order to individually distinguish whether each component in the heterogeneous walks is valid or not. Specifically, at each recurrent block (i.e., each step) t, the discriminator D takes two inputs: the generated node type $\hat{o}_t$ and node index $\hat{v}_t$, each of which is fed into an individual recurrent unit. After processing both sequences, the discriminator returns a single score $D_v(\mathbf{v}) + D_o(\mathbf{o})$ that represents the probability of the heterogeneous walk being real.

Heterogeneous Graph Assembler. To assemble a heterogeneous graph from the generated heterogeneous walks, we further propose a novel stratified heterogeneous edge sampling strategy to achieve the following steps: 1) it first samples a node $\hat{v}_i$

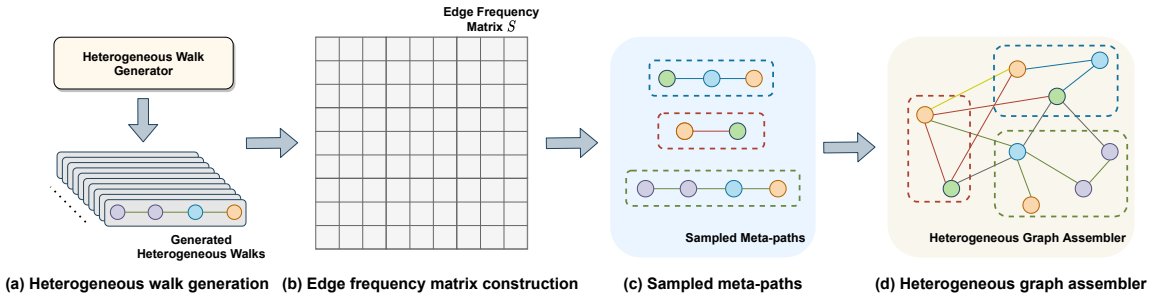


Figure 2.3: The process of heterogeneous graph assembler.

and its type $\hat{o}_i$ from all of the generated heterogeneous walks; 2) based on the node type $\hat{o}_i$, we then sample a meta-path that starts with $\hat{o}_i$; 3) we iteratively sample the next node $\hat{v}_{i+1}$ in the sampled meta-path if both of the node type $\hat{o}_{i+1}$ and edge type $\hat{l}_i$ fits the meta-path pattern.

More specifically, the generator G firstly produces a sufficient number of heterogeneous walks as shown in Figure 2.3(a). We then construct a symmetric adjacency matrix S with size $|\mathcal{V}| \times |\mathcal{V}|$ to record the count of edges observed from the sampled heterogeneous walks in each entry S_{ij}, where the $|\mathcal{V}|$ is the size of the node set. Next, we collect all of the meta-path patterns generated by the generated heterogeneous walks, as shown in Figure 2.3 (b-c). For the first step of the stratified heterogeneous edge sampling, we sample a node $\hat{v}_i$ and its type $\hat{o}_i$ based on the node degree distribution $\frac{\sum_j S_{ij}}{|\mathcal{V}|}$. For the second step, among all the meta-paths $\{\mathbf{o}_1^{(f)}, \mathbf{o}_2^{(f)}, ...\}$ that start with the node type $\hat{o}_i$, we sample a meta-path $\mathbf{o}_i^{(f)}$ based on the probability $\frac{c(\mathbf{o}_i^{(f)})}{T^{\hat{o}_i}}$, where $T^{\hat{o}_i}$ is the total count of generated meta-paths that starts with node type $\hat{o}_i$ and $c(\mathbf{o}_i^{(f)})$ is the count of meta-path pattern $\mathbf{o}_i^{(f)}$. For the third step, by following this meta-path pattern $\mathbf{o}_r = (o_1, o_2, ..., o_n)$, we iteratively sample all the nodes whose node types are regulated by the meta-path. Precisely, we sample the next node v_j by sampling all the neighbors of the current node v_i with the probability $p_{v_i v_j} = (S_{ij})/(\sum_s S_{is})$ such that all the nodes v_s belong to the specific node type o_j following the meta-path $\mathbf{o}_i^{(f)}$. The sampled node sequence $\mathbf{v}_r = (v_0, v_1, ...)$ is then added to the current under construction. We continue the stratified heteroge-

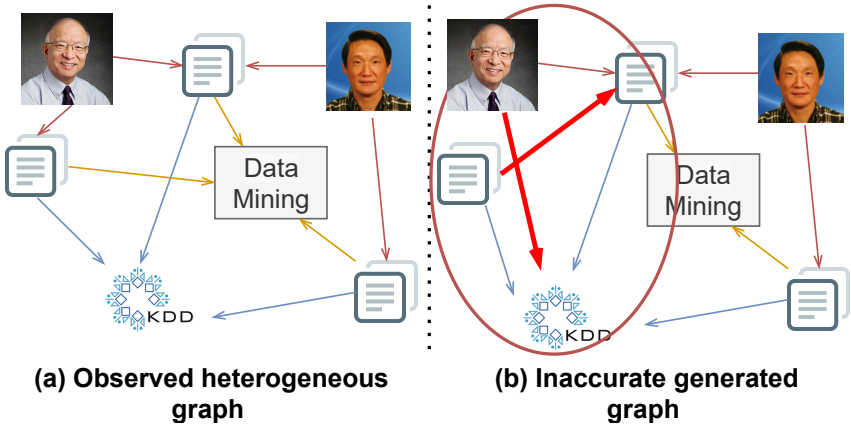


Figure 2.4: Example of two heterogeneous graphs with different semantic information: the observed meta-path patterns are different, although the node and edge distribution are the same between two graphs.

neous edge sampling strategy until the desired amount of edges is reached. The final assembled graph is visualized in Figure 2.3 (d).

2.1.4 Meta-path Information Preservation Analysis

As we discussed in Section 2.1.2, it is significant to preserve the meta-path information in our generated graph. Taking Figure 2.4 as an example, although both graphs have exactly the same structure, they are still regarded as two different heterogeneous graphs since their meta-path distributions are different. Given the importance of the meta-path information in heterogeneous graph generation, we further show that our framework can preserve this meta-path information as proved in Theorem 1.

Theorem 1. *The distribution of meta-path patterns* $\overline{\mathcal{O}}^{(r)}$ *of the generated heterogeneous graph equals the distribution of meta-path patterns* $\overline{\mathcal{O}}$ *in the observed heterogeneous graph, namely* $p(\overline{\mathcal{O}}^{(r)}) = p(\overline{\mathcal{O}})$.

Proof. We will prove that the ratio of the meta-path patterns can be preserved in three steps: 1) the ratio of different meta-path patterns can be preserved during the sampling procedure; 2) the ratio of generated meta-path patterns can be preserved

during the generation procedure; 3) the meta-path patterns can be preserved during the graph assembling procedure.

Meta-path Ratio Preservation in Sampling. Let $\overline{\mathcal{O}} = (\overline{\mathbf{o}}_1, \overline{\mathbf{o}}_2, ...)$ be the collection of meta-paths obtained from the observed heterogeneous graph $\mathcal{G}$, each $\overline{\mathbf{o}}_i$ is a meta-path in one-hot format $\overline{\mathbf{o}}_i \in \{0,1\}^{1\times R}$, where the R is the total number of different meta-path patterns. $\overline{\mathcal{O}}^{(\tau)} = (\overline{\mathbf{o}}_1^{(\tau)}, \overline{\mathbf{o}}_2^{(\tau)}, ..., \overline{\mathbf{o}}_K^{(\tau)})$ is the sequence of sampled meta-paths with sampling size K, where each meta-path $\overline{\mathbf{o}}_j^{(\tau)} \in \{0,1\}^{1\times R}$ is drawn independent and identically distributed ($i.i.d$) from $\overline{\mathcal{O}}$.

Suppose that $\boldsymbol{\mu} = [\mu_1, \mu_2, ..., \mu_R]^T$ denotes the probability of each individual meta-path pattern in $\overline{\mathcal{O}}$, it is obvious that $\mathbb{E}[\overline{\mathbf{o}}_i|\boldsymbol{\mu}] = \sum_{\overline{\mathbf{o}}_i} p(\overline{\mathbf{o}}_i|\boldsymbol{\mu})\overline{\mathbf{o}}_i = [\mu_1, \mu_2, ..., \mu_R]^T = \boldsymbol{\mu}$. Now consider the total K observations $\overline{\mathcal{O}}^{(\tau)} = (\overline{\mathbf{o}}_1^{(\tau)}, \overline{\mathbf{o}}_2^{(\tau)}, ..., \overline{\mathbf{o}}_K^{(\tau)})$, the corresponding likelihood function takes the form:

$$p(\overline{\mathcal{O}}^{(\tau)}|\boldsymbol{\mu}) = \prod_i^R \prod_j^K \mu_j^{\overline{\mathbf{o}}_{ij}^{(\tau)}} = \prod_j^K \mu_j^{\sum_n \overline{\mathbf{o}}_{nj}^{(\tau)}} = \prod_j^K \mu_j^{m_j} \tag{2.2}$$

We see that the likelihood function depends on the K data points only through the R quantities: $m_j = \sum_n \overline{\mathbf{o}}_{nj}^{(\tau)}$. Since the number of observations of $\overline{\mathbf{o}}_j^{(\tau)}$ equals 1, we achieved sufficient statistics for this distribution so that $p(\overline{\mathcal{O}}^{(\tau)}) = p(\overline{\mathcal{O}})$.

Meta-path Ratio Preservation in Generation. Since we have proved the meta-path ratio can be preserved during the sampling, the next step is to show that the distribution of generated meta-paths $p(\overline{\mathcal{O}}^{(g)})$ is equal to $p(\overline{\mathcal{O}}^{(\tau)})$. Proving $p(\overline{\mathcal{O}}^{(g)}) = p(\overline{\mathcal{O}}^{(\tau)})$ is equivalent to prove whether $p_{data} = p_g$ in the GAN setting. As proved in the works of GANs and their variants [54, 8], it showed that the objective function of the generator G is equivalent to optimize the distribution distance between p_{data} and p_g if the discriminator D is optimal. Therefore, global optimality of $p_g = p_{data}$ can be achieved if both generator G and discriminator D have enough capability. Therefore, $p(\overline{\mathcal{O}}^{(g)}) = p(\overline{\mathcal{O}}^{(\tau)})$ if both G and D are optimal in our framework.

Meta-path Ratio Preservation in Assembling. Finally, we show that our graph assembling method can also preserve the meta-path ratio from the generated data $\overline{\mathcal{O}}^{(g)}$ such that $p(\overline{\mathcal{O}}^{(g)}) = p(\overline{\mathcal{O}}^{(r)})$. As discussed in Section 2.1.3, the new graph $\hat{\mathcal{G}}$ is directly assembled by meta-paths $(\overline{\mathbf{o}}_1^{(g)}, \overline{\mathbf{o}}_2^{(g)}, ..., \overline{\mathbf{o}}_Q^{(g)})$ that are sampled $i.i.d$ from $\overline{\mathcal{O}}^{(g)}$ with sampling size Q, which is exactly the reverse procedure of Eq. (2.2).

Therefore, if both generator G and discriminator D are optimal, the multinomial distribution $p(\overline{\mathcal{O}})$ of distinct meta-path patterns can be preserved in all three steps of our generation framework. □

2.1.5 Experiment

In this section, we compare HGEN to the adaption of existing baselines, demonstrating its effectiveness in generating realistic heterogeneous graphs in diverse settings.

2.1.6 Data

Synthetic Datasets. We synthesize random heterogeneous graphs of different sizes through the combination of N overlapping homogeneous graphs, where the overlap is accomplished by node sharing. We generate three random heterogeneous graphs (named as Syn_{100}, Syn_{200}, and Syn_{500}) with node size 100, 200, and 500, respectively. The number of node types in each synthetic heterogeneous graph is 3.

Real-world Datasets. We also employ three large-scale real-world heterogeneous graph datasets in our experiment.

- **PubMed**. This dataset consists of four classes of nodes: Gene (G), Disease (D), Chemical (C), and Species (S). We construct a sub-graph that relates to all Chemical nodes labeled in [192]. There are $1,565$ nodes and $13,532$ edges.
- **IMDB**. This movie-related heterogeneous graph is adopted from [169], which contains three node types: Director (D), Actor (A), Movie (M), and Genre (G).

Graphs	Models	LCC	TC	Clustering Coef.	Powerlaw Coef.	Assortativity	Degree Distribution Dist.	EO Rate	Uniqueness
Syn-100	GraphRNN	78.43 ± 2.23	16.62 ± 5.42	0.002 ± 0.01	1.611 ± 0.09	$\mathbf{-0.153 \pm 0.07}$	$2.19e{-2} \pm 3.21e{-3}$	$37.21\% \pm 1.08\%$	$33.09\% \pm 7.06\%$
	NetGAN	80.12 ± 3.45	6.79 ± 1.76	0.001 ± 0.00	1.524 ± 0.21	-0.213 ± 0.09	$1.33e{-2} \pm 6.46e{-3}$	$8.74\% \pm 0.82\%$	$\mathbf{94.03\% \pm 0.49\%}$
	GraphVAE	99.01 ± 0.00	224.81 ± 5.13	0.70 ± 0.04	4.579 ± 0.05	-0.73 ± 0.05	$3.71e{-1} \pm 1.98e{-2}$	$11.5\% \pm 1.09\%$	$65.54\% \pm 2.98\%$
	VGAE	48.9 ± 4.63	63.7 ± 46.25	0.184 ± 0.06	$\mathbf{1.87 \pm 0.10}$	0.1 ± 0.03	$2.23e{-1} \pm 6.08e{-2}$	$\mathbf{3.23\% \pm 0.09\%}$	$51.1\% \pm 3.04\%$
	HGEN	$\mathbf{81.13 \pm 2.42}$	$\mathbf{53.12 \pm 3.78}$	$\mathbf{0.079 \pm 0.01}$	1.782 ± 0.01	-0.114 ± 0.03	$\mathbf{8.79e{-3} \pm 3.12e{-3}}$	$10.2\% \pm 0.17\%$	$92.97\% \pm 0.72\%$
	Real	85	36	0.072	1.832	-0.169	N/A	N/A	N/A
Syn-200	GraphRNN	132.76 ± 1.08	2.54 ± 0.77	0.001 ± 0.00	1.603 ± 0.01	-0.05 ± 0.01	$5.15e{-2} \pm 3.07e{-3}$	$25.81\% \pm 2.65\%$	$27.72\% \pm 3.07\%$
	NetGAN	153 ± 1.56	2.24 ± 0.35	0.001 ± 0.00	1.579 ± 0.31	-0.008 ± 0.001	$6.43e{-2} \pm 4.2e{-3}$	$11.32\% \pm 0.77\%$	$95.88\% \pm 3.19\%$
	GraphVAE	195.43 ± 1.12	51.32 ± 1.01	0.002 ± 0.001	5.377 ± 0.21	-0.75 ± 0.05	$5.38e{-1} \pm 1.7e{-2}$	$\mathbf{1.78\% \pm 0.41\%}$	$64.37\% \pm 2.94\%$
	VGAE	86.2 ± 16.93	860.4 ± 185.9	0.23 ± 0.04	$\mathbf{1.787 \pm 0.08}$	0.2 ± 0.15	$8.53e{-2} \pm 2.14e{-2}$	$3.74\% \pm 0.08\%$	$59.65\% \pm 1.46\%$
	HGEN	$\mathbf{158.5 \pm 2.64}$	$\mathbf{38.5 \pm 5.26}$	$\mathbf{0.043 \pm 0.01}$	1.732 ± 0.02	$\mathbf{-0.065 \pm 0.04}$	$\mathbf{2.25e{-2} \pm 5.5e{-3}}$	$4.22\% \pm 0.67\%$	$\mathbf{96.31\% \pm 5.11\%}$
	Real	180	28	0.037	1.809	-0.089	N/A	N/A	N/A
Syn-500	GraphRNN	311.59 ± 2.14	11.53 ± 5.57	0.004 ± 0.001	1.862 ± 0.01	1.862 ± 0.002	$4.05e{-2} \pm 1.1e{-3}$	$21.87\% \pm 0.86\%$	$29.54\% \pm 4.32\%$
	NetGAN	305.81 ± 14.28	$\mathbf{3 \pm 1.21}$	$\mathbf{0.001 \pm 0.001}$	1.812 ± 0.07	0.03 ± 0.12	$4.83e{-2} \pm 7.4e{-4}$	$6.72\% \pm 0.13\%$	$93.98\% \pm 0.21\%$
	VGAE	97.0 ± 29.24	4346.2 ± 453.62	0.193 ± 0.02	1.77 ± 0.06	-0.022 ± 0.09	$2.22e{-1} \pm 2.4e{-2}$	$5.46\% \pm 1.12\%$	$63.65\% \pm 3.1\%$
	HGEN	$\mathbf{347.88 \pm 7.63}$	74.88 ± 4.78	0.031 ± 0.01	$\mathbf{1.865 \pm 0.02}$	$\mathbf{-0.097 \pm 0.01}$	$\mathbf{2.81e{-2} \pm 3.4e{-3}}$	$\mathbf{1.49\% \pm 0.11\%}$	$\mathbf{95.89\% \pm 1.18\%}$
	Real	417	8	6.5e−3	1.978	-0.12	N/A	N/A	N/A
PubMed	GraphRNN	1563.23 ± 32.46	1549.79 ± 33.62	0.01 ± 0.007	$\mathbf{1.753 \pm 0.04}$	-0.03 ± 0.01	$1.61e{-1} \pm 3.71e{-2}$	$13.41\% \pm 1.24\%$	$54.62\% \pm 4.32\%$
	NetGAN	793.2 ± 41.5	18.3 ± 0.9	0.001 ± 0.00	1.47 ± 0.11	-0.12 ± 0.02	$6.69e{-2} \pm 1.5e{-3}$	$4.32\% \pm 0.54\%$	$78.03\% \pm 0.19\%$
	VGAE	347.9 ± 7.03	$70,982.2 \pm 4,086.53$	0.234 ± 0.01	2.48 ± 0.01	-0.466 ± 0.01	$1.38e{-1} \pm 4.8e{-3}$	$\approx \mathbf{0\%}$	$22.87\% \pm 1.68\%$
	HGEN	$\mathbf{825.6 \pm 22.1}$	$\mathbf{1569.3 \pm 31.3}$	$\mathbf{0.034 \pm 0.003}$	1.634 ± 0.07	$\mathbf{-0.143 \pm 0.08}$	$\mathbf{3.92e{-2} \pm 7.5e{-4}}$	$0.07\% \pm 0.01\%$	$\mathbf{93.91\% \pm 0.12\%}$
	Real	948	2,114	0.068	1.75	−0.208	N/A	N/A	N/A
IMDB	GraphRNN	1425.47 ± 121.5	142.13 ± 5.87	0.179 ± 0.02	2.97 ± 0.05	0.05 ± 0.04	$1.98e{-1} \pm 2.61e{-3}$	$9.87\% \pm 0.51\%$	$21.52\% \pm 3.31\%$
	NetGAN	932.5 ± 8.49	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	2.08 ± 0.01	$\mathbf{-0.25 \pm 0.07}$	$1.36e{-1} \pm 1.89e{-3}$	$7.62\% \pm 0.07\%$	$82.69\% \pm 1.27\%$
	VGAE	635.2 ± 4.16	$7,752.4 \pm 281.32$	0.141 ± 0.01	2.02 ± 0.02	-0.49 ± 0.15	$1.9e{-1} \pm 2.33e{-3}$	$\approx \mathbf{0\%}$	$42.71\% \pm 1.47\%$
	HGEN	$\mathbf{945.2 \pm 11.54}$	26.0 ± 3.28	$3.56e{-3} \pm 3.42e{-4}$	$\mathbf{2.16 \pm 0.01}$	-0.19 ± 0.04	$\mathbf{4.36e{-2} \pm 4.25e{-4}}$	$2.69\% \pm 0.04\%$	$\mathbf{88.71\% \pm 0.39\%}$
	Real	1,074	1	4.43e−4	2.51	-0.235	N/A	N/A	N/A
DBLP	NetGAN	$10,353 \pm 72.71$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	3.308 ± 0.41	-0.059 ± 0.03	$5.03e{-1} \pm 2.1e{-2}$	$5.48\% \pm 0.32\%$	$\mathbf{72.51\% \pm 0.32\%}$
	VGAE	$3,771 \pm 236.29$	1214.69 ± 452.61	0.271 ± 0.06	1.579 ± 0.07	-0.44 ± 0.11	$8.71e{-2} \pm 1.77e{-3}$	$\approx \mathbf{0\%}$	$17.26\% \pm 0.41\%$
	HGEN	$\mathbf{5,163 \pm 21.41}$	1068 ± 12.83	0.018 ± 0.001	$\mathbf{1.793 \pm 0.21}$	$\mathbf{-0.157 \pm 0.03}$	$\mathbf{5.82e{-3} \pm 1.67e{-4}}$	$1.55\% \pm 0.09\%$	$66.59\% \pm 0.17\%$
	Real	5,513	0.0	0.0	1.855	−0.201	N/A	N/A	N/A

Table 2.1: Performance evaluation over compared baselines. The *Real* rows include the values of real graphs, while the rest are the evaluation results of different algorithms. The best performance (the closest to real value) achieved under each metric for a particular dataset is highlighted in bold font. Note that we do not include GraphVAE in datasets with (≥ 500) nodes and GraphRNN in datasets with ($\geq 10,000$) nodes because the programs return errors.

We construct a subgraph that contains all the movies with a score ≥ 7.5. This graph contains $1,653$ nodes and $4,267$ edges.

- **DBLP**. This heterogeneous graph adopted from [169] contains Paper (P), Author (A), Venue (V), and Term (T) as node types. We sample a subgraph that is related to five computer science venues: *KDD*, *WSDM*, *WWW*, *ICDM*, and *ICML*. There are $1,565$ nodes and $47,885$ edges.

Experiment Setting

In our experiment, we focus on meta-paths with lengths 1, 2, and 3 as they are the most common ones in heterogeneous graphs [154]. We sample 10 graphs from each of the trained models and report results in Table 2.1. We randomly select 60% of the

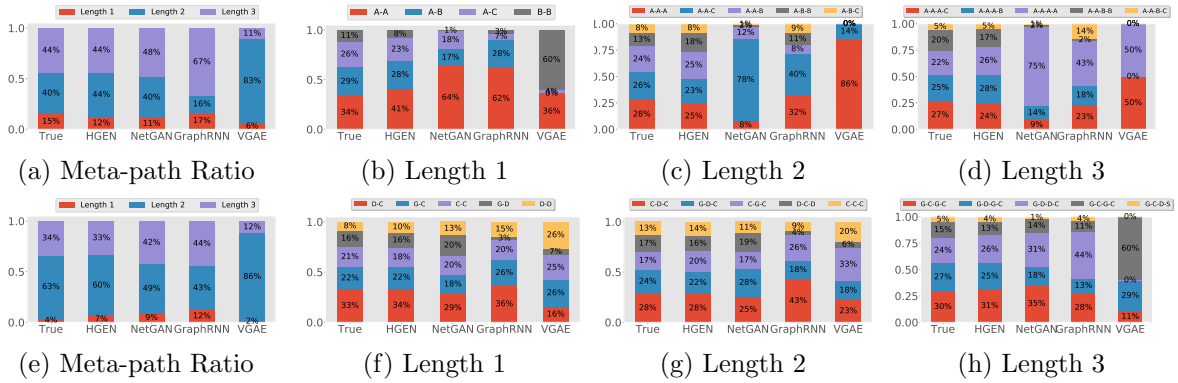
edges for training, and the remaining graph is used for testing.

Baselines. Since no direct competitors are available for heterogeneous graph generation, we carefully adapt four state-of-the-art graph generation methods: NetGAN [16], GraphVAE [150], VGAE [86], and GraphRNN [200]. We utilize node type information as a node feature of the input graph in GraphVAE and VGAE. In addition, we modify NetGAN and GraphRNN to make them available to generate node types.

Evaluation Metrics. The evaluation of heterogeneous graph generation can be divided into three aspects. 1) *Graph Statistical Properties*: we focus on six typical statistics as widely used in [16, 191, 55] for measuring the structural similarity, including LCC (the size of the largest connected component), TC (Triangle count), Clustering Coef. (clustering coefficient); Powerlaw Coef. (power-law distribution of the node degree distribution), Assortativity, and Degree Distribution Dist. (Node degree distribution Maximum Mean Discrepancy distance). 2) *Graph Novelty and Uniqueness.* Ideally, we would want the generated graphs to be diverse and similar, but not identical. To quantify this aspect, we check the uniqueness between the generated graphs by calculating their edit distances. Additionally, we calculate the EO Rate (edge overlapping rate) between the generated graphs and the testing graphs to measure the novelty of the generated graphs. 3) *Meta-path Ratio Properties*: We measure the preservation of meta-path distribution in two metrics. Firstly, we measure the meta-path length ratio preservation. Secondly, under different meta-path lengths, we also measure the distribution of the frequent meta-path patterns.

Quantitative Analysis

We evaluate HGEN's performance on standard graph statistics against various baselines, as shown in Table 2.1. HGEN consistently achieves competitive results across synthetic and real-world datasets with few exceptions. Key observations include:



(a) Meta-path Ratio (b) Length 1 (c) Length 2 (d) Length 3

(e) Meta-path Ratio (f) Length 1 (g) Length 2 (h) Length 3

Figure 2.5: The meta-path distribution comparison. 2.5a - 2.5d and 2.5e - 2.5h are the generated meta-path length distribution along with frequent meta-path patterns distribution with length 1, 2, 3 for Syn_500 dataset and PubMed dataset, respectively.

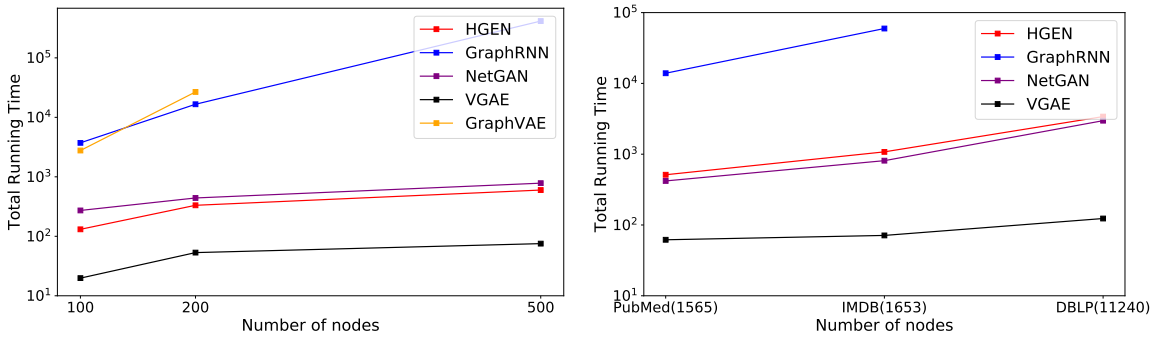
1) Node-level similarity: HGEN outperforms in most node-level metrics, showing at least a 40% improvement in node degree distribution distance. While differences in Assortativity and Power-law Coefficient are minimal, HGEN effectively captures the degree distribution across node types by jointly learning meta-path and random walk distributions. *2) Graph-level similarity*: HGEN excels in preserving community structures, particularly in datasets with rich local communities (e.g., PubMed and synthetic datasets). It leverages heterogeneous node embeddings to capture higher-order structures, resulting in superior performance on metrics such as LCC, TC, and Clustering Coefficient. However, its performance diminishes in graphs with sparse higher-order structures. *3) Random walk stability*: Both HGEN and NetGAN demonstrate stable performance across datasets compared to one-shot methods (e.g., VGAE, GraphVAE) and sequential models (GraphRNN). This stability arises from learning graph distributions via discrete random walks, which are less sensitive to varying graph characteristics. *4) Baseline limitations*: Although VGAE achieves high performance on some metrics, it struggles to generate realistic graphs due to its focus on node embeddings rather than full graph generation. Similarly, GraphRNN faces scalability issues as graph size increases, leading to less realistic outputs.

Graph Novelty and Uniqueness. The results of graph novelty and uniqueness are reported in the right two columns in Table 2.1. Specifically, HGEN achieves a generally lower EO rate across all datasets, indicating that HGEN does not purely memorize the seen heterogeneous walks in the training data. In contrast, GraphRNN has a higher EO rate, indicating GraphRNN regenerates graphs it saw during training. In addition, VGAE achieves the lowest EO rate since it fails to generate realistic heterogeneous graphs. For Uniqueness, HGEN also exceeds other one-shot and sequential-based algorithms by an evident margin, which demonstrates the diversity of the generated graphs.

Preservation of Graph Semantic Properties. To further demonstrate the performance of HGEN, we evaluate the performance of meta-path distribution preservation with other baselines. Specifically, we measure the meta-path distribution from two aspects: 1) the overall meta-path length ratio preservation in generated graphs and 2) frequent meta-path patterns under each length. The results of Syn_500 and PubMed datasets are illustrated in Figure 2.5. In general, all the methods can approximately maintain the meta-path length ratio except for VGAE. However, HGEN can constantly achieve a better performance as shown in Figure 2.5a and 2.5e. 2) As shown in Figure 2.5b - 2.5d and 2.5f - 2.5h, HGEN can outperform other methods by at least 10% in preserving the ratio of specific meta-path patterns under each length, which is expected since HGEN is able to learn and maintain the meta-path distribution from the observed graphs while others cannot.

Ablation Study

We further conduct ablation studies on the PubMed dataset to evaluate the effect of different components in HGEN, and the results are exhibited in Table 2.2. The ablative experiments are conducted based on each of the essential components in



(a) Synthetic Dataset Running Time

(b) Real Dataset Running Time

Figure 2.6: Running time comparison of different models in both synthetic and real-world datasets. It is clear that GraphVAE is not scalable in generating graphs with more than 200 nodes. GraphRNN also fails to generate large graphs (with more than $10,000$ nodes). The proposed HGEN exhibits a linear running time growth in terms of graph size growth.

	HGEN-S	HGEN-E	HGEN-A	HGEN	Real
LCC	1563.76	824.14	819.32	**825.6**	948
TC	1453.23	784.34	863.53	**1569.3**	2114
Clustering Coef.	0.026	0.015	0.016	**0.034**	0.068
Power Law Coef.	1.649	**1.652**	1.621	1.634	1.75
Assortativity	-0.09	-0.132	-0.131	**−0.143**	-0.208
Node Degree Dist.	**0.0354**	0.0388	0.0515	0.0392	N/A

Table 2.2: Ablation Study in PubMed Dataset

our architecture. Specifically, we select a single large heterogeneous walk length - 8 to replace the heterogeneous walk length 1, 2, and 3 in our model, and the resulting model is called HGEN-S. We also independently remove the heterogeneous node embedding to let the generator uniformly sample the next node, and the resulting model is named HGEN-E. Lastly, we replace the heterogeneous graph assembler with a probability-based graph assembler, namely HGEN-A.

As shown in Table 2.2, all ablative models perform similarly on node-level metrics like Powerlaw Coefficient and Assortativity, as HGEN effectively captures this information through heterogeneous walk distributions. Key observations include: *1) HGEN-S*: Although it constructs larger sub-graphs due to longer heterogeneous walks, it fails to improve the capture of heterogeneous structural information. Longer meta-

paths are often redundant, as they share common sub-parts [154]. Hence, we limit meta-path lengths to 1, 2, and 3 for flexibility and efficiency. *2) HGEN-E*: Removing heterogeneous node embeddings hampers local graph structure capture. This is because HGEN relies on neighborhood-encoded information to make node sampling structure-aware. *3) HGEN-A*: Replacing the heterogeneous graph assembler with a probabilistic one degrades performance in preserving node degree distributions. Probabilistic assemblers uniformly sample edges from generated walks, ignoring meta-paths. In contrast, HGEN leverages meta-paths as fundamental units for edge sampling, ensuring effective preservation of meta-path distributions (Theorem 1) and maintaining node degree distributions across types.

Running Time Comparison

The results of our running time experiments are shown in Figure 2.6. The running times on both synthetic and real-world datasets including both training and inference time are shown with respect to the growth of number of nodes in both synthetic and real-world datasets. All running times are in $log10$ scale. As shown in both figures, random-walk-based generative models (HGEN and NetGAN) have a constant running time growth in terms of number of nodes, which is especially important when dealing with large graphs. Even though VGAE is much faster regarding running time, it is indeed a representation learning framework based on GCN and lacks of the ability to generate realistic heterogeneous graphs, and the results are also reflected in Table 2.1. Both GraphRNN and GraphVAE fail to compare with HGEN in model scalability because their designs require at least $O(|\mathcal{V}|^2)$ to process the transformed node sequence and adjacency matrix.

2.1.7 Conclusion

In this paper, we propose a novel framework - HGEN for heterogeneous graph generation, which can jointly capture the semantic, structural, and global distributions of heterogeneous graphs. Our framework consists of a novel heterogeneous walk generator that can hierarchically generate meta-path instances (namely heterogeneous walk) and a heterogeneous graph assembler that can construct new graphs by sampling from the generated heterogeneous walks in a stratified manner. Extensive experiments on synthetic and real-world datasets demonstrate the advantages of HGEN over existing deep generative models in terms of preserving both graph statistical and heterogeneous specified properties.

2.2 Source Localization for Cross Network Information Diffusion

Source localization seeks to identify the origins of information diffusion in networks, a crucial inverse problem in understanding information propagation. This task has both practical and theoretical significance, as accurately pinpointing sources can help mitigate the spread of misinformation by disrupting critical pathways. Early works [137, 203, 173, 216] employed *rule-based* methods, which rely on predefined diffusion patterns. More recently, *learning-based* approaches [42, 110, 172] utilize deep neural networks to encode neighborhood and graph topology information, achieving state-of-the-art performance. These advances highlight the vital role of source localization in ensuring the integrity of information in digital networks.

Existing source localization techniques focus on single networks, yet many real-world systems operate as *cross-networks*. These include cross-community communications, cross-border financial transactions, and interconnected supply chains. Cross-networks introduce unique risks, such as misinformation spreading across social me-

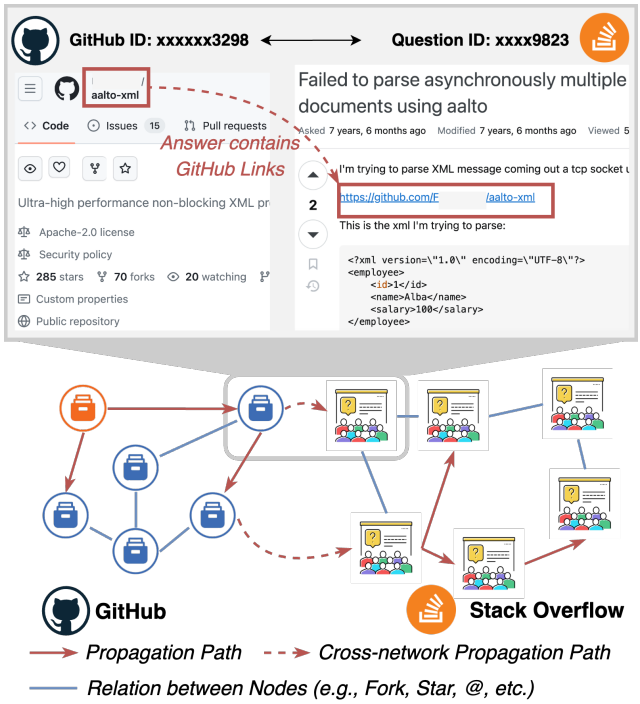


Figure 2.7: Example of misinformation propagation on cross-network between GitHub and Stack Overflow, where each node in the GitHub network denotes a repository, and each node in the Stack Overflow represents a discussion thread.

dia platforms or safety issues propagating in supply chains. For example, a malicious GitHub repository (Figure 2.7) linked to over 40 Stack Overflow threads illustrates the challenge. Less experienced users may unknowingly adopt risky solutions, potentially disrupting their systems [73]. Tracing misinformation in cross-networks is difficult, as propagation often starts in one network (GitHub) and is observed in another (Stack Overflow), involving multiple rounds of dissemination. This highlights the need for advanced source localization methods tailored to cross-network environments.

Cross-network source localization involves identifying diffusion sources in a source network using only the observed diffusion in a target network, a task that remains underexplored. This problem is particularly challenging due to the network separation, which renders traditional methods ineffective. Key obstacles include: *1) Characterizing diffusion source distribution.* Understanding the distribution of potential sources is essential for modeling diffusion processes and quantifying uncertain-

ties [110, 111, 30]. In a cross-network scenario, this requires a conditional probability model that considers the structural and dynamical properties of both networks. However, diverse topologies, node features, and propagation patterns complicate the learning objective. *2) Capturing dynamic and static node features jointly.* Effective source localization depends on both intrinsic node attributes and their connections. Incorporating node features such as text descriptions or statistical data results in high-dimensional, complex distributions. Moreover, the distinct characteristics of nodes across networks further complicate modeling their diffusion dynamics. *3) Modeling heterogeneous diffusion patterns.* Beyond source distribution, capturing the diverse propagation patterns in both networks is critical. This includes accounting for cross-network propagation paths (Figure 2.7), which play a significant role in the diffusion process.

In this work, we propose the Cross-Network Source Localization (CNSL) method for locating the diffusion sources from a source network given its diffused observation from another target network under arbitrary diffusion patterns. Specifically, for the first challenge, we design a novel framework to approximate the distribution of diffusion sources by mean-field variational inference. For the second challenge, we propose a disentangled generative prior to encoding both static and dynamic features of nodes. For the last challenge, we model the unique diffusion dynamics of each network separately and integrate the learning process of these information diffusion models with the approximation of diffusion source distribution. This ensures accurate reconstruction of diffusion sources considering the specific propagation mechanisms of each network. We summarize our major contributions of this work as follows:

- **Problem**. We design a novel formulation of the cross-network source localization and propose to leverage deep generative models to characterize the prior and approximate the distribution of diffusion sources via variational inference.

- **Technique**. We propose a unified framework to jointly capture 1) both static

and dynamic node features, and 2) the heterogeneous diffusion patterns of both networks. The approximation of diffusion sources is fully aware of various node features and the interplay of cross-network information diffusion patterns.

- **Data**. Cross-network source localization lacks high-quality data, which is highly difficult to craft. We collect and curate a real-world dataset that accounts for the Cross-platform Communication Network, which records the real-world misinformation propagation from Github to Stack Overflow. We also provide a simulated cross-network dataset using agent-based simulation to disseminate misinformation across physical and social networks.

- **Experiments**. We conduct experiments against state-of-the-art methods designed originally for single-network source localization. Results show substantially improved performance of our method for cross-network source localization.

2.2.1 Related Works

Information Source Localization. Diffusion source localization is defined as inferring the initial diffusion sources given the current diffused observation, which has attracted many applications, ranging from identifying rumor sources in social networks [69] to finding blackout origins in smart grids [146]. Early approaches [137, 215, 216, 173] focused on identifying the single/multiple source of an online disease under the Susceptible-Infected (SI) or Susceptible-Infected-Recover (SIR) diffusion patterns with either full or partial observation. Later on, Dong et al. [42] further leverage GNN to enhance the prediction accuracy of LPSI. However, existing diffusion source localization methods cannot well quantify the uncertainty between different diffusion source candidates, and they usually require searching over the high-dimensional graph topology and node attributes to detect the sources, both drawbacks limit their effectiveness and efficiency. Moreover, in the past few years, more methods

[110, 172, 138, 184] have been proposed to address the dependency of prescribed diffusion models and characterize the latent distribution of diffusion sources, which have achieved state-of-the-art results. However, their methods still may not generalize to cross-network source localization due to the unique interconnected structure.

Information Diffusion on Cross Network. The interconnection between cross-networks allows information to flow seamlessly from one platform to another through overlapping nodes. However, it is important to note that the patterns of influence and information propagation differ between various networks and can even vary within the same network. Recent studies in information diffusion across interconnected networks have made notable advancements. Earlier works [81, 185, 35, 106] have developed different frameworks for correct modeling of the information flow within different network formats, such as wireless networks, social networks, and supply chains. Later on, many works have been proposed to study different features and applications of cross-networks, e.g., mitigating cascading failures [161, 52]. However, until today, there are few works [40, 107] trying to correctly model the information diffusion pattern in the interconnected network system.

2.2.2 Cross-network Diffusion Source Localization

In this section, the problem formulation is first provided before deriving the overall objective from the perspective of the divergence-based variational inference. A novel optimization algorithm is then proposed to infer the seed nodes given the observed cross-network diffused pattern.

Problem Formulation

Cross-network $\mathcal{G} = (G_s, G_t)$ consists of a *Source Network* $G_s = (V_s, E_s)$ and a *Target Network* $G_t = (V_t, E_t)$. Both G_s and G_t are composed of a set of vertices V_s and V_t

corresponding to individual users of the network as well as a set of edges $E_s \subseteq V_s \times V_s$ and $E_t \subseteq V_t \times V_t$ denote connecting pairs of users in both networks, respectively. In addition, $f_s \in \mathbb{R}^{N_s \times F}$ and $f_t \in \mathbb{R}^{N_t \times F}$ and denote the static features of both networks (e.g., associated text embedding, user age, social relations, etc.), where F denotes the dimension of the node feature, and N_t, N_s denote the number of nodes in each network, respectively. To connect the cross-network $\mathcal{G}$, there exists a set of bridge links between G_s and G_t denoted by $L = \{(v_s, v_t) | v_s \in V_s, v_t \in V_t\}$, which represent the propagation paths from G_s to G_t.

The information propagation from G_s to G_t follows *one-directional* message passing. More specifically, the propagation initiates from a group of nodes denoted as $x_s \in \{0,1\}^{N_s}$ in the source network G_s, where each entry has a binary value representing whether the node is seed or not. After a certain time, the information propagates from G_s to G_t and infects a portion of nodes in G_t through the bridge links L. We use $y_t \in [0,1]^{N_t}$ to denote the infection probability of each node in G_t.

Problem 2 (**Cross-network diffusion source localization**)**.** Given $\mathcal{G}$ and the diffused observation of the target network y_t, the problem of diffusion source localization in cross-networks (i.e., the inverse problem of diffusion estimation) requires finding $\tilde{x}_s \in \{0,1\}^{N_s}$ from the source network G_s, such that the empirical loss $\|\tilde{x}_s - x_s\|_2^2$ is minimized, under the constraint that the diffused observation in the target graph y_t could be generated from $\tilde{x}_s$ through L.

However, reconstructing $\tilde{x}_s$ from y_t is difficult due to the following challenges. *1) The difficulty of characterizing the distribution of seed nodes in the cross-network scenario.* To consider all possibilities of the seed nodes in cross-network source localization, it's desired to model the distribution of seed nodes $p(x_s)$ by characterizing the conditional probability $p(x_s|y_t)$. However, learning $p(x_s|y_t)$ requires jointly considering the topology structure of the cross-network $\mathcal{G}$ as well as the stochastic diffusion pattern through bridge links L. Existing works cannot be directly adapted due to

the incapability of considering the complex cross-network scenario. *2) The difficulty of jointly capturing dynamic and static features of the nodes in the cross-network.* The intrinsic patterns of the seed nodes consist of both dynamic patterns (i.e., the choice of seed nodes x_s) and static patterns (e.g., node features f_s). The correlated factors lead to the high-dimensional and often intractable distribution $p(x_s)$, which makes maximizing the joint likelihood $p(x_s, y_t)$ to be hard and computationally inefficient. *3) The difficulty of jointly capturing the heterogeneous diffusion patterns of the cross-network.* The underlying diffusion process from x_s to y_t is not only affected by numerous factors (e.g., the infectiousness of the misinformation and the immunity power of the user), but the propagation patterns in the cross-network are inherently different in different networks.

Latent Distribution Learning of Seed Nodes

To cope with the first challenge of characterizing the distribution of diffusion sources in the cross-network, we propose to utilize graph topology as well as the diffused observation to define the conditional probability $p(x_s|y_t)$. Since the diffused observation y_t is conditioned on both networks $\mathcal{G}$ as well as the diffusion source x_s, we derive a conditional probability $p(y_t|x_s, \mathcal{G}) \cdot p(x_s)$, where $p(x_s)$ is the distribution of infection sources within G_s. To estimate the optimal diffusion source $\tilde{x}_s$, we employ the Maximum A Posteriori (MAP) approximation by maximizing the following probability:

$$\tilde{x}_s = \arg\max_{x_s} \; p(y_t|x_s, \mathcal{G}) \cdot p(x_s) = \arg\max_{x_s} \; p(x_s, y_t|\mathcal{G}).$$

However, since $p(x_s)$ is often intractable and entangles both static and dynamic features, we instead leverage deep generative models to characterize the implicit distribution of $p(x_s)$.

To tackle the second challenge of jointly considering all static and dynamic node

features, we propose a disentangled generative model to map the intractable and potentially high-dimensional $p(x_s)$ to latent embeddings in low-dimensional latent space. Specifically, we aim to learn the conditional distribution $p(x_s, y_t, \mathcal{G}|z_s, z_{fs})$ of x_s given two latent variables z_s and z_{fs}. Specifically, $z_s \in \mathbb{R}^{k_1}$ ($k_1 \ll N_s$) and $z_{fs} \in \mathbb{R}^{k_2}$ ($k_2 \ll N_s$) are obtained by an approximate posterior $p(z_s, z_{fs}|x_s, y_t, \mathcal{G})$, where $p(z_s, z_{fs})$ is the prior distribution of node's dynamic and static features. Note that k_1 and k_2 are the numbers of variables in each group, in order to capture the different types of semantic factors.

The goal here is to learn the conditional distribution of $p(x_s)$ given $Z = (z_s, z_{fs})$, namely, to maximize the marginal likelihood of the observed cross-network diffusion in expectation over the distribution of the latent variable set Z as $\mathbb{E}_{p_\theta(Z)}[p_\theta(x_s, y_t, \mathcal{G}|Z)]$. For a given observation of the information diffusion in the cross-network, the prior distribution of the latent representations $p(Z)$ is still intractable to infer. We propose solving it based on variational inference, where the posterior needs to be approximated by the distribution $q_\phi(Z|x_s, y_t, f_s, \mathcal{G})$. In this way, the goal becomes to minimize the Kullback–Leibler (KL) divergence between the true prior and the approximate posterior. Moreover, we assume z_s and z_{fs} capture different semantic factors. Specifically, z_s is required to capture just the independent dynamic semantic factors of which nodes are infection sources, and z_{fs} is required to capture the correlated semantic factors considering both dynamic features and static node features. To encourage this disentangling property of both posteriors, we introduce a constraint by trying to match the inferred posterior configurations of the latent factors to the prior $p(z_s, z_{fs})$ by setting each prior to being an isotropic unit Gaussian $\mathcal{N}(0, 1)$, leading to the constrained optimization problem as:

$$\begin{aligned} \max_{\theta,\phi} \quad & \mathbb{E}_{q_\phi(z_s, z_{fs}|x_s, y_t, \mathcal{G})}\left[p_\theta\left(x_s, y_t, \mathcal{G}|z_s, z_{fs}\right)\right], \\ \text{s.t.} \quad & KL\left[q_\phi(z_s, z_{fs}|x_s, f_s, y_t, \mathcal{G})||p(z_s, z_{fs})\right] < I. \end{aligned}$$

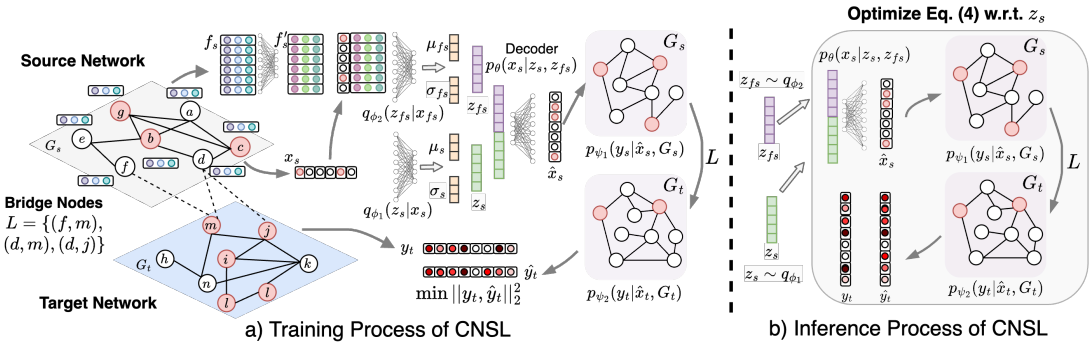


Figure 2.8: The training pipeline of CNSL contains three steps: 1) q_{ϕ_1} and q_{ϕ_2} approximate the distribution of $p(z_s, z_{fs})$ in a disentangled manner; 2) the inferred latent variables z_s and z_{fs} are concatenated to reconstruct $\hat{x}_s$; 3) the reconstructed $\hat{x}_s$ is leveraged as initial seed nodes to initiate the cross-network information propagation and predict expected diffusion $\hat{y}_t$.

Furthermore, assuming $p(z_s)$ represents the distribution of dynamic node features and $p(z_{fs})$ denotes the distribution of joint node features (entangles with both static and dynamic features), the constraint term can be decomposed as:

$$q_\phi(z_s, z_{fs}|x_s, f_s, y_t, \mathcal{G}) = q_{\phi_1}(z_s|x_s, y_t, \mathcal{G}) \cdot q_{\phi_2}(z_{fs}|y_t, f_s, x_s, \mathcal{G})$$

Then the objective function can be written as:

$$\max_{\theta,\phi} \quad \mathbb{E}_{q_\phi(z_s, z_{fs}|x_s, y_t, \mathcal{G})} \left[p_\theta \left(x_s, y_t, \mathcal{G}|z_s, z_{fs} \right) \right], \tag{2.3}$$

$$\text{s.t.} \quad KL \left[q_{\phi_1}(z_s|x_s, y_t, \mathcal{G}) || p(z_s) \right] < I_s, \quad KL \left[q_{\phi_2}(z_{fs}|y_t, x_s, \mathcal{G}) || p(z_{fs}) \right] < I_{fs},$$

where we decompose I into two separate parts (i.e., I_s and I_{fs}) of the information capacity to control each group of latent variables so that the variables inside each group of latent variables are disentangled. In practice, $q_{\phi_1}(\cdot)$ and $q_{\phi_2}(\cdot)$ are implemented as two encoders with multi-layer perceptron structure. More details can be found in Figure 2.8.

Cross Network Diffusion Model Learning

To address the third challenge, i.e., making the source localization be aware of the heterogeneous diffusion patterns between networks, locating diffusion origins x_s may not only involve estimating the distribution of seed nodes but the process should also be determined by correctly modeling the information diffusion across diverse and interlinked network structures $\mathcal{G}$. In the context of cross-network information diffusion, the diffused observation y_t is determined by the diffusion source x_s under the cross-network $\mathcal{G}$ through bridge links L. Therefore, the conditional distribution $p_\theta(x_s, y_t, x_f, \mathcal{G}|z_s, z_{fs})$ can further be decoupled as:

$$\log p_\theta(x_s, y_t, x_f, \mathcal{G}|z_s, z_{fs}) = \log[p_\psi(y_t|x_s, \mathcal{G})] + \log[p_\theta(x_s|z_s, z_{fs})],$$

where $p_\psi(\cdot)$ models the probability of the infection status y_t of nodes in G_t given seed nodes x_s in G_s. Moreover, the second term $p_\theta(x_s|z_s, z_{fs})$ reveals that the latent variables Z only encodes information from x (i.e., $y_t \perp Z|x_s, \mathcal{G}$). According to the assumption, we could also simplify both encoders as $q_{\phi_1}(z_s|x_s, \mathcal{G})$ and $q_{\phi_2}(z_{fs}|x_s, f_s, \mathcal{G})$ in Eq. (2.3) by removing y_t from the input.

Cross-network Information Propagation. Modeling the diffusion from x_s to y_t is complex due to multiple factors, such as misinformation's infectiousness and the distinct propagation patterns across networks like GitHub and Stack Overflow, which cater to different user communities. The unknown nature of these diffusion patterns prevents the use of standard models like Linear Threshold or Independent Cascade. This complexity underlines the need to decompose and simplify $p_\psi(y_t|x_s, \mathcal{G})$ to analyze the diverse diffusion behaviors in G_s and G_t through a learning approach:

$$\log p_\psi(y_t|x_s, \mathcal{G}) = \log p_{\psi_1}(y_s|x_s, G_s) + \log p_{\psi_2}(y_t|y_s, x_t, G_t). \tag{2.4}$$

Note that $\log p_\psi(y_t|x_s, G_s, G_t) = \log[\sum_{x_t} p_{\psi_1}(x_t|x_s, G_s) \cdot p_{\psi_2}(y_t|x_t, G_t)]$, where x_t inherited infection probability from y_s. In practice, we assume $p_{\psi_1}(x_t|x_s, G_s)$ follows delta distribution, where only the x_t is 1 that corresponds to the x_s and the rest of x_t's are 0. This property is also assumed in many works [141] using VAE. Therefore, $\log p_\psi(y_t|x_s, G_s, G_t)$ is simplified as Eq. (2.4). In this simplified decomposition, $p_{\psi_1}(\cdot)$ characterizes the diffusion pattern of G_s given the seed nodes x_s, which is independent of the information propagation in G_t. $y_s \in [0, 1]^{N_s}$ records the infection status of all nodes in the source network G_s. When the diffusion is complete in G_s, the infection probability is directly transferred to the target network G_t through bridge links L so that some nodes in G_t have initial infection status (denoted as x_t) to initiate the infection process in G_t. The propagation in G_t is then modeled by $p_{\psi_2}(y_t|y_s, x_t, G_t)$ by taking the graph structure G_t and initial seed infection probability x_t as inputs. More details of the derivation are provided in the Appendix.

Monotonic Constraint on Information Diffusion. The information diffusion on the regular network is often regularized by the monotone increasing property [39, 110]. In this work, we also assume the same monotonic property holds in the cross-network information diffusion, namely $y_t^{(i)} \succeq y_t^{(j)}, \forall\, x_s^{(i)} \supseteq x_s^{(j)}$. Specifically, selecting more seed nodes in G_s would result in a generally higher (or at least equal) infection probability of nodes in G_s according to the property of diminishing returns. Subsequently, the bridge links would transfer the infection probability from y_s to x_t, and similarly, the probability of each node being infected in $y_t^{(i)}$ (estimated from $x_t^{(i)}$) should be greater or equal to $y_t^{(j)}$ (estimated from $x_t^{(j)}$), such that $y_t^{(i)} \succeq y_t^{(j)}$. Therefore, owing to the monotonic increasing property of the information diffusion, we add the constraint $\lambda \left\| \max(0, y_t^{(j)} - y_t^{(i)}) \right\|_2^2, \forall\, x_s^{(i)} \supseteq x_s^{(j)}$, to Eq. (2.3), where we transform the inequality constraint into its augmented Lagrangian form to minimize $\left\| \max(0, y_t^{(j)} - y_t^{(i)}) \right\|_2^2$ and $\lambda > 0$ denotes regularization hyperparameter.

Overall Objective for Training. The training procedure of the proposed CNSL model is coupled with Eq. (2.3), Eq. (2.4), and the monotonic increasing constraint:

$$\begin{aligned}
\mathcal{L}_{\text{train}} &= \max_{\theta,\phi_1,\phi_2} \mathbb{E}_{q_\phi}\left[p_\theta(x_s, y_t, x_f, \mathcal{G}|z_s, z_{fs})\right], \qquad (2.5)\\
&\text{s.t.} \quad KL\left[q_{\phi_1}(z_s|x_s,\mathcal{G})||p(z_s)\right] < I_s, \quad KL\left[q_{\phi_2}(z_{fs}|x_s,f_s,\mathcal{G})||p(z_{fs})\right] < I_{fs},\\
&\qquad y_t^{(i)} \succeq y_t^{(j)},\ \forall\, x_s^{(i)} \supseteq x_s^{(j)},\\
&= \min_{\theta,\phi_1,\phi_2,\psi_1,\psi_2} -\mathbb{E}_{q_\phi}\big[\log p_\theta(x_s|z_s,z_{fs}) + \log p_{\psi_1}(y_s|x_s,G_s) + \log p_{\psi_2}(y_t|y_s,x_t,G_t)\big],\\
&\text{s.t.} \quad KL\left[q_{\phi_1}(z_s|x_s,\mathcal{G})||p(z_s)\right] < I_s, \quad KL\left[q_{\phi_2}(z_{fs}|x_s,f_s,\mathcal{G})||p(z_{fs})\right] < I_{fs},\\
&\qquad \left\|\max(0, y_t^{(j)} - y_t^{(i)})\right\|_2^2,
\end{aligned}$$

where we only need to sample one $x_s^{(i)}$ and many $x_s^{(j)}$'s (such that $x_s^{(i)} \supseteq x_s^{(j)}$) as training samples for each mini-batch. The $y_t^{(i)}$ and $y_t^{(j)}$'s are estimated by arbitrary diffusion patterns. For simplicity, we omit the subscript of $\mathbb{E}_{q_\phi(z_s,z_{fs}|x_s,y_t,\mathcal{G})}$ as $\mathbb{E}_{q_\phi}$ when the context is clear. The overall framework is summarized in Figure 2.8.

Cross-network Seed Set Inference

Upon training completion, the joint probability $p(z_s, z_{fs})$ is approximated by the posterior $q_\phi(z_s, z_{fs}|x_s, f_s, y_t, \mathcal{G})$. Both $p_{\psi_1}(\cdot)$ and $p_{\psi_2}(\cdot)$ effectively classify the diffusion patterns across networks. This study introduces a sampling method for $\tilde{x}_s \sim p(x_s)$ by marginalizing over $p(z_s) \cdot p(z_{fs})$ to conduct MAP estimation, where $p(x_s) = \sum_{z_s}\sum_{z_{fs}} p_\theta(x_s|z_s, z_{fs})p(z_s, z_{fs})$. However, marginalizing the standard Gaussian prior $p(z_s, z_{fs})$ necessitates extensive sampling to align the sample distribution with the target distribution, increasing computational load. Additionally, it is also hard to sample individual latent variables from the joint distribution of $p(z_s, z_{fs})$. To cope with both challenges, we consider the density over the inferred latent variables induced by the approximate posterior inference mechanism, and we propose the following objective w.r.t. z_s to infer $\tilde{x}_s$ in an optimized manner. Specifically, the inference objective

function $\mathcal{L}_{\text{pred}}$ is written as:

$$\begin{aligned}\mathcal{L}_{\text{pred}} &= \max_{z_s} \mathbb{E}\left[p_\psi(y_t|x_s,\mathcal{G})\cdot p_\theta(x_s|z_s,z_{fs})\right], \\ &\text{s.t.} \quad z_s \sim q_{\phi_1}(z_s|\hat{x}_s,\mathcal{G}), \quad z_{fs} \sim q_{\phi_2}(z_{fs}|\hat{x}_s,f_s,\mathcal{G}), \\ &= \min_{z_s} -\mathbb{E}\Big[\log p_\psi(y_t|x_s,\mathcal{G}) + \log\big[\sum\nolimits_{z_s}\sum\nolimits_{\hat{x}_s} p_\theta(x_s|z_s,z_{fs})\big]\Big] \\ &\text{s.t.} \quad z_s \sim q_{\phi_1}(z_s|\hat{x}_s,\mathcal{G}), \quad z_{fs} \sim q_{\phi_2}(z_{fs}|\hat{x}_s,f_s,\mathcal{G}),\end{aligned} \tag{2.6}$$

where we sample many $\hat{x}_s$ from the training set, and obtain equal amount of z_s from $q_{\phi_1}(\cdot)$. Note that we optimize z_s (dynamic latent variable) only, instead of both z_s and z_{fs} (static-dynamic entangled latent variable), which is rooted in the specific roles these variables play in the model. z_s is targeted for optimization because it encodes dynamic information crucial for identifying better seed nodes in the context of information diffusion. This dynamic aspect is mutable and can be optimized to improve source localization accuracy. On the other hand, z_{fs} entangles both dynamic and static information, where the static part represents unchangeable node features. Optimizing z_{fs} would be less efficient because static features, by their nature, cannot be optimized. The optimization process aims to adjust variables to improve model performance, but since static features remain constant, attempting to optimize z_{fs} would not enhance the model's ability to localize diffusion sources.

Implementation of the Seed Set Inference. We provide implementation details of the overall inference process here. Specifically, the inference framework first samples k seed node set $\hat{x}_s$ from the training set, and we can take the average value $\bar{z}_s$ and $\bar{z}_{fs}$ from the learned latent distributions by taking k different $\hat{x}_s$ as input:

$$\bar{z}_s = \frac{1}{k}\sum\nolimits_i^k q_{\phi_1}(z_s|\hat{x}_s^{(i)},\mathcal{G}), \bar{z}_{fs} = \frac{1}{k}\sum\nolimits_i^k q_{\phi_2}(z_s|\hat{x}_s^{(i)},f_s,\mathcal{G}). \tag{2.7}$$

We concatenate $\bar{z}_s$ and $\bar{z}_{fs}$ as input to minimize the inference loss in Eq. 2.6. The latent variable z_s is iteratively optimized according to the inference objective function to minimize $-\log p_\psi(y_t|x_s, \mathcal{G})$. In practice, Eq. (2.6) cannot be optimized directly, we thus provide a practical version of the inference objective function: since the diffused observation y_t fits the Gaussian distribution and the seed set x_s fits the Bernoulli distribution, we can simplify Eq. (2.15) as:

$$\mathcal{L}_{\text{pred}} = \min_{z_s} - \Big[\log \big[\prod_{i=0}^{N_s} f_\theta(z_s^{(i)}, z_{fs}^{(i)})^{x_s^{(i)}} (1 - f_\theta(z_s^{(i)}, z_{fs}^{(i)})^{1-x_s^{(i)}} \big] + \big\| \tilde{y}_t - y_t \big\|_2^2 \Big] \quad (2.8)$$

where the $\tilde{y}$ is given as the optimal influence spread (i.e., $\tilde{y}_t = N_t$). In other words, the inference objective is guided by the discrepancy between the inferred y_t and the ground truth $\tilde{y}_t$. We visualize the overall inference procedure in Figure 2.8 (b). Specifically, we sample $\bar{z}_{fs}$ and $\bar{z}_s$, according to Eq. (2.7), and leverage $p_\theta(\cdot)$ to decode $\hat{x}_s$. The predicted $\hat{x}_s$ is leveraged to initiate the cross-network diffusion and predict $\hat{y}_t$. The optimization supervision consists of 1) the mean squared loss between $\hat{y}_t$ and the ground truth y_t as well as 2) the probability of node v_i being seed node $f_\theta(z_s^{(i)}, z_{fs}^{(i)}) \in [0, 1]$.

2.2.3 Experimental Evaluation

This section reports both qualitative and quantitative experiments that are carried out to test the performance of CNSL and its extensions on a simulated dataset that simulates the spread of misinformation across a city-level population and a collected real-world cross-network dataset obtained by crawling two online networking platforms and cross-references between them.

Real-world Dataset: Cross-Platform Communication Network

We collected real-world data from GitHub and Stack Overflow to form the cross-platform communication network, where information flows from GitHub to Stack Overflow since many posts in Stack Overflow have mentioned or discussed GitHub Repositories when addressing users' questions. We started by downloading the Stack Overflow public data dump provided by the Internet Archive. Then, we extracted all the Stack Overflow posts where their post texts contain a URL to GitHub (i.e., $439,753$ posts mapping to $439,753$ repositories). We further built the Stack Overflow network by finding the question posts, answer posts, and related posts of the current $439,753$ posts. This yielded a total of $1,410,600$ Stack Overflow posts, encompassing data from 2008 up to 2023.

To obtain the GitHub network, we expanded our initial GitHub network by finding all GitHub repositories that the existing repositories depended upon. We utilized an open-source tool[1], which uses the GitHub GraphQL API to obtain the dependency information. The resulting GitHub network contains $533,240$ repositories. For our experiment, we sampled GitHub repositories from the year 2021 and their dependent repositories from the year before 2021 (i.e., $1,204$ nodes and $1,043$ edges). We then found their corresponding Stack Overflow posts (i.e., $3,862$ nodes and $3,149$ edges). We obtained the ground truth in a pseudo-setting: we randomly sampled 10% of the GitHub nodes as seed nodes, and simulated their diffusion process within the GitHub network and the Stack Overflow network (i.e., 120 GitHub seed nodes, 354 GitHub infected nodes, 195 Stack Overflow seed nodes, and 482 Stack Overflow infected nodes).

[1] `https://github.com/edsu/xkcd2347`

Simulated Dataset: Agent-Based Geo-Social Information Spread

We leverage an agent-based simulation framework based on realistic Patterns of Life [84**?** , 6, 7] to simulate the spread of misinformation across social and physical networks. In this simulation, an agent represents a simulated individual who commutes to their workplace, eats at restaurants, and meets friends and recreational sites. Inspired by the Theory of Planned Behavior [2] and Maslow's Hierarchy of Needs [126] as theories of human behavior, agents are driven by physiological needs to eat and have shelter, safety needs such as financial stability requiring them to go to work, and needs for love requiring them to meet friends and build and maintain a social network. Details of the theories of social science informing this simulation are found in [219] and details to use this simulation for data generation are described in [5].

We augmented this simulation framework to simulate the spread of misinformation using a simple Susceptible-Infectious disease model. The simulation is initialized with $15,000$ agents. A small number of n (by default, $n = 5$) agents are selected randomly as the sources of misinformation and flagged as "Infectious" and all other agents are initially flagged as "Susceptible". Agents can spread misinformation in two ways: 1) through collocation, allowing an agent to spread the misinformation in-person to other agents located at the same workplace, restaurant, or recreational site, and 2) through the social network, allowing an agent to spread misinformation to their friends regardless of their location. To allow the generation of large datasets for source localization, each spreading of misinformation is stopped after five simulation days. At this time, the following datasets are recorded:

- *Ground Truth.* The set of n agents that were initially seeded with the misinformation.
- *Misinformation Spread.* The list of agents to whom the misinformation has spread after five days.

- *The Complete Co-location Network.* This network captures the agents who meet each other and thus, may spread misinformation through co-location.

- *The Observed Co-location Network.* This network is a randomly sampled subset of agents from the complete co-location network. It represents the agents in the complete co-location network that are parts of the simulated location tracking. This network is used to simulate the realistic case of not having access to the location data of every individual.

- *The Complete Social Network.* This network records the friend and family connections of all agents which may infect each other through social contagion.

- *The Observed Social Network.* This network includes a randomly sampled subset of agents from the complete social network and simulates the social media environment. This network simulates the realistic case where an observed social media network may not capture the entire population.

- *Cross-Network Links through Identity.* Links between the two observed networks are defined through identity. Any individual agent in the co-location network is (trivially) connected to itself in the social network.

Once this data is collected, the misinformation spread status of all agents is set to "Susceptible" and n new agents are selected as the seed nodes of a new case of misinformation. This process of creating new cases of misinformation is iterated every five simulation days to create an unlimited number of realistic datasets of information spread across the physical and social spaces.

For the dataset used for the following experiments, there are $5,281$ agents and $8,276$ edges in the observed co-location network, and $5,669$ agents and $17,948$ edges in the observed social network. Each case of misinformation spread yields between 50-200 agents to which the misinformation spreads after five days. This synthetic dataset

allows us to capture realistic misinformation spread across both networks. Due to some agents not being captured in the two networks, this dataset allows us to simulate the realistic case where misinformation may spread outside of the observed networks. We provide the code for our agent-based misinformation simulation framework in a GitHub repository[2], which also contains the generated dataset used in this work.

Experiment Setup

Implementation Details. We employ a two-layer MLP for learning node features, which are concatenated with the seed vector in the subsequent stage before being input to the encoder $q_{\phi_2}(\cdot)$. Both encoders ($q_{\phi_1}(\cdot)$, $q_{\phi_2}(\cdot)$) and the decoder $p_\theta(\cdot)$ utilize three-layer MLPs with non-linear transformations. We use GNN model architecture coupled with a two-layer MLP network as the aggregation network with 64 hidden units for the two propagation models ($p_{\psi_1}(\cdot)$ and $p_{\psi_2}(\cdot)$). The learning rates for encoder-decoder, $p_{\psi_1}(\cdot)$, and $p_{\psi_2}(\cdot)$ are set to 0.0001, 0.005, and 0.01 respectively in a multi-optimization manner. Additionally, the number of epochs is 15 for all datasets, with a batch size of 2. The iteration numbers for inference are set to 2 for all datasets.

Comparison Methods. We illustrate the performance of CNSL in various experiments against two sets of methods: 1) **Rule-based methods**: *LPSI* [173] predicts the rumor sources based on the convergent node labels without the requirement of knowing the underlying information propagation model; *OJC* [216] aims at locating sources in networks with partial observations, which has strength in detecting network sources under the SIR diffusion pattern. 2) **Learning-based methods**: *GCNSI* [42] learns latent node embedding with GCN to identify multiple rumor sources close to the actual source; *IVGD* [168] propose a graph residual model to make existing graph diffusion models invertible; *SL-VAE* [110] proposed to learn the graph diffusion model with a generative model to characterize the distribution of diffusion sources. DDMSL

[2] `https://github.com/Siruiruirui/misinformation`

[187] proposed a diffusion model-based source localization method to recover each diffusion step iteratively. Note that existing comparison methods are not designed for cross-network source localization, in order to conduct a fair comparison, we repeated each model separately for two networks and learned the two networks. We used bridge links L to connect these two models.

Evaluation Metrics. Source localization is a classification task so we use two main metrics to evaluate the performance of our proposed model: *1). F1-Score (F1)* and *2). ROC-AUC Curve (AUC)*, as they are classical metrics for classification tasks. since most real-world scenarios tend to have an imbalance between the number of diffusion sources and non-source nodes (fewer diffusion sources), we additionally leverage PR@100 to evaluate the precision of the top-100 prediction returned by models.

Quantitative Analysis

We evaluated the models in different diffusion configurations. For the cross-platform communication data, the underlying diffusions are LT (Table 2.3) and IC (Table 2.4) for the first network which was followed by other three diffusion patterns (LT, IC, and SIS) for the second network in each case. For the Geo-Social information spread data (Table 2.5), the underlying diffusion pattern has been explained in Section 2.2.3. For that dataset, we used two different simulations (A and B) and also used two different types of seed selections. Here $D0$ considers the initial sources of misinformation as seed nodes and $D1$ considers the initial sources of misinformation and the infected agents on the first day as seed nodes.

Performance in the cross-platform communication network. Table 2.3 shows that CNSL excels others across all metrics and diffusion patterns. In the first network with LT diffusion pattern (LT2LT, LT2IC, LT2SIS), CNSL achieves the highest recall (RE) in all scenarios, with scores of 0.996, 0.997, and 0.997, respectively, indicating its

		LT2LT				LT2IC				LT2SIS			
Category	**Method**	**PR**	**RE**	**F1**	**AUC**	**PR**	**RE**	**F1**	**AUC**	**PR**	**RE**	**F1**	**AUC**
Rule-based	LPSI	0.156	0.841	0.263	0.583	0.141	0.849	0.242	0.533	0.079	0.942	0.127	0.497
	OJC	0.104	0.035	0.052	0.500	0.116	0.036	0.054	0.502	0.113	0.036	0.053	0.501
Learning based	GCNSI	0.103	0.858	0.184	0.636	0.103	0.866	0.184	0.622	0.114	0.801	0.199	0.635
	IVGD	0.228	0.948	0.368	0.139	0.227	0.874	0.359	0.138	0.123	0.985	0.215	0.240
	SL-VAE	0.249	0.947	0.395	0.703	0.192	0.847	0.313	0.689	0.242	0.931	0.385	0.612
	DDMSL	0.251	0.923	0.394	0.815	0.309	0.845	0.454	0.732	0.320	0.842	0.464	0.772
Our Method	CNSL	**0.332**	**0.996**	**0.498**	**0.888**	**0.332**	**0.997**	**0.498**	**0.889**	**0.332**	**0.997**	**0.498**	**0.890**
	CNSL-W/O	0.103	0.922	0.185	0.520	0.103	0.930	0.186	0.511	0.103	0.917	0.186	0.517

Table 2.3: Performance comparison for cross-platform communication network under LT diffusion pattern for the first network with LT, IC, and SIS diffusion pattern for the second network.

		IC2LT				IC2IC				IC2SIS			
Category	**Method**	**PR**	**RE**	**F1**	**AUC**	**PR**	**RE**	**F1**	**AUC**	**PR**	**RE**	**F1**	**AUC**
Rule-based	LPSI	0.124	0.868	0.217	0.489	0.215	0.657	0.324	0.562	0.129	0.906	0.226	0.522
	OJC	0.117	0.032	0.050	0.503	0.097	0.027	0.042	0.499	0.115	0.032	0.050	0.502
Learning based	GCNSI	0.142	0.638	0.233	0.623	0.170	0.476	0.251	0.627	0.152	0.602	0.243	0.630
	IVGD	0.120	0.979	0.210	0.733	**0.548**	0.391	0.083	0.439	0.115	0.825	0.195	0.733
	SL-VAE	0.254	0.881	0.394	0.719	0.195	0.909	0.321	0.703	0.185	0.829	0.302	0.592
	DDMSL	0.286	0.827	0.425	0.818	0.318	0.886	0.468	0.753	0.270	0.833	0.408	0.689
Our Method	CNSL	**0.333**	**0.990**	**0.498**	**0.887**	0.333	**0.998**	**0.499**	**0.891**	**0.332**	**0.997**	**0.498**	**0.888**
	CNSL-W/O	0.103	0.922	0.186	0.514	0.103	0.935	0.185	0.515	0.103	0.928	0.185	0.516

Table 2.4: Performance comparison for cross-platform communication network under IC diffusion pattern for first network with LT, IC, and SIS diffusion pattern for the second network.

superior ability to identify all relevant instances in the dataset. Additionally, CNSL also exhibits the best precision (PR) in LT2LT and LT2IC scenarios, and competitive precision in the LT2SIS scenario. The F1 scores, which balance precision and recall, are also highest for CNSL, peaking at 0.498 in both LT2LT and LT2IC patterns, demonstrating the method's overall efficiency and accuracy. The AUC scores for CNSL are robust, ranking highest in LT2LT and LT2SIS scenarios, signifying excellent model performance across various threshold settings. In the Table 2.4 first network with IC diffusion pattern (IC2LT, IC2IC, IC2SIS), CNSL's performance remains impressive, maintaining the highest recall scores of 0.990, 0.998, and 0.997, respectively. CNSL also boasts the highest F1 scores in all scenarios, with a notable 0.499 in IC2IC, suggesting a balanced performance between precision and recall. The AUC scores for CNSL are again the highest, with 0.887 in IC2LT and 0.891 in IC2IC, indicating its strong discriminative ability. Overall, CNSL demonstrates considerable

		G2S-A-D0				G2S-B-D0				G2S-A-D1				G2S-B-D1			
Category	**Method**	**PR**	**RE**	**F1**	**AUC**	**PR**	**RE**	**F1**	**AUC**	**PR**	**RE**	**F1**	**AUC**	**PR**	**RE**	**F1**	**AUC**
Rule-based	LPSI	0.147	0.982	0.256	0.512	0.165	0.954	0.281	0.609	0.152	0.903	0.260	0.475	0.224	0.973	0.364	0.578
	OJC	0.053	0.018	0.022	0.496	0.125	0.039	0.051	0.507	0.063	0.040	0.043	0.497	0.115	0.058	0.071	0.505
Learning based	GCNSI	0.123	**1.000**	0.216	0.744	0.117	**1.000**	0.207	0.351	0.183	**1.000**	0.300	0.250	0.221	**1.000**	0.341	0.193
	IVGD	0.139	**1.000**	0.244	0.502	0.138	**1.000**	0.242	0.500	0.218	**1.000**	0.352	0.490	0.266	**1.000**	0.409	0.500
	SL-VAE	0.364	0.863	0.512	0.707	0.289	0.788	0.423	0.611	0.289	0.754	0.418	0.664	0.425	0.893	0.576	0.725
Our Method	CNSL	**0.481**	0.816	**0.605**	**0.931**	**0.452**	0.885	**0.598**	**0.933**	**0.499**	0.779	**0.609**	**0.894**	**0.539**	0.987	**0.698**	**0.901**
	CNSL-W/O S	0.122	1.000	0.219	0.503	0.117	1.000	0.2101	0.488	0.183	0.998	0.309	0.499	0.221	0.999	0.362	0.501

Table 2.5: Performance comparison for Geo-Social Information Spread Data (G2S) for two types (A, B) of simulation. Here $D0$ considers the initial sources of misinformation as seed nodes and $D1$ considers the initial sources of misinformation and the infected agents at the first day as seed nodes.

strength in reliably identifying relevant instances across different diffusion patterns and networks, while maintaining high precision and excellent area under the ROC.

Performance in geo-social information spread data. In Table 2.5, the performance of various methods on Geo-Social Information Spread Data (G2S) is evaluated for two simulation types, A and B, with two different seeding strategies, D0 and D1. Our method, CNSL, exhibits strong performance across all scenarios. In the G2S-A-D0 simulation, CNSL achieves a high precision (PR) of 0.481, showing its effectiveness in correctly identifying misinformation spread. It also has the highest F1 score of 0.605 and an AUC of 0.931, indicating a balanced precision-recall trade-off and excellent model discrimination ability, respectively. For the G2S-B-D0 simulation, CNSL's precision (0.452) and F1 score (0.598) are notable, and the AUC of 0.933 is the highest compared to other methods, suggesting CNSL's consistency and reliability. In the G2S-A-D1 scenario, CNSL maintains a high recall (RE) of 0.779 and an impressive AUC of 0.894, which signifies its capacity to identify true misinformation cases effectively when the seeding includes infected agents from the first day. Remarkably, in the G2S-B-D1 scenario, CNSL stands out with the highest precision (0.539) and F1 score (0.698), and it achieves an outstanding AUC of 0.901. This demonstrates CNSL's superior ability to differentiate between misinformation and non-misinformation spread, especially when the initial condition includes both sources of misinformation and infected agents. The recall of 0.987 in this scenario

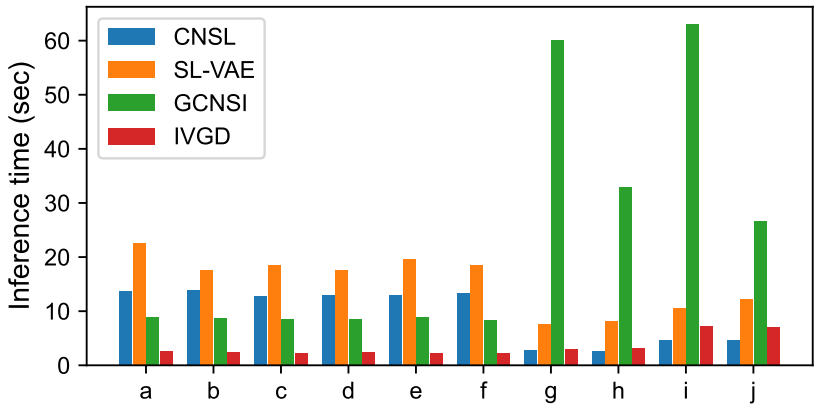


Figure 2.9: Runtime Comparison with learning based methods for dataset a) LT2LT, b) LT2IC c) LT2SIS, d) IC2LT, e) IC2IC, f) IC2SIS, g) G2S-A-D0, h) G2S-A-D1, i) G2S-B-D0, j) G2S-B-D1

also indicates that CNSL can identify nearly all instances of misinformation spread. Overall, the CNSL method outperforms both rule and learning-based methods in most metrics across different simulations and seeding strategies in geo-social networks.

Runtime Analysis. Figure 2.9 presents a runtime comparison among four learning-based methods: CNSL, SL-VAE, GCNSI, and IVGD across ten different diffusion configurations (a to j). CNSL, which is our method, shows a competitive inference time in all datasets when compared to the SL-VAE. In cross-platform communication network datasets (a) LT2LT, b) LT2IC, c) LT2SIS, d) IC2LT, e) IC2IC, and f) IC2SIS)), CNSL demonstrates an inference time that is neither the fastest nor the slowest, indicating a balanced computational demand for these more complex scenarios. However, in datasets geo-social information spread data (g) G2S-A-D0, h)G2S-A-D1, i)G2S-B-D0, and j)G2S-B-D1), CNSL's runtime is noticeably lower, suggesting that while CNSL is highly effective in identifying misinformation spread. Overall, CNSL shows a strength in providing a good balance between accuracy and computational efficiency. While there are scenarios where CNSL's runtime is higher, these may correlate with more complex network conditions where deeper analysis is necessary, which CNSL seems to handle without compromising the quality. This

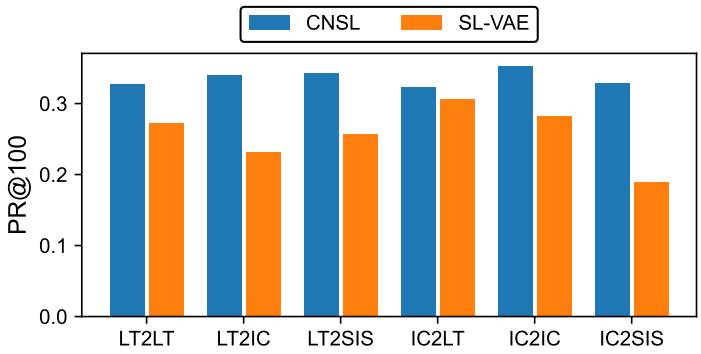


Figure 2.10: Precision@100: the precision rate of the top 100 nodes being predicted as seed nodes. The comparison is conducted between our method: CNSL and the current state-of-the-art: SL-VAE.

makes CNSL a robust method for practical applications where runtime is a critical factor alongside precision and accuracy.

Precision analysis at top 100 nodes predicted by models. Figure 2.10 illustrates the precision at top 100 (PR@100) comparison between CNSL and the state-of-the-art SL-VAE across various diffusion patterns. PR@100 measures the precision rate of the top 100 nodes predicted as seed nodes, indicating how accurately each method can identify the most influential nodes in the spread of information or misinformation. CNSL shows a strong performance in this metric, outperforming SL-VAE in all diffusion patterns. CNSL exhibits higher PR@100 rates, indicating that it is more precise in identifying the key seed nodes. This precision is crucial in scenarios where it is important to quickly and accurately pinpoint the main drivers of information spread within a network. Notably, CNSL's precision suggests that its algorithm is particularly adept at handling complex diffusion patterns where the identification of influential nodes is more challenging. The strength of CNSL, as highlighted by Figure 2.10, lies in its ability to rank the most relevant nodes higher than SL-VAE consistently. The precision at the top 100 nodes is essential for practical applications where interventions need to be targeted and efficient, such as in the case of misinformation containment or viral marketing.

2.2.4 Conclusion

In conclusion, information diffusion source localization on cross-networks requires locating the origins of information diffusion within and across networks. We propose a Cross-Network Source Localization (CNSL) framework in this work, which stands as a pivotal advancement in addressing the complexities introduced by cross-network environments, where traditional source localization methods fall short. By ingeniously approximating the distribution of diffusion sources through mean-field variational inference, encoding both static and dynamic features of nodes via a disentangled generative prior, and uniquely modeling the diffusion dynamics of interconnected networks, CNSL offers a comprehensive solution to the problem. Extensive experiments, including quantitative analysis, case studies, and runtime analysis, have been conducted to verify the effectiveness of the framework across different real-world and synthetic cross-networks. The significance of this work lies not only in its methodological innovation but also in its practical implications for safeguarding the integrity and reliability of information in an increasingly interconnected digital world.

2.3 Deep Graph Representation Learning and Optimization for Influence Maximization

As one of the fundamental research problems in network analysis, the objective of Influence Maximization (IM) is to find a set of seed nodes that maximizes the spread of influence in a social network. IM has been extensively studied in recent years due to its large commercial value. For example, consider the case of viral marketing [25] for promoting a commercial product, where a company may wish to spread the adoption of a new product from some initially selected users, the selected initial users are expected to spread the information about the product on their respective social networks. IM has become the cornerstone in many critical applications such as

network monitoring [170], misinformation containment [195], and friend recommendation [198].

As a typical type of combinatorial optimization problem, retrieving a (near) optimal seed set to maximize the influence in a network is challenging due to the stochastic nature of information diffusion and the hardness of the problem. Traditional (non-learning-based) methods for IM [90, 80, 158, 159, 132, 144] have made great progress in the last decade, and Li et al. [95] have even achieved exact solutions under specific diffusion models. The commonality of traditional methods is the explicit requirement of the information diffusion model as the model input. However, the real-world information diffusion process is complex and cannot be simply modeled by prescribed diffusion models. With the recent development of machine/deep learning, it is natural to consider a learning-based way to characterize the underlying diffusion process.

While great progress has been made in the field, current efforts on learning-based IM solutions are still in the infancy stage due to fundamental obstacles as follows. *1). The difficulty of efficiently optimizing the objective function.* Learning-based IM methods tend to solve the discrete problem in continuous space by mostly leveraging deep network representations [204, 88] and deep reinforcement learning [160, 93]. Even though they could attain a competitive performance with traditional methods, their scalability and execution efficiency are problematic due to *(a)* the need to iteratively update all node embeddings at each action and *(b)* the #P-hardness of computing the influence spread [103]. *2). The difficulty of automatically identifying and modeling the actual diffusion process.* To maximize the influence spread in a network, the underlying information diffusion pattern is an imperative part as it determines the overall information propagation outcome. However, both traditional and learning-based methods cannot characterize the underlying diffusion process without heuristics. To work around this, both traditional and current learning-based methods have been leveraging pre-defined diffusion models (e.g., Linear Threshold (LT) and

Independent Cascade (IC)) as the input to solve the combinatorial optimization problem. Although they could work well only for the process following their heuristics, the real-world network process is way more complex than the heuristics and largely unknown. *3). the difficulty of adapting solutions to various node-centrality-constrained IM problems.* There are a lot of variants of IM that relate to node centrality, e.g., the constraint on the number of seed nodes, the constraint on the total degree of seed nodes, etc. Current learning-based IM solutions do not have a well-defined paradigm for solving different node-centrality-constrained IM problems, which poses another challenge to their solution adaptivity.

To address the above challenges, we propose a novel framework - DeepIM, to solve the IM problem by developing a novel strategy that embeds the initial discrete optimization domain into a larger continuous space. Remarkably, we propose to learn the latent representation of seed sets by retaining their expressiveness and directly optimizing in the continuous space to reduce the problem's hardness. We further design a learning-based diffusion model to characterize the underlying diffusion dynamics in an end-to-end manner. Moreover, we develop a generic seed set inference framework to directly optimize and generate set embeddings under a uniform budget constraint. Finally, we summarize our contributions as follows:

- **Problem.** We formulate the learning-based IM problem as embedding the initial discrete optimization domain into continuous space for easing the optimization and identify its unique challenges arising from real applications.
- **Framework.** We propose modeling the representation of the seed set in a latent space, and the representation is jointly trained with the model that learns the underlying graph diffusion process in an end-to-end manner.
- **Adaptivity.** We propose a novel constrained optimization objective function to infer the optimal seed set by leveraging deep graph embeddings, which can

be applied under arbitrary node-centrality-related constraints.

- **Evaluation.** We conduct extensive experiments over four real-world datasets to demonstrate the performance of the proposed method. Compared with other state-of-the-art in various application scenarios, DeepIM achieves the best results in finding a seed set to maximize the influence.

2.3.1 Related Work

Learning-based Influence Maximization

influence Maximization (IM), initially framed as a combinatorial optimization problem by Kempe et al. [80], has spurred significant research and applications over the past decade. Traditional IM methods fall into three categories: simulation-based, proxy-based, and heuristic-based. These approaches have achieved efficient and accurate solutions under specific diffusion models. While Du et al. [45], Vaswani et al. [164] explored learning influence from cascade data, they still relied on predefined diffusion models, such as the Coverage function. For a comprehensive overview, see recent surveys [97, 13].

Learning-based methods address traditional IM limitations, particularly the lack of generalization. Early works [101, 3] integrated reinforcement learning (RL) with IM, inspiring numerous RL-based approaches. Current state-of-the-art methods [92, 160, 125, 93, 23] typically learn latent embeddings of nodes or networks, treating node embeddings as agent states to select seed nodes, with marginal influence gain as the reward. Beyond RL, some methods [88, 79, 135] use graph neural networks to encode social influence into node embeddings for node selection. However, learning-based IM methods face challenges in handling diverse diffusion patterns, ensuring solution quality, and achieving scalability comparable to traditional methods.

Graph Neural Network

Graph Neural Networks (GNNs) [178] are a class of deep learning methods designed to perform inference on data described by graphs. The general paradigm of GNNs alternates between node feature transformation and neighbor nodes' information aggregation. For a K-layer GNN, a node aggregates information within K-hop neighbors. Specifically, the k-th layer transformation is:

$$a^k = \mathcal{A}^k(h^{k-1}; \theta^k), h^k = \mathcal{C}^k(a^k; \theta^k), \forall 1 \leq k \leq K. \tag{2.9}$$

where a^k is an aggregated feature, and h^k is the k-th layer node feature. The flexibility of aggregation function $\mathcal{A}(\cdot)$ and combine function $\mathcal{C}(\cdot)$ functions induces different GNN models [165, 85, 183]. The high-level representations of nodes or graphs are utilized for different tasks. In this work, we leverage GNN to characterize the underlying diffusion pattern and construct an end-to-end model for estimating the influence.

2.3.2 Problem Formulation

Given a graph $G = \{V, E\}$, the problem of IM aims to maximize the number of influenced nodes in G by selecting an optimal seed node set $\mathbf{x} \subseteq V$. Particularly, the evaluation of IM relies on an influence diffusion model parametrized by θ: $\mathbf{y} = M(\mathbf{x}, G; \theta)$, where θ can be the set of infection probability on each node if $M(\cdot)$ is an independent cascade model or the set of parameters in the aggregation/combine functions if $M(\cdot)$ is GNN-based. We denote $\mathbf{x} \in \{0, 1\}^{|V|}$ as the vector representation of the source node set, where the i-th element $x_i = 1, x_i \in \mathbf{x}$ if $v_i \in \mathbf{x}$ and $x_i = 0$ otherwise. The output $y \in \mathbb{R}_+$ measures the total number of infected nodes. Based on the formalization of the influence spread, the IM problem is defined as follows:

Definition 1 (Influence Maximization)**.** The generic *IM* problem requires selecting

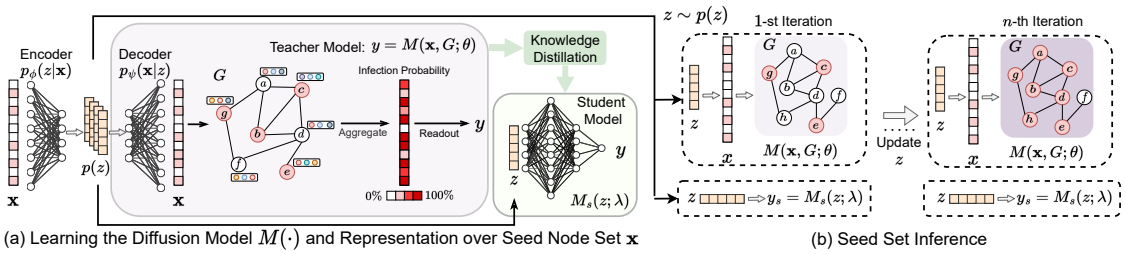


Figure 2.11: DeepIM consists of two parts. a) we leverage the autoencoder to learn and compress the latent distribution of seed node sets into lower dimension $p(z)$. b) the seed set inference scheme iteratively optimizes the proposed objective function by updating the latent variable z to maximize the influence spread.

a set of k users from V as the seed set to maximize the influence spread:

$$\tilde{\mathbf{x}} = \arg\max_{|\mathbf{x}| \leq k} M(\mathbf{x}, G; \theta), \tag{2.10}$$

where $\tilde{\mathbf{x}}$ is the optimal seed node set to produce maximal influence spread in G.

2.3.3 DeepIM

In this section, we propose the DeepIM framework to ease the computational overhead of the learning-based IM methods and automatically identify the underlying diffusion patterns. The framework can be divided into two phases: the *learning phase* is leveraged to characterize the probability of the observed seed set and model the underlying information propagation distribution, and the *inference phase* is employed to optimize the selection of seeds in continuous space to maximize the influence spread.

2.3.4 Learning Representation of Seed Set

To build an effective and efficient objective function, we propose to characterize the probability of the seed node set $p(\mathbf{x})$ over $\mathbf{x}$ given the graph G since learning $p(\mathbf{x})$ can help depict the seed set's underlying nature. However, learning such probability is not a trivial task because different nodes are inter-connected within each seed

set and highly correlated based on the topology of G. These connections make the relationship between nodes very complex and harder to decipher than other similar combinatorial problems.

Learning Probability over Seed Nodes. Instead of directly modeling the highly-intractable probability $p(\mathbf{x})$, we introduce an unobserved latent variable z to represent $\mathbf{x}$ and define a conditional distribution $p(\mathbf{x}|z)$ to quantify the likelihood. These latent variables have much lower dimensions than the observed sub-optimal seed sets, which can yield a compressed representation. Particularly, we marginalize over the latent variables to obtain $p(\mathbf{x})$: $p(\mathbf{x}) = \int p(\mathbf{x}, z)\, dz = \int p(\mathbf{x}|z)p(z)\, dz$. The posterior likelihood $p(z|\mathbf{x}) = p(\mathbf{x}|z)p(z)/p(\mathbf{x})$ allows us to infer the latent variables given the observed seed sets. In this work, we adopt autoencoder to generatively infer the posterior, where both encoder f_ϕ (parameterized by ϕ) and decoder f_ψ (parameterized by ψ) are used to characterize the likelihood of both posterior and conditional distribution, respectively. We aim to maximize the joint likelihood:

$$\max_{\phi,\psi} \ \mathbb{E}\big[p_\psi(\mathbf{x}|z) \cdot p_\phi(z|\mathbf{x})\big]. \tag{2.11}$$

Learning the End-to-end Diffusion Model. Once we have learned the latent distribution of seed nodes $p(\mathbf{x})$, the next step is to update the seed node set $\mathbf{x}$ in order to increase the marginal gain of the influence spread. Current learning-based IM solutions still assume the computation of the influence spread (i.e., $M(\mathbf{x}, G; \theta)$) relies on prescribed mathematical models. However, real-world information diffusion is complicated, and it is not easy to determine the most suitable diffusion model in practice. A chosen diffusion model may be misspecified compared to real-world data and lead to large model bias. In addition, the diffusion network structure can be also hidden from us, so we need to learn not only the parameters in the diffusion model but also the diffusion network structure [45].

In this work, we design a GNN-based diffusion model $M(\cdot)$ for accurate modeling the relationship between $\mathbf{x}$ and y with considering the overall graph topology. The output of a GNN-based diffusion function $M(\cdot)$ is composed of two functions $M = g_r \circ g_u(\mathbf{x}, G; \theta)$: 1) $\tau = g_u(\mathbf{x}, G; \theta)$, where $g_u(\cdot)$ is a GNN-based aggregation function and $\tau \in [0,1]^{|V|}$ is an intermediate output after aggregating multi-hop neighborhood information. τ denotes the *infection probability* of each node; and 2) $y = g_r(\tau; \xi), y \in \mathbb{R}_+$ denotes the final information spread, where $g_r(\cdot)$ is a normalization function (e.g., l-1 norm) and ξ is the threshold of transforming the probability into discrete value. The GNN-based $M(\cdot)$ is visualized in Figure 2.11 (a).

Definition 2 (**Score Monotonicity and Infection Monotonicity**)**.** Given a GNN-based diffusion model $M(\cdot) : 2^{|V|} \to \mathbb{R}_+$ and any two subsets $S, T \subseteq V$, $M(\cdot)$ is score monotonic if $x_S \preceq x_T$ (i.e. $S \subseteq T$) implies $M(\mathbf{x}_S, G; \theta) \leq M(\mathbf{x}_T, G; \theta)$, where $\mathbf{x}_S, \mathbf{x}_T \in \{0,1\}^{|V|}$ are vector representations of seed sets S and T, respectively. $M(\cdot)$ is infection monotonic if $x_S \preceq x_T$ (i.e. $S \subseteq T$) implies $\tau_S \preceq \tau_T$, where $\tau_S, \tau_T \in [0,1]^{|V|}$ denote the infection probability of seed sets S and T, respectively.

Monotonicity is a natural property for us in modeling the overall diffusion network structure. A monotonic diffusion model indicates the spread of influence would continue to increase. Intuitively, if we select a larger community $\mathbf{x}'$ as the seed set, the larger $\mathbf{x}'$ would intrinsically infect no fewer nodes in the whole network than a smaller seed set $\mathbf{x}$ if $\mathbf{x} \preceq \mathbf{x}'$. Ensuring the property of both monotonicities allows us to better characterize the underlying diffusion network structure and mimic the real-world diffusion pattern [41]. Hence, we add constraints to make the GNN-based diffusion model $M(\mathbf{x}, G; \theta)$ monotonic during the influence spread estimation.

Theorem 2.3.1 (**Monotonicity of GNN Models**)**.** *For any GNN-based* $M(\mathbf{x}, G; \theta) = g_r \circ g_u(\mathbf{x}, G; \theta)$*, where* $g_u(\mathbf{x}, G; \theta)$ *is formulated by Eq.* (2.9)*,* M *is score and infection monotonic if* $\mathcal{A}^k$ *and* $C^k, k \in [1, K]$*, are non-decreasing in Eq.* (2.9)*, and* g_r *is also non-decreasing.*

We further illustrate that the well-known Graph ATtention network (GAT) can be score and infection monotonic under the constraint we claimed in Theorem 2.3.1.

Corollary 2.3.2 (**Montonicity of GAT**). *M is score and infection monotonic when g_u is GAT if $\theta^k \geq 0$ in Eq. (2.9) and g_r is also non-decreasing.*

The proof of Theorem 2.3.1 and Corollary 2.3.2 are provided in Appendix. According to Theorem 2.3.1 and Corollary 2.3.2, the GNN-based $M(\mathbf{x}, G; \theta)$ has the theoretical guarantee to retain monotonicity, and the objective of learning the GNN-based $M(\mathbf{x}, G; \theta)$ is given as maximizing the following probability with a constraint:

$$\max_\theta \mathbb{E}\big[p_\theta(y|\mathbf{x}, G)\big], \quad \text{s.t. } \theta \geq 0. \tag{2.12}$$

Knowledge Distillation for Diffusion Estimation Efficiency. We have learnt the deep representation of seed nodes and an end-to-end diffusion model with a monotonicity guarantee. However, we empirically find the calculation of influence spread $M(\mathbf{x}, G; \theta)$ involves three steps: 1) decoding a node vector $\mathbf{x}$ from the learned posterior $p(\mathbf{x}|z)$; and 2) executing the GNN-based diffusion model $M(\mathbf{x}, G; \theta)$ under the graph G; and 3) normalizing the probabilistic output τ from $M(\mathbf{x}, G; \theta)$ to actual influence spread y. Even though the prediction results are accurate, the computational overhead is still a burden when dealing with million-scale networks. Inspired by recent research on knowledge distillation, we propose to leverage a small yet powerful student model supervised by $M(\mathbf{x}, G; \theta)$ to attain efficiency. Specifically, the student model $M_s(z; \lambda)$ is a lightweight neural network parametrized by λ that directly takes the latent variable z sampled from the learned $p(z)$ as input. $M_s(z; \lambda)$ directly returns the estimated influence spread y_s as output. The distillation loss between the $y = M(\mathbf{x}, G; \theta)$ (teacher model) and $y_s = M_s(z; \lambda)$ can be as simple as $\|y - y_s\|_2^2$.

End-to-end Learning Objective. Finally, in order to bridge the representation learning and the learning of the diffusion model, we propose a unified objective function in an end-to-end manner by putting together Eq. (2.11) and (2.12) as:

$$\mathcal{L}_{\text{train}} = \max_{\theta,\lambda,\psi,\phi} \mathbb{E}\big[p_\theta(y|\mathbf{x}, G) \cdot p_\lambda(y_s|z) \cdot p_\psi(\mathbf{x}|z) \cdot p_\phi(z|\mathbf{x})\big], \text{ s.t. } \theta \geq 0. \tag{2.13}$$

However, optimizing the expectation of joint probabilities could be computationally difficult. We instead derive the negative log term of Eq. (2.13) and derive its lower bound as the final learning objective according to Jensen's inequality:

$$\begin{aligned}
\mathcal{L}_{\text{train}} &= \min_{\theta,\lambda,\psi,\phi} -\log\Big[\mathbb{E}\big[p_\theta(y|\mathbf{x}, G) \cdot p_\lambda(y_s|z) \cdot p_\psi(\mathbf{x}|z) \cdot p_\phi(z|\mathbf{x})\big]\Big], \text{ s.t. } \theta \geq 0. \\
&\geq \min_{\theta,\lambda,\psi,\phi} \mathbb{E}\Big[-\log\big[p_\theta(y|\mathbf{x}, G) \cdot p_\lambda(y_s|z) \cdot p_\psi(\mathbf{x}|z) \cdot (p_\phi(z|\mathbf{x})\big]\Big], \text{s.t. } \theta \geq 0. \\
&= \min_{\theta,\lambda,\psi,\phi} \mathbb{E}\Big[-\log\big[p_\theta(y|\mathbf{x}, G)\big] - \log\big[p_\lambda(y_s|z)\big] - \log\big[p_\psi(\mathbf{x}|z) \cdot (p_\phi(z|\mathbf{x})\big]\Big], \text{s.t. } \theta \geq 0.
\end{aligned} \tag{2.14}$$

The overall objective consists of minimizing the empirical error $-\log[p_\theta(y|\mathbf{x}, G)]$ of the prediction of y with the reconstructed $\mathbf{x}$ as input and minimizing the reconstruction error. In addition, we minimize the distillation loss $-\log[p_\lambda(y_s|z)]$ to train the student model along with the overall training process. The overall framework for the training of end-to-end diffusion models and the autoencoder for learning the seed set distribution is visualized in Figure 2.11 (a).

2.3.5 Seed Node Set Inference

To infer the high-influential seed node set in the testing domain, we leverage the latent distribution $p(\mathbf{x})$ of the seed node set and the end-to-end diffusion model $M(\cdot)$ jointly from Eq. (2.14). Firstly, if the autoencoder is well trained and can retain both *continuity* (i.e., two close points in the latent space should not give two completely

different contents once decoded) and *completeness* (i.e., for a chosen distribution, a point sampled from the latent space should give "meaningful" content once decoded), the autoencoder in Eq. (2.11) can generate contents by exploiting the latent feature space $p(z)$ learned from all the examples it was trained from, i.e., $p(\mathbf{x})$. Therefore, we propose to alternatively search the optimal seed node set $\tilde{x}$ in the lower-dimensional and less-noisy latent space $p(z)$. The following corollary demonstrates it is equivalent to estimating the influence spread with the latent variable z rather than high-dimensional $\mathbf{x}$ if the autoencoder retains both continuity and completeness.

Corollary 2.3.3 (**Influence Estimation Consistency**). *For any* $M(f_\psi(z^{(i)}), G; \theta) > M(f_\psi(z^{(j)}), G; \theta)$*, we have* $M(x^{(i)}, G; \theta) > M(x^{(j)}, G; \theta)$.

The proof of Corollary 2.3.3 can be found in Appendix. According to the corollary, we could find the optimal seed set that can generate the maximal influence by optimizing z in the following joint probability: $\max_z \mathbb{E}\big[p_\theta(y|\mathbf{x}, G) \cdot p_\psi(\mathbf{x}|z)\big]$.

Adaptation to Different IM Variants with Node Centrality Constraints. Since the introduction of IM in [80], IM has been studied under various budget-constrained settings on nodes in recent years. To enhance the adaptivity of DeepIM, we design a unified constraint that allows inferring seed sets under various budgets on individual nodes. Specifically, the objective $\mathcal{L}_{\text{pred}}$ is given as:

$$\mathcal{L}_{\text{pred}} = \max_z \mathbb{E}\big[p_\theta(y|\mathbf{x}, G) \cdot p_\psi(\mathbf{x}|z)\big], \text{ s.t. } \sum\nolimits_{i=0}^{|V|} \mathcal{F}(v_i, G) \cdot x_i \leq k, \tag{2.15}$$

where $\sum_{i=0}^{|V|} \mathcal{F}(v_i, G) \cdot x_i$ is a generalized budget constraint applied on individual nodes, and k is the actual budget. For the vanilla IM problem that only requires selecting a given number of seed nodes, $\sum_{i=0}^{|V|} \mathcal{F}(v_i, G) \cdot x_i$ can be derived as $\|x \cdot \mathbf{1}\|_1$, where the $\mathbf{1} \in \{1\}^{N \times 1}$ is an all-one vector indicating the price of selecting each node are the same. In addition, for node degree constrained IM problems [90, 133], $\sum_{i=0}^{|V|} \mathcal{F}(v_i, G) \cdot x_i$ can

Algorithm 1: DeepIM Prediction Framework

Input: $\mathcal{L}_{\text{pred}}$; decoder function $f_\psi(\cdot)$; regularization function $\Phi(\cdot)$; number of training instances N; the number of iteration η; learning rate α.

1: $z = 1/N \sum_{i=0}^{N} f_\psi(\mathbf{x})$ {$\mathbf{x}$ sampled from training set.}
2: **for** $i = 0, ..., \eta$ **do**
3: $\quad \mathbf{x} \leftarrow f_\psi(z)$ {seed set $\mathbf{x}$.}
4: $\quad \mathbf{x} \leftarrow \Phi(\mathbf{x})$ {Regularize the seed set x into a valid region in terms of different constraints.}
5: $\quad z \leftarrow z - \alpha \cdot \nabla \mathcal{L}_{\text{pred}}(\mathbf{x}, z)$
6: **end for**
7: $\tilde{\mathbf{x}} \leftarrow \Phi(f_\psi(z))$ {Output the final $\tilde{\mathbf{x}}$ by decoding the final optimized z.}

be derived as $\|x \cdot A\|_1$, where $A \in \{0,1\}^{N \times N}$ is the adjacency matrix of the network G, and $\|\mathbf{x} \cdot A^i\|_1 \leq k$ represents the $l1$-norm of the total seed node degree is bounded by a budget k. The budget constraint $\mathcal{F}(v_i, G) \cdot x_i \leq k$ can also be easily designed, combined, and adapted to solve the IM variants with non-uniform prices on nodes.

Implementation Details of the Seed Set Inference. We visualize our inference procedure in Figure 2.11 (b). Specifically, the inference framework first samples a latent variable z from the learned latent distribution $p(z)$. The latent variable z is iteratively optimized according to the inference objective function Eq. (2.15) to attain a larger marginal gain (influence spread). Note that the learning-based diffusion model $p_\theta(y|\mathbf{x}, G)$ can be switched between the student diffusion model $M_s(z; \lambda)$ and the GNN-based diffusion model $M(\mathbf{x}, G; \theta)$ to achieve either efficiency or efficacy. In addition, the constrained objective function Eq. (2.15) cannot be computed directly so we provide a practical version of the inference objective function: since the diffused observation y fits the Gaussian distribution and the seed set $\mathbf{x}$ fits the Bernoulli distribution, we can simplify Eq. (2.15) as:

$$\begin{aligned} \mathcal{L}_{\text{pred}} = \min_z \Big[& -\log \big[\prod_{i=0}^{|V|} f_\psi(z_i)^{x_i} (1 - f_\psi(z_i)^{1-x_i} \big] \\ & + \big\| \tilde{y} - y \big\|_2^2 \Big] \text{ s.t. } \sum_{i=0}^{|V|} \mathcal{F}(v_i, G) \cdot x_i \leq k, \end{aligned} \tag{2.16}$$

where the $\tilde{y}$ is given as the optimal influence spread (i.e., $\tilde{y} = |V|$), and the full derivation of the above equation is provided in Appendix. Furthermore, we utilize the Projected Gradient Descent and propose a regularization function $\Phi(\mathbf{x})$ to keep the predicted seed set $\mathbf{x}$ in a valid region in terms of different constraints. For example, $\Phi(\mathbf{x})$ can be defined as selecting k nodes with the highest probabilities when the price of selecting each node is equal in Eq. (2.15). $\Phi(\mathbf{x})$ can also be defined as cost-efficiently selecting the top-k nodes from $\mathbf{x}/c(\mathbf{x})$, where $c(\mathbf{x})$ denotes the budget on one node (e.g., node degree). Finally, The optimization procedure is summarized in Algorithm 1. Specifically, we first sample an initial latent variable z on Line 1. From Line 2 - 6, we iteratively solve the optimization problem proposed in Eq. (2.15) via gradient descent optimizer (e.g., Adam) while regularizing the predicted seed set in a valid region with $\Phi(\cdot)$. Figure 2.11 (b) illustrates the overall process of the inference objective learning. The derivation details of both Eq. (2.14) and (2.16) are provided in the Appendix.

2.3.6 Experiment

In this section, we compare the performance of our proposed DeepIM framework across six real networks in maximizing the influence under various settings, following a case study to qualitatively demonstrate the performance of DeepIM.

Experiment Setup

Our primary purpose is to evaluate the expected influence spread as defined in Eq. (2.10) under various IM application scenarios. Since DeepIM can be easily adapted to different diffusion patterns, we choose two representative models that are commonly used in the IM problem, i.e., LT and IC models.

	Digg	Weibo	Power Grid	Network Science	Cora-ML	Jazz	Synthetic
Nodes	279,613	2,251,166	4,941	1,565	2,810	198	50,000
Edges	1,170,689	225,877,808	6,594	13,532	7,981	2,742	250,000

Table 2.6: The Overview of Dataset

Data. The proposed DeepIM is compared with other approaches over six real-world datasets, including Cora-ML, Network Science, Power Grid, Jazz, Digg, and Weibo. We also adopt a synthetic dataset that is a random graph with $50,000$ nodes generated by Erdos-Renyi algorithm [46]. The statistics of the data are shown in Table 2.6. We randomly sample seed node set x, and the seed size is proportional to $|V|$ of each network. We then use IC, LT, and SIS models to compute the final influence spread y. The $\{(x, y)\}$ set then serves as the training set of our algorithm.

Comparison Method.

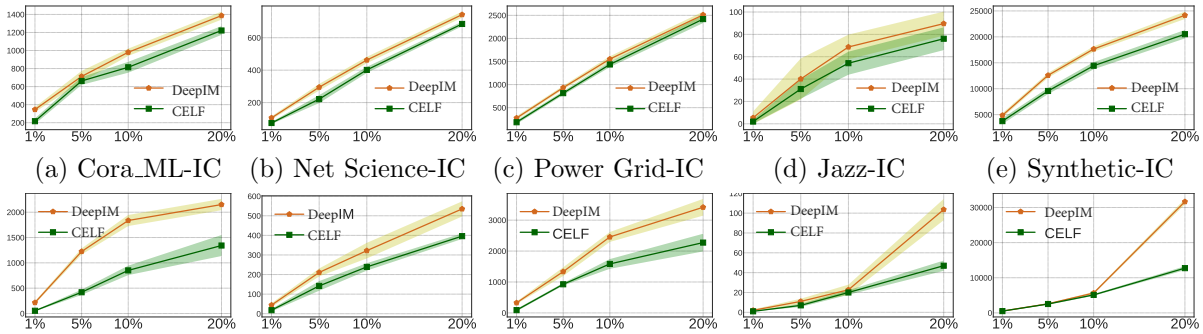
In addition to comparing our model's performance between the GNN-based diffusion model $M(x, G; \theta)$ (denoted as DeepIM) and the student diffusion model $M_s(z; \lambda)$ (denoted as DeepIM$_s$), we also adopt four sets of comparison methods, all of which are outlined as follows. *Traditional IM*: 1) Greedy [80], 2) IMM [159], 3) OPIM-C [157], and 4) SubSIM [57]. *Learning-based IM*: 1) IMINFECTOR [135], 2) PIANO [93], and 3) ToupleGDD [23]. *Online IM*: OIM [157]. *Budget-constraint IM*: CELF [90]. Other than the four sets of baselines, we also show the performance of our student model DeepIM$_s$ (i.e., DeepIM coupled with the simplified student diffusion model $M_s(\cdot)$).

Quantitative Analysis

We evaluate the performance of DeepIM in maximizing the influence against other approaches under various IM application schemes. Each model selects 1%, 5%, 10%, and 20% nodes in each dataset as seed nodes, and we allow each diffusion model to simulate until the diffusion process stops and record the average influence spread

	Cora-ML				Network Science				Power Grid				Jazz				Synthetic				Digg				Weibo			
Methods	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
Greedy	9.2	25.3	37.5	50.0	4.8	16.3	26.5	44.3	2.7	8.4	29.6	49.8	3.9	18.1	32.7	40.1	8.5	23.1	32.5	44.7	6.8	17.8	31.7	48.2	8.3	21.6	34.1	49.2
IMM	8.1	26.2	37.3	50.2	5.2	16.8	27.0	45.7	4.3	17.4	31.5	51.1	2.6	20.1	31.4	42.8	9.2	26.2	36.3	51.6	7.4	18.4	32.8	49.6	9.5	23.8	36.4	50.5
OPIM	13.4	26.9	37.4	50.9	6.6	19.4	28.9	48.6	5.7	17.7	29.7	50.1	2.4	20.1	34.4	46.8	9.6	25.3	36.6	51.7	7.6	18.5	32.9	48.9	9.7	23.7	36.6	50.3
SubSIM	10.1	25.7	36.8	51.1	4.8	15.4	27.9	44.8	4.6	19.2	31.7	50.2	3.6	18.8	37.6	44.7	9.5	26.7	36.5	51.5	7.5	18.9	33.3	49.4	9.3	23.1	36.5	50.6
OIM	8.9	27.6	38.0	51.3	4.2	16.7	26.5	48.2	5.7	17.5	31.9	50.8	2.0	18.5	36.3	42.2	9.6	26.2	36.7	51.3	7.8	18.2	33.1	49.6	-	-	-	-
IMINFECTOR	9.6	26.8	37.7	50.6	5.4	17.9	27.8	47.6	5.4	18.2	31.6	50.9	3.6	19.7	37.5	45.9	9.1	26.2	36.1	51.5	7.9	18.6	33.5	49.8	9.4	23.5	36.9	50.3
PIANO	9.8	25.2	37.4	51.1	4.7	16.3	27.1	47.2	5.3	18.1	31.7	50.2	2.2	19.2	36.6	43.2	9.1	26.4	36.2	51.6	-	-	-	-	-	-	-	-
ToupleGDD	10.6	27.5	38.5	51.5	6.3	17.8	28.3	50.5	5.4	19.3	31.6	51.3	3.3	20.4	37.2	45.7	9.5	26.8	37.1	51.4	-	-	-	-	-	-	-	-
DeepIM$_s$	13.6	27.7	38.5	51.8	6.9	19.1	29.3	50.5	5.9	20.2	31.7	51.5	3.8	21.4	38.9	47.1	10.2	26.8	37.5	51.8	7.9	18.8	33.7	50.3	10.1	24.7	36.8	50.8
DeepIM	**14.1**	**28.1**	**39.6**	**52.4**	**7.8**	**20.9**	**31.5**	**51.2**	**6.3**	**21.0**	**32.5**	**52.4**	**4.9**	**23.3**	**41.5**	**49.9**	**11.6**	**27.4**	**38.7**	**52.1**	**8.4**	**19.3**	**34.2**	**51.3**	**11.2**	**26.5**	**37.9**	**51.8**

Table 2.7: Performance over comparison methods under IC diffusion pattern. – indicates the model experiences an out-of-memory error during the execution of the dataset. (Best is highlighted with bold.)



(a) Cora_ML-IC (b) Net Science-IC (c) Power Grid-IC (d) Jazz-IC (e) Synthetic-IC

(f) Cora_ML-LT (g) Net Science-LT (h) Power Grid-LT (i) Jazz-LT (j) Synthetic-LT

Figure 2.12: The total infected nodes in the y-axis under the constraint of the budget with the node size growth (x-axis: 1%, 5%, 10%, and 20%). Figure 2.12a - 2.12e and Figure 2.12f - 2.12j are evaluated under the IC and LT model, respectively.

of 10 rounds. We report the percentage of final infected nodes (i.e., the number of infected nodes/the total number of nodes).

IM under IC Model. We evaluate the performance of DeepIM against baseline methods under the IC diffusion model. As shown in Table 2.7, DeepIM consistently outperforms other methods across all datasets. Traditional methods such as IMM, OPIM, and SubSIM, which leverage reserve-set sampling and approximation techniques, produce similar results but rely on different heuristics for efficiency, failing to capture the true distribution of seed sets. OIM performs better than traditional methods by iteratively updating edge weights but is specifically tailored for the IC model, limiting its real-world applicability. Learning-based methods (IMINFECTOR, PIANO, ToupleGDD) generally outperform traditional ones due to their superior gen-

	10,000	20,000	30,000	50,000	50,000 (Training)
IMINFECTOR	3.478s	7.842s	12.376s	16.492s	4753.67s
PIANO	5.948s	10.532s	16.575s	28.437s	14732.63s
ToupleGDD	10.476s	19.583	32.792s	58.985s	–
DeepIM$_s$	0.312s	0.616s	0.847s	1.275s	503.12s
DeepIM	1.402s	2.798s	5.124s	12.882s	1244.56s

Table 2.8: The average inference runtime (in seconds) with regard to the increase of node size and the average training time. We select 10% of nodes as the seeds.

eralization but face scalability issues, rendering them impractical for billion-scale networks like Digg and Weibo. In contrast, DeepIM offers a robust approach by learning the diffusion model end-to-end and searching for high-influence nodes in latent space, addressing both scalability and diffusion dynamics. Moreover, DeepIM$_s$ introduces a lightweight variant, retaining competitive performance with improved efficiency over other learning-based methods.

IM with Budget Constraint. Finally, we compare the quality of the seed sets generated by DeepIM and CELF under the IC and LT model with the budget constraint, and such a budget is explicitly defined as the node degree in this paper. As can be seen from Figure 2.12, our proposed method generally performs better than CELF across all networks of different sizes, and the margins are more evident under the LT model (Figure 2.12f - 2.12j). In addition, compared to CELF, the growths of influence spread in DeepIM have fewer fluctuations across all datasets, which also demonstrates the stability of DeepIM because of its capability of identifying the latent distribution seed sets while considering the budget constraint.

Scalability Analysis

We record the runtime of the seed set inference about the increase in node size against other learning-based IM solutions. As can be seen in Table 2.8, DeepIM demonstrates

near-linear growth of runtime as the graph size increases. In addition, it achieves a generally shorter inference time (on average 20% faster inference time than the second-fast IMINFECTOR) compared to other learning-based methods. In addition, our DeepIM$_s$ coupled with a lightweight end-to-end diffusion model can greatly reduce the computational cost of estimating the expected influence spread and achieves even 90% improvement in the inference time on average than our DeepIM model.

2.3.7 Conclusion

In this paper, we propose a novel framework to tackle the IM problem in a more robust and generalized way than existing learning-based IM methods. Particularly, to characterize the complex nature of the seed set, we propose to character the probability of the seed set and directly search for a more optimal seed set in continuous space. Furthermore, to solve the challenge of modeling the underlying diffusion pattern, we offer two different learning-based diffusion models to characterize the diversified diffusion dynamics with efficiency and efficacy guarantee. Finally, we propose a novel objective function that can be coupled with multiple constraints for seed node set inference, which can adapt to different IM application schemes. Extensive experiments and case studies on both synthetic and real-world datasets demonstrate the advantages of DeepIM over existing state-of-the-art methods to maximize the influence spread.

Chapter 3

Integrating Structured Knowledge and Quantifying Uncertainty in Natural Language Understanding

Natural language understanding is essential for the development of intelligent AI systems capable of meaningful human interaction. However, significant challenges persist in enabling these systems to effectively integrate structured knowledge and accurately quantify uncertainty in their predictions. This thesis addresses these challenges through two key studies.

Firstly, we explore the task of open-ended commonsense reasoning, which involves answering commonsense questions without predefined answer options. This task is particularly challenging due to the vast search space of potential answers and the necessity for implicit multi-hop reasoning across diverse pieces of knowledge. Traditional question-answering methods and retrieval-based approaches often struggle under these conditions. To overcome these limitations, we propose a novel method that leverages pre-trained language models to iteratively retrieve reasoning paths from external knowledge bases without requiring task-specific supervision. By constructing

these reasoning paths, our approach effectively narrows the search space and enhances reasoning capabilities, leading to significant improvements over state-of-the-art methods on commonsense reasoning benchmarks.

Secondly, we investigate the complexities of predictive uncertainty in the context of in-context learning—a feature of large language models (LLMs) that allows them to perform tasks based on a few examples provided in the input prompt. While in-context learning has expanded the applicability of LLMs, it raises concerns about the trustworthiness of their responses, especially due to the potential for generating hallucinations. Our research examines two primary sources of uncertainty in these models: aleatoric uncertainty, arising from variability in the input demonstrations, and epistemic uncertainty, stemming from the model's inherent limitations and configurations. We introduce a novel formulation and estimation method to quantify both types of uncertainty, providing an unsupervised, plug-and-play solution to better understand and mitigate risks associated with LLM predictions in in-context learning scenarios.

By integrating structured knowledge and quantifying uncertainty, this thesis advances the field of natural language understanding, enhancing both the reasoning capabilities and reliability of AI systems in handling complex language tasks.

3.1 Open-ended Commonsense Reasoning with Unrestricted Answer Scope

Current research on commonsense reasoning conventionally formulates the problem into a multiple-choice question answering (QA) format, where the best answer is expected to be chosen from a list of candidates for the given commonsense question. However, there are many practical and real-world scenarios where a small list of answer candidates or an answer scope (i.e., a relatively large set of concepts where the correct answer exists) are missing or not even provided (e.g., arbitrary questions

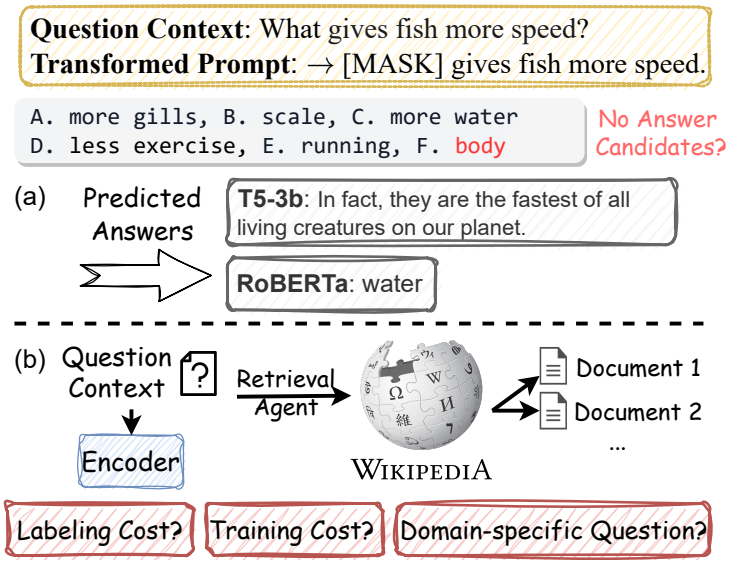


Figure 3.1: Current approaches can only partially solve the open-ended commonsense reasoning with unrestricted answer scope.

asked in the search engine), which requires the intelligent system to *understand* the commonsense question rather than picking a correct answer from a pre-defined pool. In this study, we focus on the **open-ended commonsense reasoning**, where we answer commonsense questions with two **constraints**: i.e., *without regulating an answer scope* and *without a pre-defined answer candidates list*. Open-ended commonsense reasoning is inherently challenging due to its core obstacle: the unrestricted answer scope would result in an extremely large search space, where the model cannot retrieve relevant answers effectively and efficiently.

Two approaches can be adapted to partially solve the open-ended commonsense reasoning (i.e., solving only one constraint). On the one hand, PLMs have been demonstrated to excel in various NLP tasks by using prompts. However, as shown in Figure 3.1 (a), both RoBERTa-large [120] and T5-3b [78] may not provide satisfying answers to the commonsense question since they can only leverage their own corpus to fill the mask in prompts. Without providing answer candidates, PLMs may have a limited capacity to obtain and accurately predict the answer, which requires structured reasoning. Even though lots of methods [99, 197, 209] have emerged to

incorporate external knowledge bases and PLMs to perform joint reasoning, their methods still require a small set of answer candidates, which is not applicable in the open-ended scenario.

Researchers have developed various knowledge-augmented retrieval methods to resolve the necessity of a few answer candidates in the open-ended QA problem, and their general inference scheme is visualized in Figure 3.1 (b). Specifically, instead of regulating a small set of answer candidates, knowledge-augmented retrieval methods [122, 15, 100] have designed an *answer scope* that directly contains the correct answer, and they leverage learning-based ranking algorithms to select the best answer. Although the form of the answer scope can vary, including a large set of conceptual entities and a set of question-related documents, building such an answer scope is still a resource-consuming and *ad-hoc* process. Moreover, well-trained retrievers are dependent on specific answer scopes, which are less applicable in real-world applications. For example, it's impossible to provide a relevant document set when answering commonsense questions during a conversation with a chatbot.

In this work, we present the external KnowlEdge-Enhanced Prompting method (KEEP) to achieve open-ended commonsense reasoning without pre-defining an answer candidate set and an answer scope. Firstly, to eliminate the requirement of answer candidates, KEEP leverages an external knowledge base (e.g., ConceptNet) as the answer searching space and iteratively extracts multi-hop reasoning paths relevant to the question. To avoid searching exhaustively over the whole knowledge base, we leverage PLMs to formulate the overall search criteria. The key insight is PLMs have certain reasoning abilities through their large-scale model parameters, which can be utilized to provide implicit knowledge in determining whether or not to keep expanding the reasoning paths or adopt the entity in the path as the final answer. Therefore, without restricting specific answer scopes and direct supervision of the reasoning process, KEEP can be applied in most real-world scenarios requiring

commonsense reasoning. To further enhance the reasoning ability of the PLM, we propose to leverage task-agnostic reasoning paths extracted directly from the external knowledge base as training instances to finetune the PLM.

We summarize our main contributions as follows.

- **We formulate the novel open-ended commonsense reasoning problem.** The open-ended commonsense reasoning is formulated as a multi-hop reasoning task iteratively conducted on an external knowledge graph.
- **We provide explanations along with the prediction results.** We leverage the implicit knowledge stored in PLMs to guide the overall searching/reasoning process on Knowledge Graphs, and the retrieved reasoning paths serve as additional explanations to justify the answer choice.
- **We empirically demonstrate the performance of our method against other methods**. Our proposed method excels other comparison methods in multiple metrics under the open-ended setting. Ablation studies and various case studies further prove the effectiveness of our method.

3.1.1 Related works

Neural Commonsense Reasoning. Combining PLMs and external knowledge for reasoning has recently gained lots of attention [26, 31]. State-of-the-art methods have been invented to inject commonsense knowledge into language models, either by pre-training on knowledge bases [122, 20], finetuning the model on the test domain [15], or leveraging structured knowledge base (e.g., ConceptNet) [197, 209, 33] so that they can infer with additional retrieved knowledge. However, none of these works except for LLMs [115] can be trivially adapted to solve the open-ended commonsense reasoning since they require substantial training instances for pre-training/finetuning or a list of pre-existing answer candidates designed for the question.

Open-ended Commonsense Reasoning. To date, a few attempts are trying to solve the open-ended commonsense reasoning. Gerber et al. [51] and Roemmele et al. [142] have first formulated the open-ended commonsense reasoning problem and leveraged the statistical natural language processing techniques. However, their performance is rather limited, and they may only provide a series of knowledge statements instead of providing a plausible answer. With the development of deep language models, a number of conversational and Mask Language Models (MLMs) (e.g., GPT-3 [18] and RoBERTa [120]) can also perform the task by leveraging prompt tuning and learning. However, even the powerful GPT-3 may not provide satisfying answers in the zero-shot setting by only relying on its own corpus. In addition, our work is also related to the work of *open-ended* commonsense reasoning [100], which formulated open-ended commonsense reasoning as a concept ranking process. Their approach still entails a training procedure on a given document set that is related to the commonsense question, which deviates from the main purpose of open-ended commonsense reasoning: lack of pre-defined answer candidates and finetuning data.

Explanation Generation for Commonsense Reasoning. Other than predicting the correct answer, it is also important to explore explicit reasoning steps behind the answer selection. Other than works that require direct supervision to predict explanation [136], Bosselut et al. [17] proposed to leverage knowledge graphs to acquire reasoning paths as the explanation in an unsupervised way. However, this approach requires pre-defined answers to guide the reasoning, which is not applicable in open-ended commonsense reasoning. The other line of works [149, 68, 119, 114] have also been utilizing model-generated text as the clarification of the commonsense question and empirically demonstrating the performance can be boosted by augmenting the query with knowledge statements. However, their models still require answer candidates as the model input. Additionally, purely relying on the language model

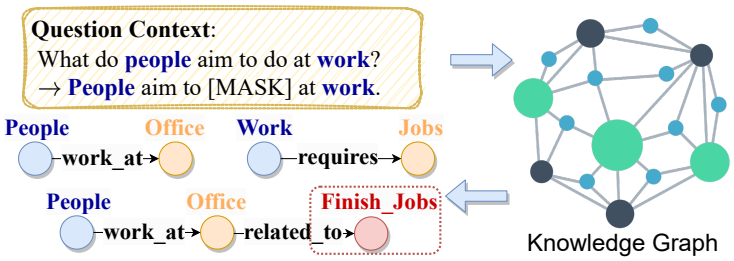


Figure 3.2: Example of the open-ended commonsense reasoning: the model takes the question as input and returns supporting reasoning paths (i.e., knowledge statements) with the best answer.

still lacks the model transparency, and the generated knowledge statement cannot be empirically served as the answer explanation [119].

3.1.2 Proposed Method

In this section, we first introduce the problem formulation, and then discuss the detailed framework of the proposed method, which can be divided into three components: 1) entity extraction and linking, 2) local knowledge graph expansion, and 3) training strategy and answer prediction.

Problem Formulation

We aim to solve open-ended commonsense reasoning questions by jointly using knowledge from a *PLM* and a *structured knowledge graph* G. The knowledge graph (KG) $G = (V, E)$ (e.g., ConceptNet[1]) is a multi-relational heterogeneous graph [108, 112]. V is the set of entity nodes, $E \subseteq V \times R \times V$ is the set of edges that connect nodes in V, where R represents a set of relation types (e.g., *locates_at* or *requires*). Specifically, given an open-ended commonsense reasoning question q **without** *providing answer candidates* and *regulating an answer scope*, the target of this work is to determine 1) a local KG $G_q \in G$ contains relevant information of q; 2) a set of reasoning paths $\mathbf{k} = \{k_1, k_2, ..., k_m\}$ extracted from G_q; and 3) an entity $\hat{a}$ extracted from $\mathbf{k}$ that is pre-

[1]https://conceptnet.io/

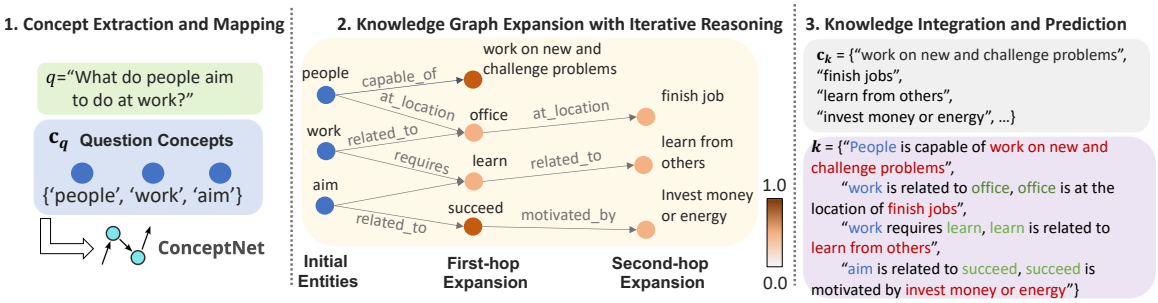


Figure 3.3: The framework of the proposed method consists of 1) concept extraction and entity linking; 2) local knowledge graph expansion with iterative reasoning steps, and 3) knowledge integration and final answer prediction.

cise to answer the question q. For example, in Figure 3.2, to answer a commonsense question *"What do people aim to do at work?"*, we aim at first extracting all relevant reasoning paths from the external KG that can provide us with logical information to answer the question. Among all the paths, we select the most precise one (i.e., *people* $\rightarrow$ *office* $\rightarrow$ *finish_jobs*) and extract the answer $\hat{a} =$ *finish_jobs* such that the following joint likelihood can be maximized.

$$P(\hat{a}, \mathbf{k}|q, G_q) = P(\mathbf{k}|q, G_q) \cdot P(\hat{a}|\mathbf{k}) \tag{3.1}$$

Challenges. However, maximizing the joint likelihood in Equation (3.1) is challenging due to two obstacles. First, retrieving the question-relevant reasoning paths $\mathbf{k}$ (i.e., knowledge statements) is difficult since we cannot build a local KG between question entities and answer candidates under the open-ended setting as [197, 209, 99, 121] do. Moreover, without regulating a pre-defined answer scope as Lin et al. [100] does, the search space would be the whole knowledge graph. Exhaustively expanding a multi-hop neighborhood that is relevant to the question on the knowledge graph would cause severe scalability issues.

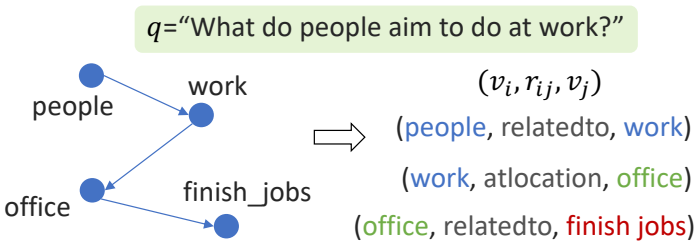
Next, to solve both challenges, we discuss how to initiate the local KG and iteratively reason over it to find all plausible knowledge statements and the most convincing answer. We demonstrate the overall framework in Figure 3.3.

Local Graph Construction and Expansion

Knowledge Graph Entity Linking. Conceptual knowledge graphs (e.g., ConceptNet) enable a variety of useful context-oriented reasoning tasks over real-world texts, which provides us with the most suitable structured knowledge in open-ended commonsense reasoning. To reason over a given commonsense context using knowledge from both PLM and G, the first step of the framework is to extract the set of critical entities $\mathbf{c}_q = \{c_q^{(1)}, ..., c_q^{(i)}, ...\}$ from the question q that have the surjective mapping to a node set $V_q \in V$ in the KG. Since q is often presented in the form of non-canonicalized text and contains fixed phrases, we follow the prior work [14] to map informative entities $\mathbf{c}_q$ from q to conjunct concept entities V_q in KG by leveraging the latent representation of the query context and relational information stored in G.

Reasoning Over Local Knowledge Graph. To imitate the human reasoning process, we aim to retrieve reasoning paths within L hops from G to form the local knowledge subgraph G_q that has the highest coverage to the question concepts $\mathbf{c}_q$. Ideally, each path in G_q can be regarded as a reasoning chain that helps to locate the most precise answer and its explanation to the question q. However, expanding L-hop subgraph G_q from $\mathbf{c}_q$ is computationally prohibited. Unlike other works [197, 99] that build G_q between the question q and all answer candidates, the open-ended commonsense reasoning problem does not provide any directions (i.e., answer candidates) or limited answer scope. The typical node size of a 3-hop local KG with $|\mathbf{c}_q| = 3$ could easily reach $1,000$ on ConceptNet, and many nodes are irrelevant under the current question context.

Reasoning Path Pruning. In order to make the process of reasoning path expansion scalable, we incorporate the implicit knowledge in PLMs to prune irrevelent paths. Specifically, we pair the question q with the text of node v along with



Relation Groups	Merged Relations	Relation Text
antonym/distinctfrom	antonym	is the antonym of
atlocation/locatednear	atlocation	is at location of
relatedto/similarto/synonym	relatedto	is related to

Figure 3.4: Knowledge statement transformation and cloze-based prompt construction.

the reasoning-path-transformed knowledge statement to form a cloze-based prompt $W = [q; v_j; (v_i, r_{ij}, v_j)]$ in order to turn the local graph expansion problem into an explicit reasoning procedure by directly answering the question with its derived reasoning path. For example, in Figure 3.4, the prompt is formatted as *What do people aim to do at work?* <***answer_node***>, *because* <***reasoning path***>. We leverage a pre-defined template to transform the triplet (v_i, r_{ij}, v_j) into natural language. For instance, the triplet (*work*, *antonym*, *unemployment*) can be translated to *work is the antonym of unemployment* as illustrated in Figure 3.4. Note that a KG typically contains many edge types that have similar meanings (e.g., both *antonym* and *distinct_from* have the same meaning *antonym*); therefore, we merge similar edge types into a unified template and illustrate a few examples of the templates in Figure 3.4. To evaluate whether we keep the reasoning path, we propose leveraging the PLM to score the relevance of each reasoning path given the context of the question. Formally, suppose the prompt W consists of N tokens $W = \{\omega_1, ..., \omega_{n-1}, \omega_n, \omega_{n+1}, ..., \omega_N\}$, the commonsense score $\phi_l(W)$ of the logical sentence W composed at l-th hop expansion is defined as:

$$\phi_l(W) := \sum_{n=1}^{N} \log \left(p_\theta(\omega_n | W_{\setminus n}) \right) / N, \tag{3.2}$$

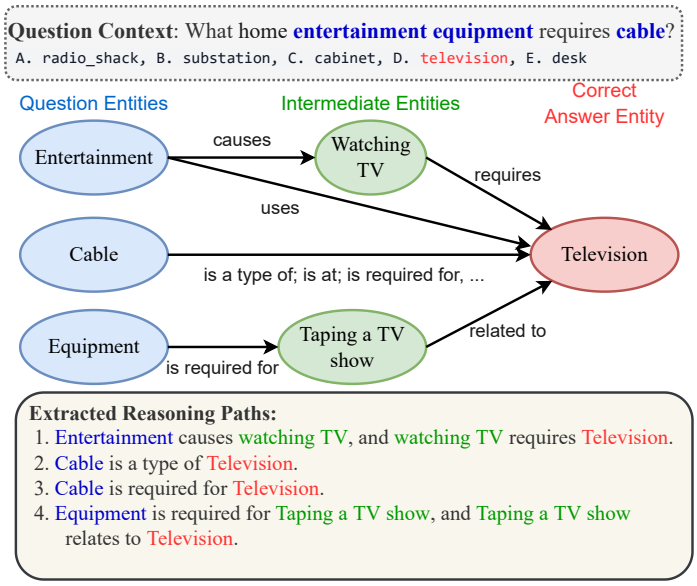


Figure 3.5: Training Corpus Generation. For each commonsense question in the training set, we discover all the reasoning paths between entities in the question and the correct answer entity in ConceptNet. All the reasoning paths are transformed into sentences by templates and thus serve as the finetuning corpus of our model.

where the $W_{\backslash n}$ indicates replacing the token ω_n to the [MASK], and the denominator N reduces the influence of the sentence length on the score prediction. Intuitively, $\log(p_\theta(\omega_n|W_{\backslash n}))$ can be interpreted as how probable a word ω_n given the context. For example, by filling *blue* and *red* into the masked logical statement $W_{\backslash n}$ = *"The sky is* [MASK]", *blue* should have a higher score than *red*.

As we iteratively expand G_q, each $\phi_l(W)$ scores a unique reasoning path at a particular $l \in [1, L]$ depth in the graph. As marked in Figure 3.3, a higher score $\phi_l(W)$ indicates the node v_j should be kept for the next $(l+1)$ hop expansion.

Training Strategy and Answer Prediction

Training Strategy. The proposed framework is able to answer open-ended commonsense questions with any off-the-shelf language models and the ConceptNet. However, we empirically find the performance of the off-the-shelf PLM is rather limited (i.e., commonsense score $\phi_l(\cdot)$ is less distinguishable between different prompts) when dealing with long-range reasoning paths (e.g., $L \geq 2$). In order to further enhance the

PLM's reasoning capability, we propose to finetune PLMs on the knowledge examples constructed from ConceptNet. Specifically, we aim to enhance the p_θ's reasoning capability by correctly identifying the knowledge triplets on ConceptNet. As depicted in Figure 3.5, given a commonsense question $q =$ *"What home entertainment equipment requires cable?"* and its correct answer $\tilde{a} =$ *"television"*, we identify reasoning paths $[(v_1, r_1, v_2), ..., (v_{L-1}, r_{L-1}, v_L)]$ on G from each entity $c_q^{(i)}$ in $\mathbf{c}_q$ to $\tilde{a}$. Note that there may exist multiple paths $c_q^{(i)}$ to $\tilde{a}$; e.g., *"Cable is a type of Television"* and *"Cable is required for Television"*. Each reasoning path is then transformed as natural language sentences with templates as illustrated in the table of Figure 3.4. We follow the standard masked language modeling task to finetune the model. By randomly masking a small portion (i.e., 15%) of tokens in each sentence, we aim to let the PLM comprehend the latent logic behind each retrieved path by learning to fill masks.

Answer Prediction. After we obtained the subgraph G_q consisting of all reasoning paths $\mathbf{k}$ within L-hop with a high commonsense score, each path $k_i \in \mathbf{k}$ can be regarded as an individual supporting knowledge explanation to an answer a_i.

$$\log p_\theta(a_i|k_i) \propto \phi_L = \sum\nolimits_{l=1}^{L} \phi_l,$$

where the ϕ_L denotes the final score for each answer a_i within L-hop and can be interpreted as approximating the likelihood of answer a_i given a singular reasoning path $\{c \to v_1 \to \cdots \to a\}$. To improve efficiency, we utilize beam search to only keep high-confidence reasoning paths. We can thus pick the answer $\hat{a}$ and its reasoning path $\hat{k}$ with the highest score ϕ_L as the final answer and supporting knowledge.

3.1.3 Experiment

We leverage RoBERTa-large [120] as our base PLM. We empirically verify the performance of the proposed method against other methods on commonsense reasoning

benchmark datasets under the open-ended setting. Due to the space limit, more experiments, case studies, and implementation details can be found in Appendix B.1.

Experiment Setting

Dataset. We evaluate our method on two commonsense reasoning benchmarks. *1) CommonsenseQA (CSQA)*: Talmor et al. [156] that contains 1,140 test cases. *2) QASC*: Khot et al. [83] that contains 917 test cases. We only keep the question and discard the attached multiple-choice answers.

Comparison Methods. Since open-ended commonsense reasoning with unrestricted answer scope is a novel setting, we have no direct opponents to compete. All the QA models either require pre-defined answer candidates or a specific answer scope, which *CANNOT* be applied in the real open-ended scenario. In this work, we compare our model against the following baselines. 1) *RoBERTa-large* [120] is an MLM that is trained with dynamic masking, a larger batch size, and a larger vocabulary size. 2) *DeBERTa-v3-large* [62] improves the BERT and RoBERTa models using disentangled attention and enhanced mask decoder. 3) *RelBERT* [163] is a finetuned model based on RoBERTa, which particularly focuses on improving the relation embedding and leverages relational triplets extracted from ConceptNet as training corpus. 4) *T5-3b* [78] is an encoder-decoder-based language model pre-trained on a multi-task mixture of unsupervised and supervised tasks. We use its 3b version that contains 3 billion parameters. 5) *UnifiedQA* [82] is a unified pre-trained language model specifically for generative question-answering tasks, which is based on T5-large. 6) *GPT-3* [18] is one of the largest language models with 175 billion parameters, which is powerful and excels other methods in multiple NLP tasks. We use all language models in the zero-shot setting to follow the open-ended application scenario.

Method		CSQA			QASC		
		Top-1	Top-3	Top-5	Top-1	Top-3	Top-5
Masked Language Model	DeBERTa-v3-large	0.273	0.426	0.607	0.254	0.554	0.618
	RoBERTa-large	0.275	0.477	0.682	0.294	0.523	0.578
	RelBERT	0.302	0.567	0.698	0.362	0.574	0.601
Generative Language Model	T5-3b	0.426	0.471	0.501	0.422	0.546	0.572
	UnifiedQA	0.395	0.439	0.517	0.379	0.513	0.602
	GPT-3	0.476	0.654	0.769	0.452	0.573	0.749
Ours	KEEP (w/o finetuning)	0.385	0.615	0.776	0.467	**0.742**	0.821
	KEEP	**0.523**	**0.714**	**0.798**	**0.489**	0.732	**0.829**

Table 3.1: Top-1, 3, and 5 prediction accuracy made by human annotators for each model. (The higher the better)

Evaluation Criteria. Since we do not have ground truth to evaluate the prediction correctness, we generate answer candidates for each commonsense question and work with human annotators to indicate whether there exists a precise answer that could answer the given question. For baselines with MLMs: RoBERTa, DeBERTa, and RelBERT, we design prompts that allow each model to fill the mask with top-N answer choices. For baselines with generative language models, T5-3b, UnifiedQA, and GPT-3, they are prompted to generate direct answers (within 20 tokens). In addition to human judgment, we also incorporate the commonsense score (Equation (3.2)) to evaluate the perplexity of the answer choice. Specifically, we choose the best answer from all the candidates generated by each model and concatenate the answer to the original question. Following the way of evaluating the commonsense score of the sentence in Zhou et al. [214], We use GPT2-large [139] and LLaMA2-70B [162] as the base model to calculate the score since GPT2-large is not included in our comparison methods. Intuitively, a PLM should assign higher probabilities to answers that are semantically and syntactically correct to the question.

Results

Qualitative Analysis. Table 3.1 summarizes the Top-N accuracy results. For each approach, the test results are obtained by evaluating if there is a precise answer

Method	GPT-2		LLaMA-2-70B	
	CSQA	QASC	CSQA	QASC
DeBERTa-large	12.498	15.528	94.391	62.497
RoBERTa-large	8.589	10.788	91.467	59.582
RelBERT	9.327	8.543	77.813	54.284
T5-3b	9.152	9.314	76.789	56.759
UnifiedQA	8.573	8.439	58.929	45.621
GPT-3	6.527	**7.528**	43.325	33.569
KEEP	**6.139**	7.692	47.472	37.793
Ground Truth	4.844	5.327	34.675	29.579

Table 3.2: The commonsense score of each model, which is calculated by GPT-2 and LLaMA-2 through concatenating each question and the most suitable answer generated from each model. (The lower the better)

in the Top-1, 3, and 5 generated answers. As shown in the table, our proposed method excels both MLMs and generative language models by an evident margin (achieved approximately 9%, 20%, and 15% improvement than the second-best on Top-1, 3, and 5 accuracy on both datasets, respectively). Additionally, We report several observations from the table to explain the results: *1) PLMs do not generalize well on unseen entities.* Without relying on pre-defined answer candidates, PLMs do not make satisfied predictions of reasoning-related prompts. Especially for MLMs like DeBERTa and RoBERTa, most of the correct answers in the reasoning questions are not even encountered during pre-training due to their heavy reliance on memorization in the pre-training process. *2) PLMs are becoming a promising alternative to external knowledge bases.* As we can see from the table, generative PLMs (i.e., T5, UnifiedQA, and GPT-3) generally perform well on the Top-1 accuracy on both datasets, which indicates signs of capturing relational knowledge in a zero-shot setting reasonably well compared to our proposed method. However, their performances do not show evident improvements if we can choose from Top-3 and 5 candidates. Even though MLMs can achieve improvements at each level, their performance still cannot be compared to ours since standard PLMs lack knowledge awareness without accessing external knowledge. *3) Commonsense reasoning ability is not fully determined by*

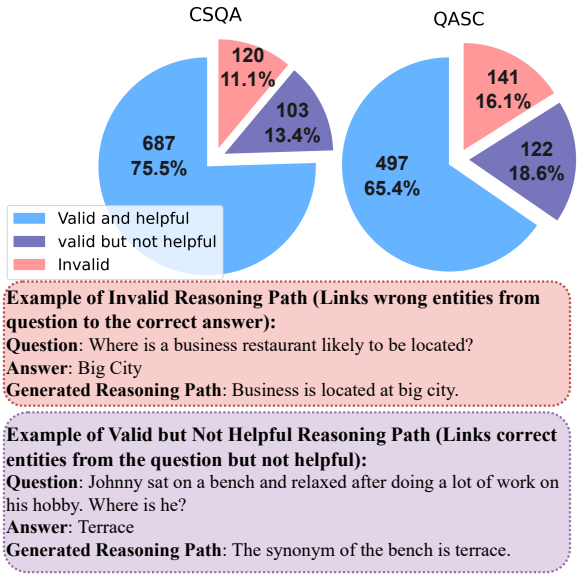


Figure 3.6: The Percentage of Valid Reasoning Paths in the Correct Prediction. There are 910 correct predictions in the CSQA dataset, and 809 of them are valid. In QASC dataset, there are 638 out of 760 reasoning paths that are valid based on human judgment.

Setting	Top-1 Score	
	CSQA	QASC
Ours (Reasoning Path Length $L = 3$)	0.523	0.489
w/o finetuning	0.385	0.467
w/o explanation	0.449	0.423
Reasoning Path Length $L = 1$	0.379	0.422
Reasoning Path Length $L = 2$	0.515	0.476

Table 3.3: Ablation Study.

model sizes. Without proper finetuning on target datasets, larger language models may find it harder to mine latent and unstructured knowledge, which indicates their performance may deteriorate. For instance, RoBERTa-large and RelBERT contain 300 million of parameters while T5-3b contains more than 3 billion of parameters. However, both RoBERTa-large and RelBERT have competitive performance with T5-3b on both datasets.

Quantitative Analysis. Table 3.2 depicts the average commonsense score of the predicted answer from each comparison method. Specifically, our human annotators

pick the most suitable answer from each model's predicted answer candidates, and we concatenate the answer with the question to form a sentence. We use the vanilla GPT-2-large to calculate the commonsense score of each sentence by Eq. (3.2). As shown in Table 3.2, our method achieves competitive performance with GPT-3 across two datasets. First, lacking support from external knowledge bases and pre-defined answer candidates, MLMs may not perform well in generating answers only relying on prompts. Generative models tend to generate long and coherent sentences as the answer rather than short words/phrases. Even though they may not fit the commonsense, they can still achieve better commonsense scores than MLMs.

Validity of Reasoning Paths. In addition, we also incorporate human evaluation to check the validity of the generated reasoning paths for our methods, and we report the percentage of valid reasoning paths in Figure 3.6. Note that the invalid reasoning path is defined as falsely linking the correct entity in the question to the semantically correct answer, and the valid but not helpful reasoning paths denote our model links the correct entity from the question to a semantically correct answer on ConceptNet, but the reasoning path may not aid the language model make predictions. We also give examples of both cases in the figure. Statistics show that the majority (nearly 90% on the CSQA and 80% on the QASC) of the generated reasoning paths are grammatical and valid to the question. On top of that, around 70% of them can be helpful and relevant to the context of the question.

Ablation Study. Next, we conduct an ablation study to investigate the importance of each component in the model, and the results are reported in Table 3.3. Firstly, the performance of our method drops around 25% without finetuning, which indicates the logical sentences transformed from reasoning paths can indeed help the model navigate to the most correct answers that fit the commonsense in ConceptNet. Secondly, we discard concatenating the reasoning-path-transformed explanation to the prompt,

CSQA Test Case	Generation Results
	Prompt: What do people aim to do at work? →(People aim to [MASK] at work.)
DeBERTa-v3-large	burst
RoBERTa-large	succeed
RelBERT	improve
T5-3b	What tools are they going to use? What products is their life like?
UnifiedQA	The answer is not to just go. "Work" is only part of the solution to unemployment.
GPT-3	People aim to do many different things, depending on their individual goals and aspirations.
KEEP (Ours)	Work on new and challenging problems. **Reasoning Chain**: **Work** is done by **People**, **People** desires to **work on new and challenging problems**.
QASC Test Case	Generation Results
	Prompt: What is saturated fat at room temperate? →(The saturated fat at room temperate is [MASK].)
DeBERTa-v3-large	unchanged
RoBERTa-large	negligible
RelBERT	zero
T5-3b	It is a major source of energy in the human body.
UnifiedQA	You will find fats like *butter* or *margarine*, the main components of the food chain.
GPT-3	Saturated fat is a type of fat that is solid at room temperature.
KEEP (Ours)	Solid Object. **Reasoning Chain**: **Fat** is related to **Butter**, **Butter** is a type of **solid object**.

Table 3.4: Test cases of all comparison methods on both datasets. Under the open-ended setting, KEEP excels in other methods and achieves competitive performance with GPT-3 in generating answers and valid reasoning paths.

and the performance also drops approximately 15% in both datasets. As with other explanation-aided commonsense reasoning models [149, 119], the logical sentence can indeed help the model make better predictions. Finally, we also investigate how the length of reasoning paths impacts the model performance. Specifically, the length of reasoning paths denotes the maximal hop of neighbors our method could explore from the question entities. Intuitively, if we regulate the length of reasoning paths to be short, it may not reach answers that require multi-hop reasoning. However, if we set a large path length, the model may generate noisy paths and the search time would be unacceptable. As shown in the table, there is a large performance gap if we set the reasoning path length to be 1, which indicates most of the answers do not exist within

the first hop of question entities. Ae increase the length, the performance difference between $L = 2$ and 3 (our adopted length) is very small. Therefore, considering both effectiveness and efficiency, we adopt $L = 3$ as the maximal path length.

Case Study. Finally, we demonstrate a few examples from both datasets to see how the retrieved reasoning path can help the PLM to make the correct prediction under the open-ended setting. As shown in Table 3.4, masked Language models, i.e., RoBERTa, DeBERTa, and RelBERT, generally can only predict a single token to fill the mask. Even though they can make feasible predictions in some cases, they cannot provide valid reasoning chains. Generative language models predict the answer in an autoregressive way, which could generate a full sentence to answer the question. However, without proper training on the test domain, even the strongest GPT-3 cannot provide a precise answer for questions like *What do people aim to do at work?*. As opposed to existing approaches, by reasoning over the external KG, KEEP can generate precise answers and provide a reasoning chain to support the answer choice without any learning steps during the inference.

3.1.4 Conclusion

We present an off-the-shelf framework KEEP to predict answers for open-ended commonsense reasoning without requiring answer candidates and a pre-defined answer scope. By integrating the implicit knowledge stored in PLMs and the external knowledge base, KEEP retrieves relevant reasoning paths and extracts suitable answers for commonsense questions while maintaining both efficiency and efficacy. We believe this work poses a new direction to automated commonsense reasoning under the zero-shot and open-ended setting in the Large Language Model era [115].

3.2 Uncertainty Quantification for In-Context Learning of Large Language Models

Large Language Models (LLMs) have revolutionized diverse domains by serving as general task solvers, which can be largely attributed to the emerging capability: *in-context learning*. By providing demonstrations of the task to LLMs as part of the prompt, LLMs can quickly grasp the intention and make corresponding responses to the particular task [129]. In this paradigm, LLMs can quickly adapt to solve new tasks at inference time (without any changes to their weights). Advanced LLMs, e.g., GPT-4 and LLaMA, have achieved state-of-the-art results on LAMBADA (commonsense sentence completion), TriviaQA (question answering) [181], and many tasks in other domains [115, 113].

While in-context learning has achieved notable success, LLMs remain vulnerable to well-known reliability issues like hallucination [140, 11]. Uncertainty quantification has emerged as a popular strategy to assess the reliability of LLM responses. In past years, numerous works [180, 104, 116, 4, 87] have been proposed to quantify the uncertainty of LLM response. These approaches either return a confidence score or directly compute variance/entropy across multiple responses; however, they often overlook the complex nature of LLMs and their reliance on provided demonstrations in in-context learning, so that existing methods may not provide insights into the underlying causes or the interactions among different factors contributing to uncertainty.

A natural question therefore arises: when LLM uses in-context learning to predict a wrong answer with high uncertainty, can we indicate if it is caused by the demonstration examples or by the model itself? Given LLM's responses to a particular task, it's essential to decompose the uncertainty into its primary sources to address the question. Specifically, *Aleatoric Uncertainty (AU)* refers to variations

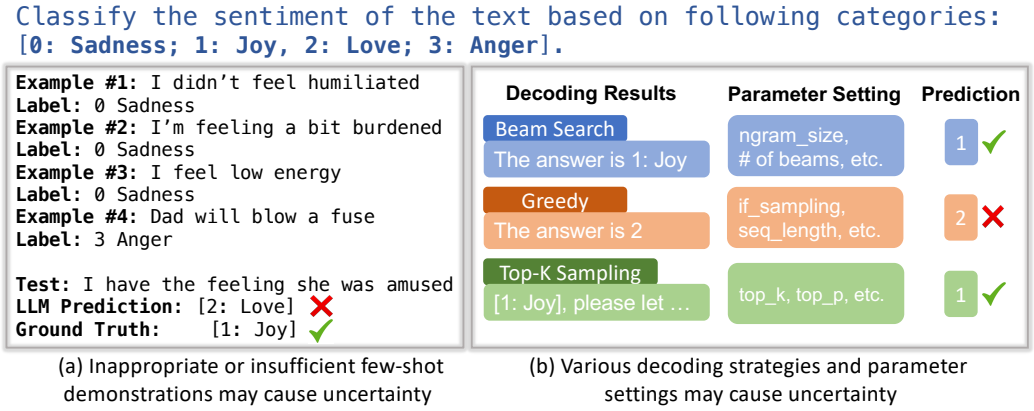


Figure 3.7: Uncertainty in LLM's prediction can stem from two aspects: a) *Demonstration Quality*: LLMs are likely to make wrong predictions if the demonstrations are inappropriate; b) *Model Configuration*: different decoding strategies (e.g., beam_search and top_k sampling) and their parameter settings may return different predictions.

in the data, often linked to the demonstration examples. As shown in Figure 3.7 (a), LLM's output can easily be disturbed by inappropriate demonstrations since the provided demonstrations do not cover all possible labels. The noise and potential ambiguity of these demonstrations could bring uncertainty, which, in turn, may hinder the accuracy of the response. In contrast, *Epistemic Uncertainty (EU)* stems from ambiguities related to the model parameters or different configurations. As depicted in Figure 3.7 (b), different decoding strategies (e.g., beam search and greedy decoding) and their hyperparameter settings can have different decoding results. Recognizing and quantifying the uncertainty from the model's perspective can also be critical in evaluating the generated responses, which allows us to understand the model's confidence level toward the task and make necessary adjustments (e.g., choosing a more powerful model or conducting an ensemble prediction).

Despite the strides made by existing works in understanding the total uncertainty, the decomposition of uncertainty in the realm of in-context learning remains underexplored. In this work, we propose a novel framework for quantifying the uncertainty of in-context learning to aleatoric and epistemic components from the generated outputs. Our contributions are summarized as follows.

- **Problem.** We formulate the problem of uncertainty decomposition that extracts epistemic and aleatoric uncertainties from the predictive distribution of LLMs with in-context learning.

- **Method.** We propose quantifying both aleatoric and epistemic uncertainty from the mutual information perspective. A novel entropy-based estimation method is also designed to handle the free-form outputs of LLMs.

- **Experiment.** Extensive experiments are conducted to evaluate different aspects of uncertainty, followed by specific applications and case studies to show how two types of uncertainty influence the model's performance.

3.2.1 Uncertainty Decomposition of In-context Learning

We first formulate the process of in-context learning as Bayesian Neural Networks with latent variables. Based on the formulation, we propose to decompose the predictive uncertainty into its epistemic and aleatoric components from the mutual information perspective, followed by a novel way to estimate both uncertainties based on the entropy of the prediction's distribution.

Background

LLMs are typically trained using maximum likelihood estimation on a large corpus of text. The training goal is to maximize the likelihood of the observed data under the model: $\mathcal{L}(\Theta) = \prod_{i \leq N} p(\omega_i | \omega_1, \omega_2, \ldots, \omega_{i-1}; \Theta)$, where each $\omega_i \in \mathbf{x}$ is a token in a sentence $\mathbf{x} = [\omega_1, \ldots, \omega_N]$, and Θ denotes the set of parameters.

Latent Concept. From the Bayesian point of view, LLM's in-context learning ability is obtained by mapping the training token sequence $\mathbf{x}$ to a latent *concept* z [181]. The concept z is a latent variable sampled from a space of concepts $\mathcal{Z}$, which

defines a distribution over observed tokens ω_i from a training context $\mathbf{x}$:

$$p(\omega_1, \ldots, \omega_N) = \int_{z \in \mathcal{Z}} p(\omega_1, \ldots, \omega_N | z) p(z) dz.$$

The concept can be interpreted as various document-level statistics, such as the general subject matter of the text, the structure/complexity of the text, the overall emotional tone of the text, etc.

In-context Learning. Given a list of independent and identically distributed (i.i.d.) in-context demonstrations (contain both question and answer) $[\mathbf{x}_1, \ldots, \mathbf{x}_{T-1}]$ concatenated with a test question (without the task answer) $\mathbf{x}_T$ as prompt. Each demonstration $\mathbf{x}_i$ in the prompt is drawn as a sequence conditioned on the same concept z and describes the task to be learned. LLMs generate a response $\mathbf{y}_T$ to the test question $\mathbf{x}_T$ based on the aggregated prompt $\mathbf{x}_{1:T}$:

$$p(\mathbf{y}_T | \mathbf{x}_{1:T}) = \int_{z \in \mathcal{Z}} p(\mathbf{y}_T | \mathbf{x}_{1:T}, z) p(z | \mathbf{x}_{1:T}) dz.$$

In-context learning can be interpreted as *locating* a pre-existing concept z based on the provided demonstrations $\mathbf{x}_{1:T-1}$, which is then employed to tackle a new task $\mathbf{x}_T$. Including more high-quality demonstrations within the prompt can help refine the focus on the relevant concept, enabling its selection through the marginalization term $p(z|\mathbf{x}_{1:T})$. Note that formulating in-context learning as Bayesian inference with latent variables is more of a hypothesis; however, demystifying the in-context learning from the view of Bayesian inference offers a probabilistic interpretation of how LLM learns and adapts to new data in context.

In this work, we focus on quantifying the predictive uncertainty of LLMs in deterministic NLP tasks, such as text classification. Specifically, we address tasks where the training dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ consists of token sequences $\mathcal{X} = \{\mathbf{x}\}$ and their

corresponding target outputs $\mathcal{Y} = \{\mathbf{y}\}$. For LLMs, the generation process is defined by the function $\mathbf{y} = f(\mathbf{x}, z; \Theta)$, where $f : \mathcal{X} \times \mathcal{Z} \to \mathcal{Y}$ is a deterministic function. The output $\mathbf{y}$ exhibits stochastic behavior, influenced by the latent concept $z \sim p(z|\mathbf{x}_{1:T})$ and the model parameters/configurations Θ (e.g., temperature, etc.).

Predictive Uncertainty Formulation of In-context Learning

We formulate the predictive distribution of in-context learning for predicting $\mathbf{y}_T$ given few-shot demonstrations $\mathbf{x}_{1:T-1}$ and a test case $\mathbf{x}_T$ as:

$$p(\mathbf{y}_T|\mathbf{x}_{1:T}) \approx \int p(\mathbf{y}_T|\Theta, \mathbf{x}_{1:T}, z) \cdot p(z|\mathbf{x}_{1:T}) q(\Theta) dz \, d\Theta, \tag{3.3}$$

where $p(\mathbf{y}_T|\Theta, \mathbf{x}_{1:T}, z)$ is approximated by a Bayesian Neural Network-based likelihood function $\mathcal{N}(f(\mathbf{x}_{1:T}, z), \mathbf{\Sigma})$, and $\mathbf{\Sigma}$ is the covariance matrix contains the variances and covariances associated with LLM parameters. $q(\Theta)$ is the approximated posterior of the LLM's parameters Θ. Eq. (3.3) approximates LLM outputs following a Gaussian distribution, which serves as an initial framework for generating predictions based on input data and accompanying demonstrations: $p(\mathbf{y}_T|\mathbf{x}_{1:T})$, which entangles different types of uncertainties. We first present the overall pipeline of our uncertainty quantification framework, followed by formulation on decomposing the total uncertainty based on mutual information and a novel way to estimate the uncertainty. Note that LLMs can be categorized into white-box and black-box models [115] based on their transparency. Quantifying mutual information involves accessing the probability of generated tokens, which is not applicable to black-box LLMs. In this study, we also provide a decomposition way from the variance perspective for black-box LLMs. Due to the space limit, the variance-based decomposition can be found in Appendix B.2.1.

Framework Pipeline. In this work, we employ a Bayesian framework to quantify the predictive uncertainty from LLMs, and the overall pipeline is visualized in Figure

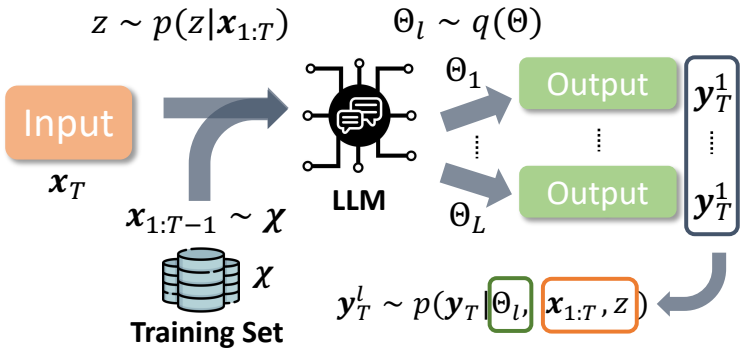


Figure 3.8: Uncertainty Quantification of In-context Learning Pipeline: we want to quantify the uncertainty that comes from 1) different in-context demonstrations $\mathbf{x}_{1:T}$; and 2) different model configurations Θ_l.

3.8. Specifically, the input $\mathbf{x}_{1:T}$ is composed of a test query $\mathbf{x}_T$ and a set of demonstrations $\mathbf{x}_{1:T-1}$ sampled from $\mathcal{X}$. By sampling different model parameters/configurations $\Theta_l \sim q(\Theta)$, LLM can return different outputs $\mathbf{y}_T^l \in [\mathbf{y}_T^1, \cdots, \mathbf{y}_T^L]$ based on the conditional probability $p(\mathbf{y}_T|\Theta_l, \mathbf{x}_{1:T}, z)$. The collection of outputs $[\mathbf{y}_T^1, \cdots, \mathbf{y}_T^L]$ records the total uncertainty regarding Θ_l and demonstrations $\mathbf{x}_{1:T-1}$.

Entropy-based Decomposition

As a widely adopted measure of uncertainty, entropy provides a quantifiable and interpretable metric to assess the degree of confidence in the model's predictions [123]. Since white-box LLMs can return the probability of each token in the generated sequence, it naturally makes entropy-based uncertainty measures applicable uniformly across different types of white-box LLMs.

Epistemic Uncertainty (EU). Let $H(\cdot)$ be the differential entropy of a probability distribution, the total uncertainty in Eq. (3.3) can be quantified as $H(\mathbf{y}_T|\mathbf{x}_{1:T})$, which entangles both aleatoric (i.e., demonstration $\mathbf{x}_{1:T-1}$) and epistemic (i.e., model parameter Θ) uncertainties. To estimate the EU, we condition Eq. (3.3) on a specific realization of the model parameter Θ, yielding $p(\mathbf{y}_T|\mathbf{x}_{1:T}, \Theta) = \int p(\mathbf{y}_T|\mathbf{x}_{1:T}, z, \Theta)p(z|\mathbf{x}_{1:T})dz$ with an associated entropy $H(\mathbf{y}_T|\mathbf{x}_{1:T}, z, \Theta)$. The expected value of this entropy under different demonstration sets can be expressed as $\mathbb{E}_z[H(\mathbf{y}_T|\mathbf{x}_{1:T}, z, \Theta)]$, which serves

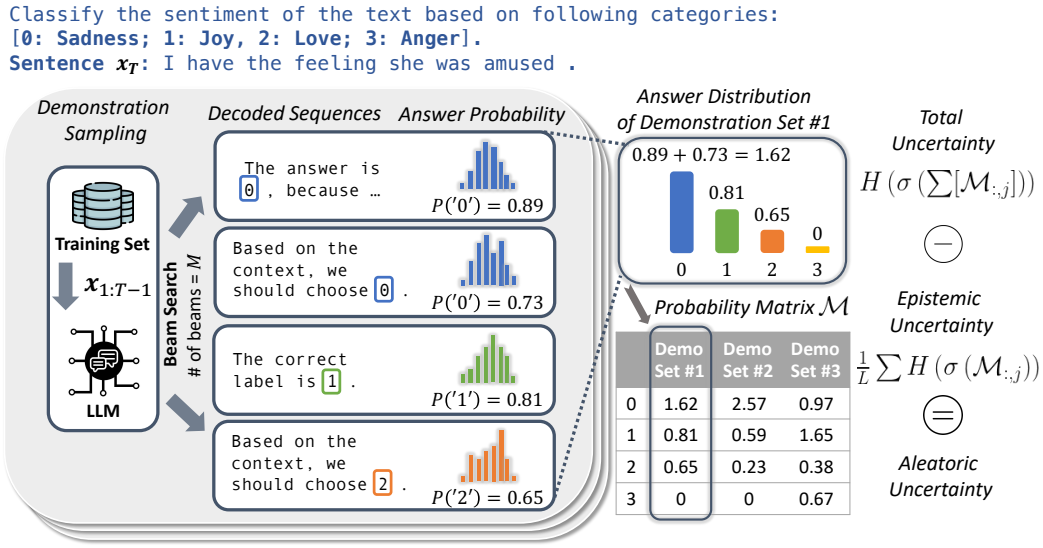


Figure 3.9: Framework of entropy-based uncertainty estimation, which consists of 1) generating M sequences based on a set of $\mathbf{x}_{1:T-1}$; 2) selecting token(s) that is relevant to the answer and extract the probabilities; 3) aggregating the token probabilities of M sequences into a distribution of predicted labels; 4) iterating the process L times corresponding to L different demonstration sets and form a probability matrix $\mathcal{M}$, where the column denotes different demonstration sets and the row denotes labels of the dataset.

as a metric to quantify the EU in Eq. (3.3).

Aleatoric Uncertainty (AU). In terms of AU, the randomness comes from different sets of demonstration $\mathbf{x}_{1:T-1}$ and their corresponding latent concept z. To estimate AU, we can quantify the mutual information between $\mathbf{y}_T$ and latent concept z, which can often be leveraged as an evaluation metric of AU [176]. As we have already obtained the EU, AU can subsequently be calculated as the discrepancy between the total uncertainty and the epistemic uncertainty:

$$I(\mathbf{y}_T, z|\Theta) = H\left(\mathbf{y}_T|\mathbf{x}_{1:T}, \Theta\right) - \mathbb{E}_z\left[H(\mathbf{y}_T|\mathbf{x}_{1:T}, z, \Theta)\right]. \tag{3.4}$$

The entropy $H\left(\mathbf{y}_T|\mathbf{x}_{1:T}, \Theta\right)$ can be approximately calculated as $-\sum_t\left[p(\omega_t^{\mathbf{y}_T}) \cdot \log p\left(\omega_t^{\mathbf{y}_T}\right)\right]$, where $p(\omega_t^{\mathbf{y}_T})$ represents the probability of each possible next token $\omega_t^{\mathbf{y}_T}$ given the input prompt $\mathbf{x}_{1:T}$. Therefore, the AU in Eq. (3.4) can be approximated by sampling

many z (by sampling different sets of demonstrations) to obtain different $\mathbf{y}_T$ conditioning on one set of parameters Θ. The group of $\mathbf{y}_T$ can then be used to approximate the respective entropies for each group of demonstrations $\mathbf{x}_{1:T-1}$:

$$\begin{aligned} I(\mathbf{y}_T, z|\Theta) &= H\left(\mathbf{y}_T|\mathbf{x}_{1:T}, \Theta\right) - \mathbb{E}_z\left[H(\mathbf{y}_T|\mathbf{x}_{1:T}, z, \Theta)\right] \\ &\approx \sum^{M\times L} H(\mathbf{y}_T) - \frac{1}{M}\sum_{m=1}^{M}\sum_{l=1}^{L}\left[H(\mathbf{y}_T^{\Theta_m,l})\right], \end{aligned} \tag{3.5}$$

where $[\mathbf{y}_T^{\Theta_m,l}]$ are obtained corresponding to different demonstrations $[\mathbf{x}_{1:T-1}^1, \ldots, \mathbf{x}_{1:T-1}^L]$, and $[\Theta_1, \ldots, \Theta_M]$ are sampled from $q(\Theta)$. However, in many cases, direct sampling from the posterior is hard since it requires a prohibitive number of samples to approximate it effectively. Beam search is then used as an efficient alternative to find high-quality hypotheses. This approach can be viewed as a form of importance sampling, where hypotheses are drawn from high-probability regions of the space. Each hypothesis $\mathbf{y}_T$ observed during the beam search process is associated with uncertainty, which is importance-weighted in proportion to $p(\mathbf{y}_T|\mathbf{x}_{1:T}, z)$. Beam Search thus serves as a practical and efficient way to sample from the posterior by focusing on the most relevant parts of the hypothesis space.

In addition, since calculating the entropy $H\left(\mathbf{y}_T\right)$ entails to obtain the joint probability of the generated tokens $p(\mathbf{y}_T) = (\omega_1^{\mathbf{y}_T}, \ldots, \omega_T^{\mathbf{y}_T})$, entropy-based method may only be applicable to white-box LLMs.

Entropy Approximation

The generation of LLMs is generally free-form, which makes the uncertainty estimation for in-context learning still different from well-studied classification models that have specific labels. Specifically, not only may the LLM not always be able to return an expected answer, but the generated sequence may also consist of placeholder tokens. Calculating the entropy of the whole generated sequence would involve re-

dundant tokens. Therefore, in this work, we propose to approximate the entropy of the output $H(\mathbf{y}_T)$, and the process is summarized in Figure 3.9.

Given the probability distributions of the generated tokens $p(\mathbf{y}_T)$ for one set of demonstrations, we only select token(s) $\omega_t^{\mathbf{y}_T}$ that directly answer the provided question. Taking the text classification task as an example, LLM is asked to directly output a numerical value standing for a predefined category (e.g., 0: Sadness, 1: Joy, etc.). The probability of the token $\omega_t^{\mathbf{y}_T}$ that represents the numerical value is then leveraged to denote the overall distribution of $p(\mathbf{y}_T)$. We aggregate the answer probabilities from all M decoded sequences and transform them as an answer distribution (as shown in the top right corner in Figure 3.9). After repeating the process L times, where L corresponds to L different sets of demonstrations, we have a matrix $\mathcal{M}$ recording the answer distributions of choosing different demonstrations and model configurations (as shown in the lower right corner in Figure 3.9). The EU and AU can then be approximated as:

$$EU = \frac{1}{L}\sum H\left(\sigma(\mathcal{M}_{:,j})\right), \quad AU = H\left(\sigma\left(\sum[\mathcal{M}_{:,j}]\right)\right) - \frac{1}{L}\sum H\left(\sigma\left(\mathcal{M}_{:,j}\right)\right),$$

where $\sigma(\cdot)$ normalizes the column $\mathcal{M}_{:,j}$ into a probability distribution, and entropy $H(\cdot)$ can be calculated as $-\sum_{k=1}^{K}\left(p(\mathcal{M}_{k,j}) * \log(p(\mathcal{M}_{k,j}))\right)$ if the number of labels is K. Note that we have instructed LLMs not to generate tokens with less semantic meaning, such as dashes, spaces, or non-related words. In practice, our adopted LLMs can follow the instructions only to return desired answers so that the whole sentence will be the answer tokens (no need to select tokens).

3.2.2 Related Works

Uncertainty Quantification and Decomposition. Uncertainty quantification aims to measure the confidence of models' predictions, which has drawn attention

from various domains [212, 124]. Measuring uncertainty is essential in many real-world NLP applications where making a wrong prediction with high confidence can be disastrous (e.g., assessing the confidence in a translation or a generated piece of information). This is especially important in foundation models since we do not have enough resources to finetune the model [1]. To better understand the uncertainty, the primary focus is on understanding and categorizing the sources of uncertainty for interpreting the models' outputs more effectively. The output uncertainty can typically be categorized into *Aleatoric Uncertainty* that arises from the inherent noise in the data, and *Epistemic Uncertainty* that arises due to inappropriate model architecture or overfitted/underfitted parameters. Existing methods [29, 36, 123] have come up with various methods (e.g., Bayesian neural network, Deep Ensembles, and Monte Carlo Dropout) to decompose the uncertainty.

Uncertainty in Language Models. Existing LLMs often neglect the importance of uncertainty in their responses. Earlier works [179, 37, 74] on uncertainty in language models have focused on the calibration of classifiers (e.g., applying dropout to the model parameters or leveraging ensemble voting) to better assess the confidence of the generated output. When it comes to the era of LLMs, multiple works [180, 201, 104, 87, 47] have been proposed to measure the uncertainty of LLM's prediction in multiple aspects (e.g., lexical uncertainty, text uncertainty, and semantic uncertainty) for multiple NLP tasks. Another line of works [77, 4, 22] instead tries to analyze how to extract knowledge from a language model correctly and self-evaluate the correctness with a confidence score. However, despite these commendable efforts, existing methods still lack an effective way to directly quantify and decompose the uncertainty inherent in the outputs of LLMs with in-context learning.

3.2.3 Experiments

We evaluate the uncertainty decomposition procedure on realistic natural language understanding problems. By comparing state-of-the-art uncertainty quantification methods, we aim to examine what type of uncertainty is the most promising indicator of high confidence for LLMs. In addition, we also provide generalization analysis and two specific out-of-distribution detection applications. Extra experiments and experiment settings are provided in the Appendix.

Experiment Setup

We evaluate the decomposed uncertainties on open-source LLMs with different model sizes. We leverage LLaMA-2 [162], which is the most widely applied open LLM, with its 7B, 13B, and 70B model sizes. The primary experiments are conducted with LLaMA-2 models. In order to further demonstrate the generalization ability of our method, we apply our uncertainty quantification method on OPT-13B [206].

Data. We consider different Natural Language Understanding tasks. *1) Sentiment Analysis*: Emotion [145] contains $2,000$ test cases and six classes; Financial Phrasebank (Financial) [124] contains 850 financial news and three sentiment classes; Stanford Sentiment Treebank v2 (SST2) [151] consists of 872 sentences from movie reviews and two classes. *2) Linguistic Acceptability:* The Corpus of Linguistic Acceptability (COLA) [174] is about English acceptability judgments, which has $1,040$ test cases and two classes. *3) Topic Classification:* AG_News [208] contains $1,160$ test cases and four classes.

Demonstration & Model Configuration Sampling. We evaluate each method by choosing two strategies to randomly sample in-context learning demonstrations. 1) *Random*: we randomly sample demonstrations (training instances with labels) from

the training set regardless of their labels. 2) *Class*: we randomly sample demonstrations but ensure there is at least one demonstration per label class. To generate various sequences based on one set of demonstrations, we adopt Beam Search with beam width = 10 to approximate the sampling process of $\Theta \sim q(\Theta)$.

Comparison Methods. We evaluate the following baseline uncertainty estimation methods: 1) *Likelihood-based Uncertainty* (Likelihood) [123] calculates the sum of log probabilities of all tokens generated from language models and normalizes it by the sequence length. 2) *Entropy-based Uncertainty* (Entropy) [179] calculates the entropy of the probability distributions of the generated tokens. 3) *Semantic Uncertainty* (Semantic) [87] is the most advanced entropy-based uncertainty estimation method, which groups generated sequences into clusters according to their semantic embeddings. The average entropy across all groups is viewed as the uncertainty score.

Evaluation Metrics. We show the prediction accuracy of each dataset. In addition, we leverage two standard metrics: the Area under Precision-Recall Curve (AUPR) and AUROC (ROC) to evaluate the uncertainty. AUPR calculates the area under the Precision-Recall curve. AP is high when both precision and recall are high, and low when either of them is low across a range of confidence thresholds. ROC represents the likelihood that a correct answer is selected. An ideal ROC rating is 1, whereas a random uncertainty estimate would yield ROC = 0.5.

Quantitative Analysis

We compare the performance of different methods in assessing the misclassification samples based on their perspective uncertainty scores. We follow the procedure: 1) We use LLMs to classify all examples in the dataset with different beam search branches and demonstrations; 2) we use different uncertainty quantification methods to obtain a score associated with each test instance; 3) we assign each example a 0 if

	Inference Model	ACC	Likelihood		Entropy		Semantic		Ours (EU)		Ours (AU)	
			AUPR	ROC	AUPR	ROC	AUPR	ROC	AUPR	ROC	AUPR	ROC
Emotion	LLaMA-7B-random	0.407	0.423	0.426	0.448	0.501	0.598	0.607	**0.688**	**0.667**	0.625	0.579
	LLaMA-7B-class	0.411	0.562	0.423	0.657	0.538	0.697	0.653	**0.745**	**0.696**	0.691	0.601
	LLaMA-13B-random	0.501	0.597	0.613	0.584	0.503	0.612	0.625	**0.645**	**0.681**	0.559	0.585
	LLaMA-13B-class	0.533	0.641	0.578	0.593	0.554	0.652	0.701	**0.622**	**0.686**	0.526	0.599
	LLaMA-70B-random	0.584	0.512	0.462	0.491	0.452	0.657	0.696	**0.667**	**0.713**	0.531	0.663
	LLaMA-70B-class	0.592	0.537	0.484	0.469	0.442	0.622	0.689	**0.659**	**0.721**	0.612	0.693
Financial	LLaMA-7B-random	0.379	0.821	0.532	0.728	0.438	0.715	0.624	**0.731**	**0.672**	0.669	0.582
	LLaMA-7B-class	0.397	0.593	0.505	0.548	0.362	0.732	0.699	**0.803**	**0.711**	0.753	0.589
	LLaMA-13B-random	0.476	0.894	0.571	0.652	0.463	0.705	0.545	0.718	0.512	**0.729**	**0.573**
	LLaMA-13B-class	0.477	0.752	0.594	0.692	0.531	0.694	0.543	**0.765**	**0.610**	0.758	0.592
	LLaMA-70B-random	0.530	0.816	0.509	0.754	0.493	0.679	0.688	**0.779**	**0.754**	0.734	0.642
	LLaMA-70B-class	0.537	0.668	0.469	0.623	0.439	0.774	0.649	**0.893**	**0.804**	0.739	0.659
SST-2	LLaMA-7B-random	0.856	0.149	0.636	0.135	0.587	0.244	0.593	**0.286**	0.683	0.205	**0.702**
	LLaMA-7B-class	0.897	0.230	0.666	0.196	0.579	0.253	0.577	0.248	**0.701**	**0.302**	0.673
	LLaMA-13B-random	0.866	0.268	0.472	0.204	0.467	**0.355**	0.712	0.314	0.677	0.326	**0.816**
	LLaMA-13B-class	0.928	0.178	0.425	0.113	0.439	0.343	0.631	**0.397**	**0.836**	0.367	0.639
	LLaMA-70B-random	0.932	0.091	0.597	0.137	0.475	0.258	0.565	**0.318**	**0.764**	0.298	0.571
	LLaMA-70B-class	0.938	0.132	0.552	0.185	0.531	0.312	0.679	0.331	**0.851**	**0.362**	0.697
COLA	LLaMA-7B-random	0.599	0.388	0.557	0.329	0.443	0.358	0.502	**0.416**	**0.562**	0.377	0.517
	LLaMA-7B-class	0.639	0.392	0.523	0.381	0.478	0.425	0.526	**0.473**	**0.587**	0.401	0.506
	LLaMA-13B-random	0.652	0.389	0.498	0.287	0.512	0.433	0.562	0.469	**0.572**	**0.488**	0.565
	LLaMA-13B-class	0.649	0.412	0.418	0.342	0.517	0.426	0.548	0.456	0.568	**0.523**	**0.641**
	LLaMA-70B-random	0.826	0.481	0.599	0.312	0.471	0.372	0.625	0.317	**0.716**	**0.329**	0.676
	LLaMA-70B-class	0.852	0.357	0.612	0.397	0.588	0.397	0.613	0.389	**0.727**	**0.425**	0.682

Table 3.5: The performance comparison on the misclassification rate based on the uncertainty score from each approach. For each dataset, correct predictions are labeled as 0 and incorrect ones are labeled as 1. We show the AUPR and ROC (the higher the better) based on the uncertainty score and misclassification rate with two types of demonstration selection strategy: Random and Class as well as three LLaMA model sizes: 7B, 13B, and 70B.

it was classified correctly or a 1 if it was misclassified; and 4) we calculate AUPR and AUROC based on the misclassification rate and uncertainty score. Ideally, misclassified samples should have higher uncertainty scores. The results are shown in Table 3.5. Note that our proposed method can decompose the uncertainty into epistemic uncertainty (EU) and aleatoric uncertainty (AU), we thus show the performance of EU and AU separately.

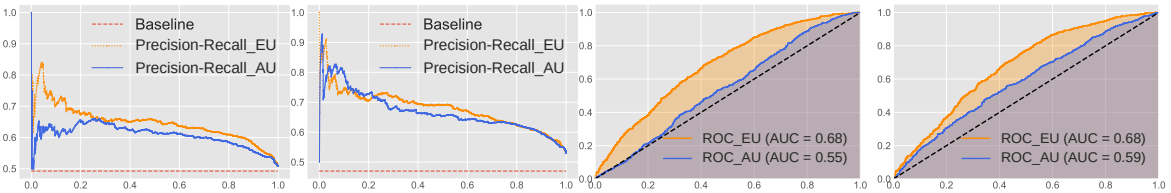
As shown in the table, in most cases, our proposed methods (EU and AU) consistently show higher AUPR and ROC scores across all datasets, which indicates a

better performance in assessing misclassification samples based on uncertainty scores. Moreover, we also draw some observations from the table. *1. Class Sampling Strategy Proves Superior*: The class sampling strategy generally yields higher AUPR and ROC scores across datasets, which proves it is more effective than random demonstration sampling. Class sampling ensures that each class is represented in the sample and reduces sampling bias, which is crucial in scenarios where the dataset might be imbalanced or where certain classes are underrepresented. *2) Increasing Model Size Enhances Performance*: Larger models (moving from 7B to 70B) tend to have better performance in terms of AUPR and ROC. Specifically, there's a general trend of increasing AUPR and ROC scores as model size increases from 7B to 13B to 70B for all comparison methods. Some datasets and metrics do not strictly follow this trend. For instance, in the EMOTION dataset, the 70B model sometimes shows a slight decrease in performance compared to the 13B model. The inconsistencies in performance improvement with larger models, especially for EU, hint at the complexity of uncertainty assessment in different contexts and datasets. *3. Treating all tokens equally can be harmful in uncertainty quantification*: both Likelihood and Entropy Uncertainty treat all tokens equally. However, some tokens carry greater relevance and representativeness than others, owing to the phenomenon of "linguistic redundancy". However, most uncertainty estimators treat all tokens equally when estimating uncertainty, disregarding these inherent generative inequalities.

Generalization Capability

We further show how our method performs when applied to different LLMs. We compare the performance of misclassification rate when using OPT-13B and LLAMA-2-13B. We compute the precision-recall (PR) curve and ROC curve using two backbone LLMs on the EMOTION dataset, and the results are shown in Figure 3.10.

As shown in Figure 3.10, our method exhibits consistent trends across different



(a) PR by OPT-13B (b) PR by LLaMA-2-13B (c) ROC by OPT-13B (d) ROC by LLaMA-2-13B

Figure 3.10: The performance of misclassification rate using two backbone LLMs: OPT-13B and LLaMA-2-13B on Emotion dataset. (a) and (b) demonstrate the precision-recall curves (x-axis is the recall and y-axis is the precision) for OPT-13B and LLaMA-2-13B; (c) and (d) demonstrate the ROC curve (x-axis is the false positive rate and y-axis is the true positive rate) for OPT-13B and LLaMA-2-13B.

LLMs. The precision-recall curves of both uncertainties (Figure 3.10 (a) and 3.10 (b)) between the two methods are almost identical, and the model's capability between two LLMs is also reflected in the PR curves of EU. Furthermore, by comparing Figure 3.10 (c) and 3.10 (d), the ROC curves of both LLMs show a similar pattern, with the AUC scores not deviating significantly. Specifically, both OPT-13B and LLaMA-2-13B exhibit the same Area Under ROC (AUROC) curve = 0.68 for AU. Since LLaMA-2-13B is a more powerful LLM than OPT-13B, our method can quantify that EU of LLaMA-2-13B (AUROC = 0.59) is better than OPT-13B (AUROC = 0.55). This finding further supports our method maintains its performance irrespective of the underlying model and its robust generalization capability.

Misclassification Rate with Out of Domain Demonstration

Out-of-domain in-context Demonstration refers to the test instance being coupled with less relevant or out-of-domain demonstrations, which the model may be misled and not handle the test instance reliably. In this work, we analyze the misclassification rate of out-of-domain Demonstration in the Emotion dataset (six-class sentiment analysis task) by providing LLMs with relevant demonstrations (sampled from Finance Phrasebank which is a three-class sentiment analysis task) and complete out-of-domain demonstrations (sampled from COLA which is a binary linguistic

acceptability task). We conduct the task with two demonstration selection strategies, and the results are provided in Table 3.6.

	LLaMA-13B-Random		LLaMA-13B-Class	
	EU	AU	EU	AU
Original Demo	0.681	0.585	0.686	0.599
Relevant Demo	0.688 (+1.0%)	0.541 (−7.5%)	0.671 (−2.2%)	0.524 (−12.5%)
OOD Demo	0.671 (−1.4%)	0.501 (−13.3%)	0.673 (−1.8%)	0.497 (−17.0%)

Table 3.6: Comparison of AUROC in misclassificatin rate on EMOTION dataset, where "Original Demo" indicates we sample demonstrations from its original training set, "Relevant Demo" indicates we sample demonstrations from Finance Phrasebank Dataset (a relevant sentiment analysis task, and "OOD Demo" indicates we sample demonstrations from an irrelevant dataset: COLA.

As shown in the table, changes in the performance of the EU are relatively minor under all conditions, suggesting that the model is more stable or less sensitive to the changes in demonstration data within this metric. In contrast, the AU shows more significant fluctuations, which implies that the AU is more sensitive to the quality and relevance of demonstration data. When relevant demonstrations from the Finance Phrasebank sentiment analysis dataset are used, there's a slight improvement or a minor decrease in EU, but a notable decrease in AU. This suggests that even relevant but not identical data can confuse the model, especially for the AU. With out-of-domain demonstrations from COLA, the model's performance drops more significantly, with the AU metric showing a dramatic decrease of up to 17.0%, which indicates that the model struggles significantly when the demonstrations are not relevant to the task it's being tested on.

Out-of-domain Demonstration Detection

Out-of-domain (OOD) demonstration refers to coupling a test instance with less relevant or OOD demonstrations, potentially leading the model to be misled and handle the test instance unreliably. In this study, we investigate whether uncertainty scores

	Semantic		Ours (EU)		Ours (AU)	
	AUPR	ROC	AUPR	ROC	AUPR	ROC
Relevant Demo	0.702	0.644	**0.742**	**0.935**	0.657	0.682
OOD Demo	0.698	0.712	**0.784**	**0.941**	0.773	0.607

Table 3.7: Out-of-domain demonstration detection conducted with LLAMA-2-13B on EMOTION Dataset.

can effectively distinguish between in-domain and OOD demonstrations. In our labeling scheme, in-domain demonstrations are labeled as 0, while OOD demonstrations are labeled as 1. AUPR and ROC analyses are performed based on the labels and uncertainty scores, with results summarized in Table 3.7. Specifically, we conduct experiments on the EMOTION dataset, involving two scenarios: in-domain demonstrations (sampled from its training set) and relevant demonstrations (sampled from Finance Phrasebank, a three-class sentiment analysis task). Additionally, we compare in-domain demonstrations with complete OOD demonstrations (sampled from COLA, a binary linguistic acceptability task).

As shown in Table 3.7, compared to the state-of-the-art Semantic Uncertainty and the AU, the EU demonstrates the best indicator to detect both less relevant and OOD demonstrations. Intuitively, the model's predictions would be impacted by irrelevant and OOD demonstrations and exhibit large variance. AU is less effective than EU in detecting OOD demonstrations since the demonstrations already have large inherent variability. Semantic Uncertainty instead cannot really distinguish what is the root cause of the predictive uncertainty.

3.2.4 Semantic Out-of-distribution Detection

Semantic out-of-distribution (SOOD) detection refers to distinguishing test samples with semantic shifts from the given demonstrations and the prompt. In this study, we mask out a few classes and ask LLMs to classify test samples into the rest of the classes. The method is expected to return a higher uncertainty score of SOOD test

	Semantic		Ours (EU)		Ours (AU)	
	AUPR	ROC	AUPR	ROC	AUPR	ROC
7B	0.477	0.532	**0.548**	**0.658**	0.461	0.570
13B	0.417	0.468	**0.525**	**0.592**	0.414	0.437

Table 3.8: Semantic out-of-distribution detection using LLAMA-2 7B and 13B on EMOTION Dataset.

samples. Specifically, we mask two classes *1: sadness* and *2: anger* out of six classes from the EMOTION dataset and ask LLM to categorize a given test sample only into the rest four classes. The SOOD samples are labeled as 1 and in-distribution samples are labeled as 0. Results of AUPR and ROC are recorded in Table 3.8 in terms of different model sizes.

As shown in the table, EU still performs the best as a better indicator to recognize SOOD samples across different model sizes. SOOD samples are semantically different from the provided demonstrations, and the task description also masks out the correct class of these SOOD samples, leading to higher uncertainty in the model's predictions. Given the inappropriate task description and demonstrations, AU may not necessarily perform better in the presence of SOOD samples.

3.2.5 Conclusion

We provide a novel approach to decompose the predictive uncertainty of LLMs into its aleatoric and epistemic perspectives from the Bayesian perspective. We also design novel approximation methods to quantify different uncertainties based on the decomposition. Extensive experiments are conducted to verify the effectiveness and better performance of the proposed method than others. We believe this research stands as a significant stride toward harnessing the full potential of LLMs while being acutely aware of their performance boundaries. For future works, we plan to extend our method to other forms of data [21] and tasks [210] to quantify the uncertainty.

Chapter 4

Representation Learning on Textual-edge Graphs for Link Prediction

In network science, the prevalence of networks with rich text on edges, also known as Textual-edge Graphs (TEGs) [194, 59], increasingly become significant, as they encapsulate a wealth of relational and contextual information critical for diverse applications. The text associated with edges in networks can dramatically deepen our understanding of network dynamics and behavior. For instance, in a social media network, when a user responds to another's post, the reply not only creates a directed edge but also includes specific text that can reveal the sentiment, intent, or relationship between users. Similar cases can also be found in citation networks where text on edges is the exact reference quote. Both examples illustrate how text-rich edges are pivotal in accurately interpreting and leveraging networked data for advanced analytical purposes. In TEGs, link prediction is a unique yet open question due to the rich textual information embedded on edges.

Extensive works have been devoted to studying graphs with rich text, which can

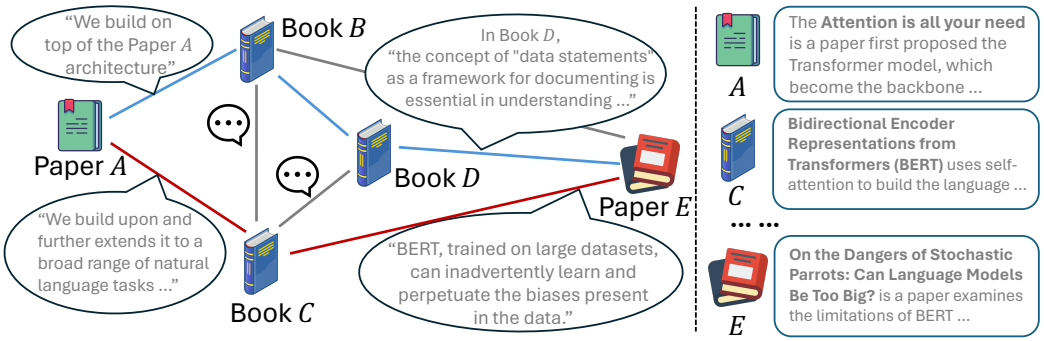


Figure 4.1: An example of textual-edge graphs: two books are connected by citation links. Predicting whether there'll be a citation between A and E needs to jointly consider both topology and semantic information embedded on nodes and their edges.

be categorized into *Graph Neural Networks (GNN)*-based and *language model*-based methods. GNN-based works have extensively studied the topology connection between nodes and designed various variants. Specifically, works [59, 217, 53] typically compress the text embedded on edges to latent vectors by text encoders (e.g., Word2Vec [128] and BERT [38]), and iteratively merge edge features with/without node features [72, 196]. The current most advanced edge-aware GNN [75] tries to refine the text encoder with the GNN training to obtain better representation. However, it still follows the neighbor aggregation way, which may not comprehensively consider overall semantics. On the other hand, owing to the strong text understanding ability of LLMs [115], researchers [28, 48, 199, 186] have directly used language models to solve graph mining tasks on textual graphs by designing various prompts to express or summarize the topology connection into natural language.

While these approaches have advanced the study of text-rich graphs, they tend to simplify the diverse text on edges, potentially losing crucial information necessary for tasks such as *link prediction*, where edge text is key to understanding the relationships within the graph. Both GNN-based and LLM-based methods may fall short of addressing the link prediction on TEGs due to two challenges, respectively.

Challenge 1: Understand graph topology in language models. For LLM-based approaches, existing works have worked on prompting LLMs by expressing or summarizing graph topology into text, but the graph topology is generally expressed in a linear shape, which may lead to a significant loss of graph-based structural information. For example, in Figure 4.1, if we summarize the relation between Book A and E as Breadth-first Search, the dependencies along specific paths would be overlooked. This approach would not capture the multi-hop interactions and the rich contextual dependencies among nodes and edges. On the other hand, if we summarize all paths from A to E, the same edge, such as the negative reference on edge C to E, could appear in several paths. Representing this repeated edge in text multiple times leads to unnecessary duplication and potential noise, complicating the model's ability to discern the true nature of the relationships. If the graph is too large, potential overflow of the language model's context window limitation may also emerge.

Challenge 2: Comprehensively consider context information on all connections. For GNN-based methods, predicting links in TEGs requires a comprehensive examination of all potential paths connecting two nodes. Neighborhood aggregation approaches often focus on immediate neighbors and fail to account for the complex interactions that can occur in textual-edge graphs. As shown in Figure 4.1, there are many multi-hop paths with different intermediate nodes between A and B, and each edge is annotated with different contexts, including user descriptions and user comments, describing their complex relations. It's already hard for existing GNN-based methods to use one latent vector to describe the complex context on edges. Moreover, user's preferences may also differ from one path to another (i.e., $A \rightarrow C \rightarrow E$ is negative while $A \rightarrow B \rightarrow D \rightarrow E$ is positive), creating a conflicting semantic landscape. However, GNN's neighbor aggregation would treat these paths uniformly, potentially diluting the sentiment difference in these paths.

Present Work. To effectively make link predictions on TEGs by jointly considering rich semantic information and graph topology, in this paper, we propose a novel representation learning framework, LINK2DOC, that transforms local connections between nodes into a coherent document for better-reflecting graph topology along with semantic information. To process rich textual information efficiently, we further propose a stratified representation learning framework that captures multi-scale interactions between target nodes. The crafted document further enhances GNNs to make link predictions in a contextualized way. The key contributions are summarized as follows:

- **Problem**. We formulate the problem of link prediction on textual-edge graphs and highlight the unique challenges of learning representations on textual-edge graphs for link predictions.
- **Method**. We propose an integrated framework to jointly consider topology and semantic information in textual-edge graphs, which consists of 1) coherent document composition to summarize semantic relations between node pairs in plain language; 2) a specialized Transition Graph Neural Network to process topology information between target nodes in a stratified manner; and 3) a self-supervised learning module to combine semantic understanding and topology processing ability for better link prediction on textual-edge graphs.
- **Experiment**. We empirically compare our method against existing state-of-the-art in four real-world datasets. Results have shown our proposed methods can elevate the performance of general GNNs and achieve competitive performance against edge-aware GNNs.

4.1 Related Works

Edge-aware Graph Representation Learning. Earlier research of Graph Neural Networks (GNNs) [178, 33] tends to only focus on node features. Later on, research on heterogeneous graph representation learning [192] began to consider categorical information on edges. In text-attributed graphs, to more comprehensively utilize edge information during the network representation learning, some edge-aware GNNs [217, 53, 72, 196] were proposed to consider edge text by designing various architectures (e.g., attention on edges, node and edge role switch, etc.). The recent state-of-the-art method EdgeFormer [75] involved pre-trained language models and proposed to better consider edge text by designing a cross-attention mechanism to merge node and edge representation in Transformer layers. However, these approaches still use the neighborhood aggregation way to obtain graph representation and cannot consider the local connections as a whole unit. Neighborhood aggregation may not always work especially when nodes are dissimilar (as shown in Figure 3.1) [182]. Moreover, existing edge-aware GNNs tend to deliberately erase text on nodes and only explore the effect of edge information on various graph-related tasks [75], which lacks the flexibility to extend to other text-attributed scenarios.

Language Modeling Augmented Graph Learning. Large language models have been proven to have the ability to interpret graph-structured data [70, 56, 76]. In the past year, many works [27, 28, 66, 134] have been proposed to prove LLMs have great potential (and even become state-of-the-art) to classify nodes in text-attributed graphs. However, how LLMs can better assist link prediction in text-attributed graphs is still an under-explored area, let alone the more complicated scenario of edge-attributed graphs that contain rich textual information on both edges and nodes. Existing works [218, 211, 96, 175] have tried to distill implicit knowledge from LLMs to smaller GNN models for text-attributed graph tasks, but they still

focus on learning good node embeddings. In edge-attributed graphs, text on edges cannot be uniformly processed in the same manner as node text, necessitating more specialized techniques that account for the unique semantics and structural roles of edge attributes in enhancing graph-based learning models.

4.2 Link Prediction on Textual-edge Graph

We begin by introducing key notations and formulating the problem of link prediction on Textual-edge Graphs. We then describe a novel way of constructing a transition document that summarizes the relationship between node pairs for link prediction. Finally, we provide an LLM-enhanced Graph Neural Network framework that learns the local topology and semantic information to retain both efficiency and efficacy.

4.2.1 Problem Formulation

A Textual-edge graph (TEG) is a type of graph in which both nodes and edges contain free-form text descriptions. These descriptions provide detailed, contextual information about the relationships between nodes, enabling a richer representation of relational data than in traditional graphs.

Definition 3 (Textual-edge Graphs)**.** A TEG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an undirected graph, which consists of a set of nodes V and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Each node $v_i \in \mathcal{V}$ contains a textual description d_i, and each edge $e_{ij} \in \mathcal{E}$ also associates with free-form texts d_{ij} describing the relation of (v_i, v_j).

In this work, we target the *Link Prediction* task on TEGs, where we aim to predict the existence (or the label) of edges between pairs of nodes $(v_i, v_j) \notin \mathcal{E}$ based on the neighborhood information of (v_i, v_j). Due to the rich edge text information, local edges in TEGs can inherently be represented by natural language sentences. For

example, the connection $v_i \rightarrow e_{ij} \rightarrow v_i$ can be represented as "d_i `is connected to` d_j `via` d_{ij}", which directly describes the relation in plain text.

Compared to graphs that have edges labeled with simple, predefined categories, TEGs feature edges annotated with free-form text, which offer detailed and contextual relationship descriptions. Take Figure 4.1 again as an example, books are connected by textual edges, where text on edges consists of exact quotations that one book cites another. To predict whether A and E will have a citation link, we not only need to analyze semantics embedded within each edge's description, but different paths may also depict different semantic meanings due to the varying textual descriptions and types of relationships they represent. As noted in Figure 4.1, the Red Path contains negative reference from A to E, while the Blue Path indicates another group of researchers endorse the research conducted by book A.

Challenge. The rich and complex text on edges makes the link prediction on TEGs not a trivial task, and there are two essential difficulties regulating existing *GNN-based* and *LLM-based* methods, respectively. For edge-aware GNN-based methods, directly combine and update each node v_i's feature based on its neighbor's $\mathcal{N}_{v_i}(v_j)$ features as well as features of edges e_{ij}. However, neighborhood aggregation would fall short since semantics carried on each edge d_{ij} needs to be viewed in the context of the whole connections from s to t. For LLM-based methods, existing works tend to prompt language models by linearly summarizing graph topology, e.g., "$G_{(s,t)}$ contains $s, v_1, v_2, ..., v_n, t$ nodes, v_1 is connected to v_2 via d_{12}, v_2 is connected to v_4 via d_{24}, etc." This way may let LLMs fail to understand how information propagates from s to t and the contextual dependencies among nodes.

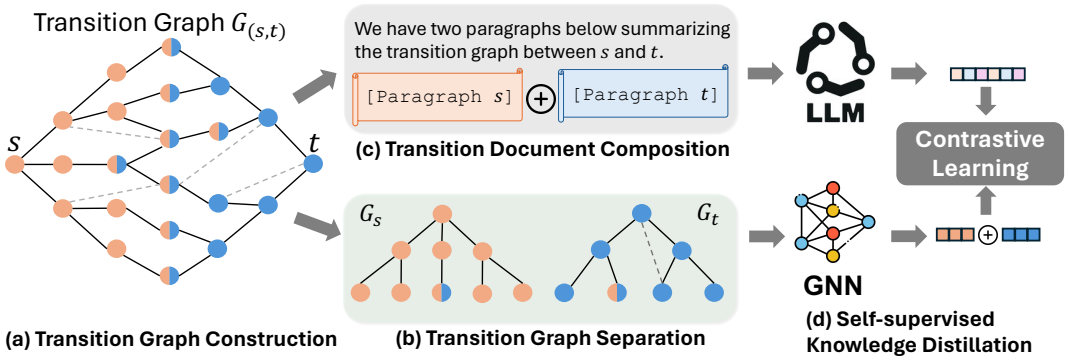


Figure 4.2: Overall framework of LLM-enhanced link prediction on Textual-edge graphs, where orange and blue nodes in $G_{(s,t)}$ belong to s's and t's local neighborhood (namely G_s and G_t), respectively. Half blue and half orange nodes denote shared nodes between G_s and G_t.

4.2.2 Overall Architecture

In this work, we introduce LINK2DOC, a novel approach that leverages a self-supervised learning scheme to endow GNNs with text comprehension capabilities akin to those of LLMs. LINK2DOC is designed to preserve and synergize rich semantic information, topology information, and their interplay within TEGs for link prediction. We propose learning and aligning representations from two complementary perspectives: the *text view* and the *graph view*. The *text view*, termed *Text-of-Graph*, organizes the text associated with TEG's nodes in a way that reflects the graph's topology, forming a structured document that inherently captures logical and relational data. Conversely, the *graph view*, or *Graph-of-Text*, transforms the nodes and topology of TEGs into structured graph data. By employing pretrained language models (PLMs), the text view adeptly maintains textual integrity, while the graph view, processed through GNNs, ensures the retention of graph-specific characteristics. Aligning these views allows each representation to enrich the other, fostering a holistic understanding where textual nuances inform graph structures and vice versa.

Specifically, as noted in Figure 4.2 (a), to reduce the search space and to eliminate the noise from unrelated connections, we propose to formulate a (s, t)-transition graph containing all the possible routes through which s could correlate to t.

Definition 4 (Transition Graph)**.** For any pair of two entities (s, t) in the Textual-edge Graph, all paths from s to t collectively form an (s, t)-transition graph, which is denoted by $G_{(s,t)}$. We use n and m to denote the number of nodes and edges in $G_{(s,t)}$, respectively. Figure 4.1 exemplifies an (s, t)-transition graph, where s is Book A and t is Book B. In practice, the length of paths can be upper-bounded by an integer K, which can usually be set as the diameter of the Textual-edge graph.

The transition graph $G_{(s,t)}$, as well as all the text on edges, provide the necessary information needed to understand the relation between s and t. Next, as shown in Figure 4.2(b), we separate two subgraphs: G_s and G_t from $G_{(s,t)}$ for preserving local neighborhood of s and t, respectively. In Figure 4.2(c), to view the topology and semantic information of $G_{(s,t)}$ in a joint unit, we compose a manageable and coherent document that expresses $G_{(s,t)}$ as a human-written document. Finally, in Figure 4.2(d), we distill the text processing ability from large language models to graph neural networks for inference scalability while maintaining performance.

4.2.3 Transition Document Construction

In this work, to address the first challenge, we need to summarize local relations in $G_{(s,t)}$ to comprehensively understand the relation between (s, t). Since state-of-the-art LLMs are predominantly trained on human-written documents and books, in this work, we propose a novel way to express $G_{(s,t)}$ as a structured document $d_{(s,t)}$ [89], complete with an introduction, sections, subsections, and a conclusion, to let language models better understand the overall semantics between s and t with preserving topology. The overall process is illustrated in Figure 4.3.

Node-centric Paragraph Composition. For source node s and target node t, we first conduct Breadth-first Search (BFS) to extract their respective local structure with depth L, i.e., G_s and G_t from their transition graph $G_{(s,t)}$. As shown in Figure

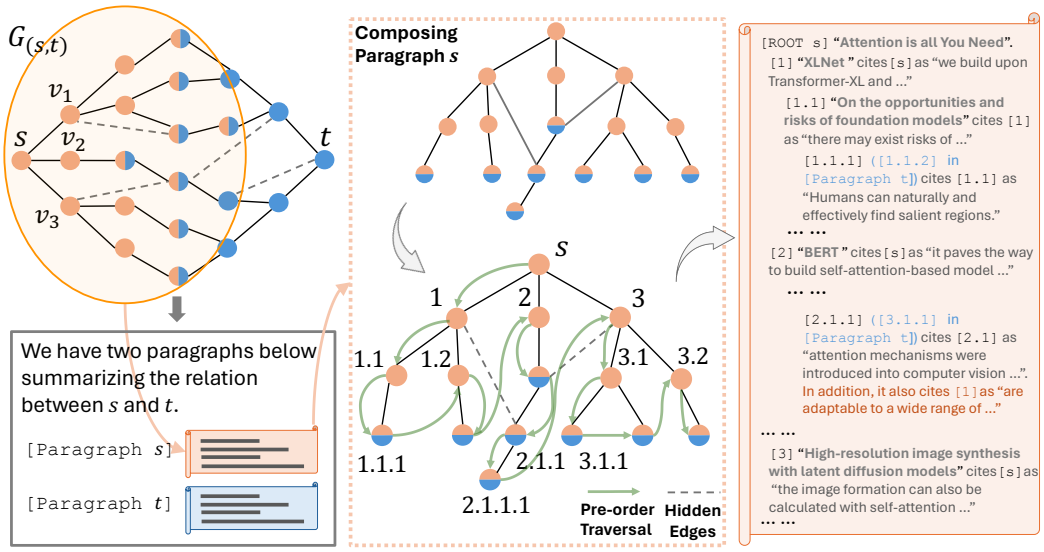


Figure 4.3: We first split $G_{(s,t)}$ into G_s (nodes are marked with orange) corresponding to the local structure of s (G_t is omitted due to space limit). Commonly shared nodes are marked with half blue and half orange. We transform the local structure of G_s into a paragraph that summarizes hierarchical relation with s being the root. For better visibility, hidden edges are highlighted with orange, and commonly shared nodes are highlighted with blue.

4.3, nodes in $G_{(s,t)}$ that belong to G_s are marked with orange, and nodes belonging to G_t are marked with blue. The structural data (i.e., BFS tree) obtained is transformed into a textual paragraph aimed at both human readability and machine processability. Specifically, taking the BFS tree G_s rooted at source node s as an example, s acts as the main subject of the paragraph's textual summary. We conduct pre-order traversal to go over all nodes in both trees. The first and subsequent levels of BFS neighbors are detailed in separate sections and subsections, akin to a detailed outline:

1. **Root**: Start with a sentence that describes node s and its immediate connections: "`[s] Root` s `has three connections.` s `is connected to` v_1 `via` d_{s1}`,` s `is connected to` v_2 `via` d_{s2}`, and` s `is connected to` v_3 `via` d_{s3}``''``, where d_{s1}, d_{s2}, and d_{s3} are textual descriptions on edges e_{s1}, e_{s2}, and e_{s3}.

2. **First-hop Neighbors**: For each first-hop neighbor of s, we provide a section detailing its connections: "`[1].` `Node` v_1 `has two connections.` v_1

`is connected to` v_{11} `via` d_{11}`, and` v_1 `is connected to` v_{12} `via` d_{12}"; "`[2]. Node` v_2 `has one connection.` v_2 `is connected to` v_{21} `via` d_{21}"; and "`[3]. Node` v_3 `has two connections.` v_3 `is connected to` v_{31} `via` d_{31}`, ...`".

3. **Second (and following)-hop Neighbors**: Subsections under each first-hop neighbor detail further connections: "`[1.1]` v_{11} `is connected to ...`" "`[1.2]` v_{12} `is connected to...`", etc.

This structure is recursively applied up to L levels deep, ensuring each node's direct connections are thoroughly described, capturing the intricate topology of the graph. The notation "`[X]`" refers to earlier parts of the summary where the connected node was initially described, aiding in understanding the network's connectivity beyond a simple hierarchical structure.

Hidden Edges. The neighborhood of s and t may not always form tree structures. As shown in Figure 4.3, node v_{122} is not only the child node of v_{12}, v_{122} also links back to v_1 to form a triangle structure. To consider a more holistic view of the node relationships, we add an extra description to the node stating its connection to pre-existing nodes. Specifically, we introduce the hidden edge information of node v_{122} in `[1.2.2]` as "`In addition,` v_{122} `is also linked to [1]` v_1 `via ...`". By letting each mention of a hidden edge direct back to the respective section "`[X]`", we aim to ensure clarity and maintain the coherence of the graph's description.

Transition Graph Document Construction. The unified $d_{(s,t)}$, which consists of paragraphs from s and t, aims to not only present isolated descriptions but also to highlight the interconnected nature of G_s and G_t. In this work, we aim to illuminate the interconnectedness between G_s and G_t by identifying and highlighting nodes that appear in both G_s and G_t's local structures. These common nodes are pivotal as they link the context of one paragraph to the other. As shown in Figure 4.3, the

common nodes are marked with half orange and half blue. For each common node, we include a cross-reference in the text where the node is mentioned, which is done by adding a note after the section index. For example, node v_{122} has a section index "`[1.2.2]`" in s's paragraph, and a section index "[`1.2.1`]" in t's paragraph. We then add `([1.2.1] in Paragraph` t`)` after the section index of v_{121} in s's paragraph. We conduct the cross-reference in the other paragraph reversely.

In practice, we keep the depth L to be half of the diameter of the $G_{(s,t)}$ so that G_s and G_t can each cover their close neighbor information as well as an adequate number of common nodes. We further enhance the document's coherence by adding an introduction to the start of the document. Finally, the generated $d_{(s,t)}$ can be viewed as a structured document. More details can be found in Figure 4.3.

4.2.4 Transition Graph Neural Network

After obtaining a document $d_{(s,t)}$ summarizing both topology and semantic information of $G_{(s,t)}$, scalability still poses a significant challenge for language models on large-scale graphs. To conduct link prediction on a large $G_{(s,t)}$, we need to compose many documents between node pairs with duplicated content (e.g., $d_{(s,t_1)}$ and $d_{(s,t_2)}$ may largely overlap if t_1 and t_2 are neighbors).

On the other hand, GNNs are inherently designed to process graph structures efficiently, making them a promising alternative for this task. Although LLMs may not be capable of conducting large-scale link predictions, the implicit knowledge can still be utilized to train GNNs. However, a straightforward application of GNNs faces limitations: a single GNN may not fully capture the intricate interplay between the graph's structural properties and the semantic information on nodes and edges, especially in large graphs where $G_{(s,t)}$ between two nodes s and t can encompass thousands of nodes due to a diameter as small as 4 [113]. Moreover, independently learning local representations for s and t fails to account for the multi-scale interactions crucial for

accurate link prediction. Each hop in the graph can reveal different structural and semantic information—immediate neighbors contribute local properties, while nodes farther away provide broader contextual insights.

Transition Graph Neural Network (TGNN). To better process rich textual information on large-scale graphs efficiently, we propose TGNN and introduce a novel stratified representation learning framework to captures multi-scale interactions between target nodes s and t by considering different "cuts" in the transition graph $G_{(s,t)}$. Each cut is defined by a pair $(n, K-n)$, where K is the diameter of the $G_{(s,t)}$ between s and t, i.e., the longest path length between s and t in $G_{(s,t)}$. For each cut, TGNN encompasses two directed graph convolution processes (with shared parameters): one focuses on learning the representation of s of its n-hop neighborhood, and the other focuses on t's $(K-n)$-hop neighborhood. The TGNN update function at n-th layer for node u is given as:

$$\mathbf{h}_u^{(n)} = f_\theta\left(\mathbf{h}_u^{(n-1)}, \text{AGG}\left(\{\mathbf{h}_v^{(n)}, e_{u\to v} : v \in \mathcal{N}(u)\}\right)\right), \tag{4.1}$$

where v is the child node of u, θ is the parameter of the TGNN update function, $\text{AGG}(\cdot)$ is the aggregation function of TGNN. By calculating Equation (4.1) for $n = 1, 2, \cdots, K-1$, we will obtain all the embeddings of s and t for all the cuts, namely $\{(\mathbf{h}_s^{(n)}, \mathbf{h}_t^{(K-n)})\},\ n = \{1, 2, \cdots, K-1\}$.

Accelerating Transition Graph Representation Learning by Deduplicating TGNN Computation. However, naively implementing all K cuts would necessitate $2K$ times the calculation of the whole TGNN on the entire graph, resulting in duplicated computation which will be prohibitive, especially for non-small graphs that are common in the real world. To mitigate this, we exploit the hierarchical nature of our transition graph to re-order the message-passing process as a cascading

process from higher-hop neighbors of s (or t) progressively to the lower and lower layers. The TGNN update function of cut n for node s given as:

$$\begin{aligned}
\mathbf{h}^{(n)}_{u\in\mathcal{L}_n} &= \mathbf{x}_u, \\
\mathbf{h}^{(n)}_{u\in\mathcal{L}_{n-1}} &= f_\theta(\mathbf{h}^{(n-1)}_{u\in\mathcal{L}_{n-1}}, \mathrm{AGG}(\{\mathbf{h}^{(n)}_v, e_{u\to v} : v \in \mathcal{N}_{\mathcal{L}_n}(u)\})), \\
\mathbf{h}^{(n)}_{u\in\mathcal{L}_{n-2}} &= f_\theta(\mathbf{h}^{(n-2)}_{u\in\mathcal{L}_{n-2}}, \mathrm{AGG}(\{\mathbf{h}^{(n-1)}_v, e_{u\to v} : v \in \mathcal{N}_{\mathcal{L}_{n-1}}(u)\})), \\
&\vdots \\
\mathbf{h}^{(n)}_s &= f_\theta(\mathbf{h}^{(n-1)}_s, \mathrm{AGG}(\{\mathbf{h}^{(n)}_v, e_{u\to v} : v \in \mathcal{N}(u)\})),
\end{aligned} \tag{4.2}$$

where v and u are nodes in the transition graph $G_{(s,t)}$ and $\mathcal{L}_n$ means all the n-hop neighbors of s. The computation of $\mathbf{h}^{(n)}_s$ follows the cascading order that aggregates higher-hop neighbors' information first, as shown in Eq. (4.2). Since the cascading (a.k.a, Eq. (4.2)) for cut n actually uses the node embeddings calculated in cut $n-1$, we, therefore, propose to calculate the cascading processes in a sequential order $n = 1, 2, \cdots, K-1$ so the cut $n-1$'s computation is naturally used for cut n, saving the later from re-compute. Therefore, the entire calculation for all the K cuts for s is the same as one calling of TGNN and hence we accelerate the compute by K times.

Aligning Topology Representation of $G_{(s,t)}$ with Semantic Information. To bridge the gap between the semantic understanding capabilities of LLMs and the structural learning strengths of the TGNN, in this work, we leverage LLMs to generate embeddings $\tilde{\mathbf{h}}_{(s,t)}$ of $d_{(s,t)}$ to guide the training of TGNN in a self-supervised learning manner:

$$\tilde{\mathbf{h}}_{(s,t)} = f_{LM}\left(d_{(s,t)}\right),\ \mathbf{h}_{(s,t)} = g(\bar{\mathbf{h}}_s \oplus \bar{\mathbf{h}}_t),\ \bar{\mathbf{h}}_s = \frac{1}{K-1}\sum\nolimits_{n=1}^{K-1}\mathbf{h}^{(n)}_s,\ \bar{\mathbf{h}}_t = \frac{1}{K-1}\sum\nolimits_{n=1}^{K-1}\mathbf{h}^{(n)}_t$$

where $\oplus$ denotes embedding concatenation, $f_{LM}(\cdot)$ denote the LLM query function. We transform text on nodes and edges with pre-trained language models (e.g., LLaMA

models or OpenAI's Embedding Models) and feed these attributes along with adjacency matrix of G_s and G_t to GNNs. The outputs are then concatenated and transformed into LLM's embedding space by a projection function $g(\cdot)$ with non-linear transformation. We seed to align the latent embeddings $\tilde{\mathbf{h}}_{(s,t)}$ produced by the LLM with the embeddings $\mathbf{h}_{(s,t)}$ generated by the GNN:

$$\ell_{KD} = -\mathbb{E}\left[\log \frac{\exp\left(\text{sim}(\tilde{\mathbf{h}}_{(s,t)}, \mathbf{h}_{(s,t)})/\tau\right)}{\sum_{k=1}^{K} \exp\left(\text{sim}(\mathbf{h}_{(s,t)}, \mathbf{h}_{(s,h)})/\tau\right)}\right], \tag{4.3}$$

where the objective function is based on temperature-scaled cross-entropy loss (NT-Xent) [24] to enforce the agreement between $\tilde{\mathbf{h}}_{(s,t)}$ and $\mathbf{h}_{(s,t)}$ compared with latent embedding $\mathbf{h}_{(s,h)}$ from negative pairs. Furthermore, to calibrate $\mathbf{h}_{(s,t)}$ more towards the link prediction (and edge classification) task, we incorporate standard binary cross-entropy loss ℓ_{LP} for tuning GNNs. Note that for dealing with highly imbalanced label distribution for edge classification tasks, we use weighted cross-entropy loss (e.g., Focal Loss [102]) instead.

Finally, the overall objective of the LLM-enhanced Representation Learning for predicting links on edge-attributed graphs is written as $\ell = \lambda_1 \ell_{KD} + \lambda_2 \ell_{LP}$, where λ_1 and λ_2 are hyperparameters.

Complexity Analysis. Given the transition graph $G_{(s,t)}$ with the diameter K, we use Breadth-first Search (time complexity $O((N+E)/2)$ (BFS) with the depth $K/2$ to extract G_s and G_t, respectively. We then use pre-order traversal to obtain the document $d_{(s,t)}$, which leads the total complexity to be $O(N+E+N)$. The time complexity for Pre-trained LMs to process $d_{(s,t)}$ is $O(P^2)$, where P denotes the number of tokens in $d_{(s,t)}$. Moreover, the GNN module requires $O(|E| \cdot f + N^2)$, where f denotes the dimension of the embedding on nodes/edges. Overall, The complexity encapsulates the stages of BFS tree construction, document processing with Transformers, and GNN

learning, which gives the training time complexity $O(2N + E + P^2 + |E| \cdot f + N^2)$. However, during the inference stage, the complexity of our work simply reduces to the complexity of normal GNNs: $O(|E| \cdot f + N^2)$ as we do not need to construct documents during the inference phase.

4.3 Experiment

4.3.1 Experiment Setup

Setup. This paper focuses on link prediction on TEGs, which aims to predict whether there will be a strong connection between two nodes in the adopted datasets based on their transition graph. We run experiments on five real-world networks: Amazon-Movie [63], Amazon-Apps [63], GoodReads-Children [166], GoodReads-Crime [166], and StackOverflow. More specific dataset statistics can be found in the Appendix **??**. We evaluate the performance using four standard metrics: Mean Reciprocal Rank (MRR), Normalized Discounted Cumulative Gain (NDCG), Area Under ROC Curve (AUC) metric, and F1 score.

Comparison Methods. We compare our model with *general GNNs*, *language model integrated GNNs*, and *large language models*. For *general GNNs*, we select MeanSAGE [61], MaxSAGE [61], GIN [183] and RevGAT [91], which only use an adjacency matrix as the input. For *language model-enhanced GNNs*, we utilize Pre-trained LMs, e.g., BERT [38], to acquire text representations on edges. Our baselines consist of BERT + Graph Transformer (GTN) [202], BERT + GINEConv [64] and BERT + EdgeConv [171]. Furthermore, we also incorporate state-of-the-art edge-aware GNN - Edgeformer [75], which is constructed based on graph-enhanced Transformers to combine language modeling into each layer of the Graph transformer. A novel graph foundation model - THLM [218] is also included that integrates language modeling

with GNN training. Finally, we adopt state-of-the-art LLMs, i.e., LLaMA-3-70B and GPT-4o by directly translating the transition graph $G_{(s,t)}$ between the node pair (s, t) to natural language as [48] do. Note that state-of-the-art *language model integrated GNNs*, namely EdgeFormer, cannot incorporate text on nodes. For a fair comparison, we present two variants of our method: Link2Doc does not consider node texts, and Link2Doc-NT takes node text into account.

Implementation Details. To process all node and edge text, we leverage OpenAI's embedding model[1] with dimension $3,072$. For both *general GNNs* and *language model enhanced GNNs*, the dimensions of the initial node and edge embeddings are further normalized to 64 and 128 respectively. Additionally, for the Edgeformer model, we adhere to the same experimental settings as outlined in [75]. Our model uses Graph Transformer (GTN) as our backbone in Eq. (4.1), where both node and edge embeddings mirror those of *language model integrated GNNs*. For Link2Doc, we follow the same settings as general GNNs to obtain node and edge embeddings from OpenAI's embedding model. We set $\lambda_1 = 1$ and $\lambda_2 = 2$ respectively. All GNN baseline layers are set to 2. The temperature τ in Eq. (4.3) to 2. We use Adam as the optimizer with a learning rate of $1e-5$. The batch size is $1,024$. We run our model and other baselines 10 times with different random seeds and report the average performance.

4.3.2 Results on Link Prediction

As can be seen from Table 4.1 and Table 4.2, Link2Doc can consistently achieve better performance than other methods. Specifically, Link2Doc outperforms the second best on average 5% of both AUC and F1 (Table 4.1) and 10% of both MRR and NDCG across all datasets (Table 4.2). We further draw several observations from the results. *1) There are no clear differences between general GNNs and edge-*

[1] https:/platform.openai.com/docs/guides/embeddings/embedding-models

	Goodreads-Children		Goodreads-Crime		Amazon-Apps		Amazon-Movie		StackOverflow	
	AUC	**F1**	**AUC**	**F1**	**AUC**	**F1**	**AUC**	**F1**	**AUC**	**F1**
General GNN										
MaxSAGE	0.870	0.637	0.858	0.624	0.727	0.571	0.706	0.527	0.895	0.631
MeanSAGE	0.828	0.611	0.829	0.615	0.700	0.569	0.686	0.525	0.887	0.615
RevGAT	0.862	0.622	0.839	0.619	0.662	0.562	0.689	0.541	0.819	0.533
GIN	0.859	0.571	0.857	0.577	0.705	0.543	0.692	0.512	0.873	0.605
LM-enhanced GNN										
GTN	0.880	0.654	0.863	0.640	0.728	0.572	0.742	0.539	0.911	0.675
GINEConv	0.881	0.657	0.864	0.636	0.701	0.573	0.692	0.543	0.920	0.681
EdgeConv	0.879	0.646	0.860	0.622	0.692	0.551	0.682	0.532	0.835	0.563
THLM	0.871	0.651	0.871	0.635	0.718	0.587	0.749	0.534	0.911	0.659
Edgeformer	0.882	0.662	0.862	0.643	0.722	0.580	0.744	0.540	0.903	0.663
LLMs										
LLAMA-3-70B	0.832	0.573	0.869	0.587	0.694	0.509	0.643	0.482	0.252	0.471
GPT-4o	0.878	0.609	0.889	0.604	0.712	0.512	0.659	0.503	0.407	0.561
Link2Doc	**0.902**	**0.705**	**0.901**	**0.652**	**0.762**	**0.588**	**0.753**	**0.553**	**0.938**	**0.697**
Link2Doc-NT	–	–	–	–	**0.769**	**0.595**	**0.759**	**0.565**	**0.940**	**0.707**

Table 4.1: The performance comparison of Link Prediction on all datasets (the higher the better), where the bests are highlighted with **bold**, and the second bests are highlighted with underline. Note that − indicates the dataset does not have text on nodes so that Link2Doc-NT cannot be conducted.

aware GNNs: both types of GNNs show comparable performance, with edge-aware GNNs having a slight edge but not consistently outperforming general GNNs in all metrics, which implies co-training language models with GNNs still cannot capture the subtle cues on edge texts. For instance, while Edgeformer achieves the second-best AUC on the Goodreads-Children dataset, it doesn't consistently outperform general GNNs like MaxSage across all datasets. *2) Directly summarizing topology and letting LLMs make predictions may not perform well*: LLMs, such as LLAMA-3-70B and GPT-4o, tend to underperform compared to specialized GNNs and our proposed Link2Doc. It is evident that state-of-the-art LLMs may not be able to fully understand graph topology from the linear topology summarization, highlighting the advantage of the composed document. *3) Text on nodes can further improve the performance:* As can be seen from the table, even though Link2Doc can achieve a generally better performance than other approaches, by considering the text on nodes, Link2Doc-NT can achieve a generally better performance than its no node-text version.

	Goodreads-Children		Goodreads-Crime		Amazon-Apps		Amazon-Movie		StackOverflow	
	MRR	NDCG	MRR	NDCG	MRR	NDCG	MRR	NDCG	MRR	NDCG
General GNN										
MaxSAGE	0.2059	0.3342	0.2130	0.3372	0.2119	0.3938	0.2148	0.4299	0.2256	0.3313
MeanSAGE	0.2156	0.3619	0.2006	0.3199	0.2179	0.3951	0.2433	0.4340	0.2155	0.3351
RevGAT	0.2079	0.3567	0.1921	0.2997	0.2039	0.3865	0.2253	0.4318	0.2159	0.3369
GIN	0.2147	0.4160	0.2354	0.3644	0.2313	0.3486	0.2061	0.4305	0.2254	0.3351
LM-enhanced GNN										
GTN	0.2239	0.4207	0.2536	0.4398	<u>0.3134</u>	0.4296	0.2872	<u>0.4958</u>	0.2321	0.4201
GINEConv	<u>0.2458</u>	<u>0.4399</u>	0.2628	<u>0.4629</u>	0.2916	<u>0.4467</u>	0.2587	0.4472	<u>0.2340</u>	<u>0.4243</u>
EdgeConv	0.2389	0.4281	0.2486	0.4265	0.2871	0.4318	0.2492	0.4432	0.2326	0.4218
THLM	0.1732	0.2998	0.2416	0.3964	0.2337	0.3845	<u>0.2969</u>	0.4284	0.1696	0.3283
Edgeformer	0.1754	0.3000	0.2395	0.3875	0.2239	0.3771	0.2919	0.4344	0.1754	0.3339
LLMs										
LLAMA-3-70B	0.1356	0.2127	0.0692	0.0778	0.0500	0.1692	0.0683	0.1657	0.1421	0.2144
GPT-4o	0.2079	0.4106	<u>0.2684</u>	0.3633	0.2740	0.3697	0.2352	0.4228	0.2299	0.3751
Link2Doc	**0.3167**	**0.5988**	**0.3518**	**0.6115**	**0.4139**	**0.5287**	**0.3926**	**0.6141**	**0.3618**	**0.6359**

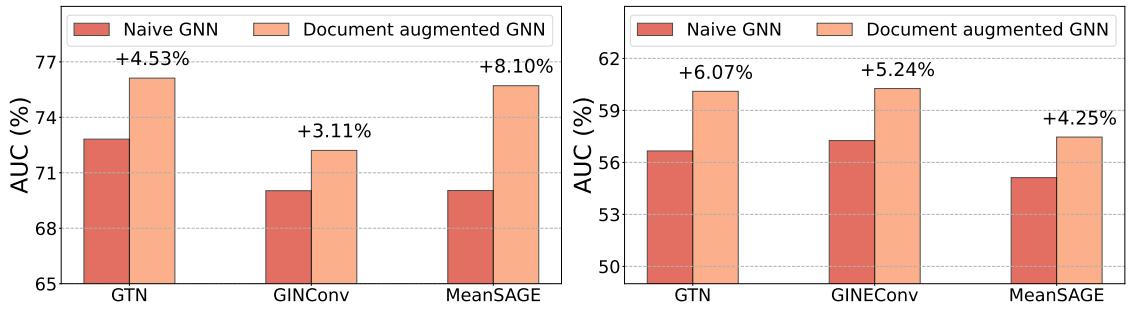
Table 4.2: The performance comparison of Link Prediction on all datasets (the higher the better), where the bests are highlighted with **bold**, and the second bests are highlighted with <u>underline</u>.

	Amazon-APPs	
	AUC	F1
Mean-SAGE	0.551	0.479
GINE	0.573	0.488
GTN	0.567	0.503
Edgeformer	0.612	0.526
Link2Doc	**0.626**	**0.541**

Table 4.3: Comparison on Edge Classification on Amazon-APPs dataset.

4.3.3 Results on Edge Classification

For the edge classification task, we aim to predict the category of each edge based on its associated text and local network structure. There are 5 categories for edges in the Amazon-APPs dataset (i.e., from 1 star to 5 star). The results of the 5-class edge-type classification are shown in Table 4.3. As clearly can be observed in the table, Link2Doc improves the AUC by an average of 5% and the F1 score by 4.2% compared to other models, demonstrating its superior performance in edge classification.



(a) Link Prediction Improvement (b) Edge Classification Improvement

Figure 4.4: Leveraging composed documents to enhance base GNNs on Amazon-APPs dataset.

4.3.4 Ablation Study and Parameter Analysis

We further demonstrate the effectiveness of each component in our framework and analyze the importance of different hyper-parameter settings.

Performance Elevation from Transition Document. We first aim to check the general performance elevation brought by the composed transition document $d_{(s,t)}$. We adopt three GNNs using BERT to obtain embeddings on edges and illustrate whether using the composed $d_{(s,t)}$ as a reference would improve their performance in both link prediction and edge classification tasks. The results on the Amazon-APPs dataset are presented in Figure 4.4. As can be seen from the figure, the document-augmented GNNs excel in their general version with an average improvement of 5%. By summarizing all relations from s to t as a coherent document, language models can give positive feedback on the proposed self-supervised learning module to guide different GNNs in learning.

Hyperparameter Analysis. We then aim to investigate the sensitivity of the key hyperparameter λ_2 and their impact on Link2Doc's performance. Specifically, since λ_1 controls the basic objective function for link prediction, we fix $\lambda_1 = 1$ and show the link prediction performance on the Amazon-APPs dataset under different λ_2

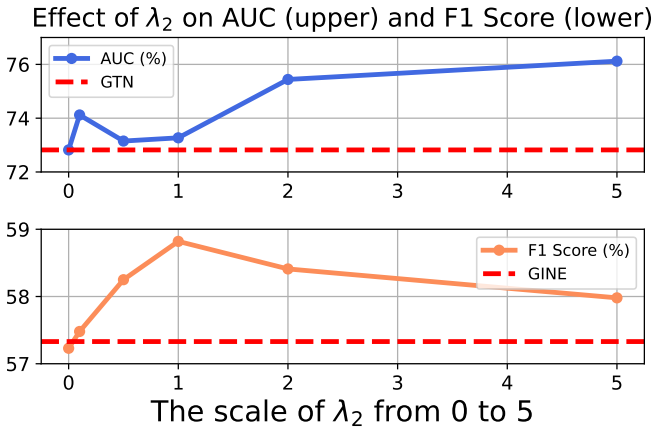


Figure 4.5: The performance on Amazon-APPs.

	Children	Crime
Inference Time (s)		
EDGEFORMER	1450.67	3307.66
LINK2DOC	49.35	23
Training Time (h)		
EDGEFORMER	12.17	12.65
LINK2DOC	5.253	7.04

Table 4.4: Comparison of inference and training time on Goodreads-Children and Goodreads-Crime.

values (ranging from 0 to 5). As shown in Figure 4.5, both metrics show consistent results across varying parameter values. By comparing with the second-best methods (highlighted with red dash horizontal lines), LINK2DOC with various λ_2 values can achieve overall better results. This demonstrates that our model maintains superior performance across different configurations, highlighting its stability and effectiveness.

4.3.5 Runtime Analysis

We further illustrate the runtime comparison between our proposed LINK2DOC with the state-of-the-art competitor - Edgeformer. The comparison table (Table 4.4) highlights the efficiency of our method, Link2Doc, which significantly outperforms Edgeformer in both inference and training times. For the Goodreads-Children dataset, Link2Doc is approximately 29 times faster in inference and more than twice as fast in

training. On the Goodreads-Crime dataset, Link2Doc demonstrates an even greater advantage, being about 144 times faster in inference and almost twice as fast in training. These improvements stem from Link2Doc's design, which does not require fine-tuning language models; instead, it builds document representations and uses a pre-trained GNN, avoiding the extensive matrix calculations in Edgeformer's cross-attention design. Consequently, Link2Doc is more scalable and efficient for large-scale link prediction tasks.

4.4 Conclusion

In this work, we study the problem of link prediction on textual-edge graphs, where existing GNN-based and LLM-based methods may fall short of jointly capturing both semantic and topology information to make more accurate link predictions. We present a novel framework LINK2DOC that learns and aligns the semantic representation and topology representation by 1) building a structured document to preserve both topology and semantic information; 2) proposing a Transition Graph Neural Network module for better learning representations of the transition graph; and 3) designing a self-supervised learning module to let TGNN have text understanding ability like LLMs. Our method generally outperforms other approaches from multiple aspects on five datasets.

Chapter 5

Conclusion and Future Works

This dissertation explores the synergy between language modeling and graph data mining in their respective data mining problems. I have specifically focused on three tasks: *1) enhancing graph data mining by exploiting semantic information on networks*, *2) integrating structured knowledge and quantifying uncertainty in natural language understanding*, and *3) representation learning on textual-edge graphs for link prediction.*

Completed Work: Enhancing graph data mining by exploiting semantic information on networks. We first develop a generative framework to generate heterogeneous graphs. The proposed framework - *HGEN* learns the latent distribution of observed heterogeneous graphs and generates novel ones by preserving various heterogeneous graph properties. Next, for the task of information diffusion source localization, we have proposed a generic framework *SL-VAE* for probabilistic source localization in graph diffusion. To handle more complex network structures, we extend *SL-VAE* to handle cross-network information source localization: the proposed framework *CNSL* further incorporates different features on nodes and develops a novel objective function such that the reconstruction of diffusion sources in the source network is fully aware of and guided by heterogeneous diffusion patterns in

cross-networks. Finally, for the task of influence maximization, we developed a novel framework *DeepIM* generatively learns the latent representation of seed nodes and adapts to various node-centrality constraints, offering a flexible, data-driven solution to the IM problem.

Completed Work: Integrating structured knowledge and quantifying uncertainty in natural language understanding. We present the external KnowlEdge-Enhanced Prompting method (KEEP) to achieve open-ended commonsense reasoning without pre-defining an answer candidate set and an answer scope. KEEP leverages an external knowledge base (e.g., ConceptNet) as the answer searching space and iteratively extracts multi-hop reasoning paths relevant to the question by Pretrained Language Models. In addition, we study the complexities of predictive uncertainty of using in-context learning in large language models. This research delves into different sources of uncertainty in LLMs, namely the aleatoric uncertainty from provided demonstrations, and the epistemic uncertainty from the language model's configurations. We provide a novel formulation and an estimation method to quantify both uncertainties, offering an unsupervised and plug-and-play approach to understand and mitigate the uncertainty when using in-context learning in using LLMs.

Completed Work: Representation Learning of Textual-edge Graphs for Link Prediction. Textual-edge Graphs (TEGs), characterized by rich text annotations on edges, are increasingly significant in network science due to their ability to capture rich contextual information among entities. In this paper, we present a novel framework - LINK2DOC, designed especially for link prediction on TEGs. Specifically, we propose to summarize neighborhood information between node pairs as a human-written document to preserve both semantic and topology information. We also present a specialized GNN framework to process the multi-scaled interaction between target nodes in a stratified manner. Finally, a self-supervised learning model is

Task	Description	Status
Research Area A	*Enhancing graph data mining by exploiting semantic information on networks*	
A1	Deep generation of heterogeneous network: problem formulation & proposed framework	Completed
A2	Cross-network source localization: problem formulation, framework, and dataset construction	Completed
A3	Deep influence maximization: problem formulation & proposed framework	Completed
A4	Experiment validation on synthetic and real-world datasets	Completed
Research Area B	*Integrating structured knowledge and quantifying uncertainty in natural language understanding*	
B1	The definition and formulation of the problem	Completed
B2	Using language models to guide reasoning path retrieval	Completed
B3	Estimating the uncertainty associated with in-context learning	Completed
B4	Experiment validation on synthetic and real-world datasets	Completed
Research Area C	*Representation Learning of Textual-edge Graphs for Link Prediction*	
C1	The proposal of unique challenges of link prediction in TEGs	Completed
C2	The design of Node-centric paragraph composition for transition graphs	Completed
C3	The design and implementation of LLM-enhanced GNN framework	Completed
C4	Experiment validation on synthetic and real-world datasets	Completed
D	Thesis Revision	Completed

Table 5.1: Research tasks and status

utilized to enhance the GNN's text-understanding ability from language models. Empirical evaluations, including link prediction, edge classification, parameter analysis, runtime comparison, and ablation studies, on five real-world datasets demonstrate that LINK2DOC achieves generally better performance against existing edge-aware GNNs and language models in link predictions.

5.1 Research Task

The major research tasks are described as follows, and the current status of these tasks is listed in Table 5.1.

5.1.1 Enhancing graph data mining by exploiting semantic information on networks

- **Deep generation of heterogeneous network: problem formulation & proposed framework (A1).** I propose a novel framework - HGEN for heterogeneous graph generation, which can jointly capture the semantic, structural, and global distributions of heterogeneous graphs. Our framework consists of a

novel heterogeneous walk generator that can hierarchically generate meta-path instances (namely heterogeneous walk) and a heterogeneous graph assembler that can construct new graphs by sampling from the generated heterogeneous walks in a stratified manner.

- **Cross-network source localization: problem formulation, framework, and dataset construction (A2).** I formulate the unique Cross-network Source Localization problem, which aims at characterizing the distribution of diffusion sources given the graph topology, graph features, and final diffused observation. The proposed framework captures 1) both static and dynamic node features, and 2) the heterogeneous diffusion patterns of both networks. The approximation of diffusion sources is fully aware of various node features and the interplay of cross-network information diffusion patterns.

- **Deep influence maximization: problem formulation & proposed framework (A3).** I propose a novel framework to tackle the IM problem in a more robust and generalized way. Particularly, to characterize the complex nature of the seed set, we propose to character the probability of the seed set and directly search for a more optimal seed set in continuous space. Furthermore, to solve the challenge of modeling the underlying diffusion pattern, we offer two different learning-based diffusion models to characterize the diversified diffusion dynamics with efficiency and efficacy guarantee. Finally, we propose a novel objective function that can be coupled with multiple constraints for seed node set inference, which can adapt to different IM application schemes.

- **The experiment verification on both sets of datasets (A4).** I conduct experiments against state-of-the-art methods designed originally for different research tasks. Results show substantially improved performance of the proposed methods.

5.1.2 Integrating structured knowledge and quantifying uncertainty in natural language understanding

- **The definition and formulation of the novel problem (B1).** The open-ended commonsense reasoning problem is formulated as a multi-hop reasoning task iteratively conducted on an external knowledge graph, where no pre-defined answers are provided. The predictive uncertainty of LLMs associated with in-context learning is also formulated as two types of uncertainties from different sources.

- **The model design of the knowledge-augmented commonsense reasoning (B2).** I leverage the implicit knowledge stored in pretrained language models to guide the overall searching/reasoning process under both zero-shot and finetuning settings. Then, I utilize the retrieved reasoning paths as additional explanations to justify the answer choice.

- **Estimating the uncertainty associated with in-context learning (B3).** I propose a novel formulation and corresponding estimation method to quantify both types of uncertainties. The proposed method offers an unsupervised way to understand the prediction of in-context learning in a plug-and-play fashion.

- **Experiment validation on synthetic and real-world datasets (B4).** I empirically demonstrate the performance of our method against the state-of-the-art competitors, which excels other comparison methods in multiple metrics under different settings.

5.1.3 Textual-edge Graph Representation Learning for Link Prediction

- **The proposal of unique challenges of link prediction on TEGs (C1).** I formulate the unique problem of leveraging language models in link prediction on TEGs since GNNs often fall short of fully capturing the contextualized semantics on edges and graph topology.

- **The design design of node-centric paragraph composition for transition graphs (C2).** I propose a novel method to summarize graph connections in plain language. The coherent document composition summarizes local topology information between node pairs in plain language.

- **The design and implementation of LLM-enhanced GNN framework (C3).** I propose to use an integrated framework to jointly consider topology and semantic information in TEGs, which consists of a specially designed Transition Graph Neural Network (TGNN) and a self-supervised learning module to teach TGNN language processing ability.

- **Experiment validation on synthetic and real-world datasets (C4).** I empirically compare our method against existing state-of-the-arts in five real-world datasets. The results reveal the proposed method can elevate the performance of general GNNs and achieve competitive performance against edge-aware GNNs.

5.2 Published Works

5.2.1 Publication during PhD Study

Conference Publications

17. [**NeurIPS 2024**] Guangji Bai, Yijiang Li, **Chen Ling**, Kibaek Kim, and Liang Zhao. SparseLLM: Towards Global Pruning of Pre-trained Language Model. *Thirty-eighth Annual Conference on Neural Information Processing Systems.* Vancouver, Canada, 2024. [12]

16. [**NeurIPS 2024**] Zhuofeng Li, Zixing Gou, Xiangnan Zhang, Zhongyuan Liu, Sirui Li, Yuntong Hu, **Chen Ling**, Zheng Zhang, and Liang Zhao. TEG-DB: A Comprehensive Dataset and Benchmark of Textual-Edge Graphs. *Thirty-eighth Annual Conference on Neural Information Processing Systems.* Vancouver, Canada, 2024. [98]

15. [**ICDM 2024**] Mingchen Li, **Chen Ling**, Rui Zhang and Liang Zhao. A Condensed Transition Graph Framework for Zero-shot Link Prediction with Large Language Models. *The 24th IEEE International Conference on Data Mining.* Abu Dhabi, UAE, 2024. [94]

14. [**KDD 2024**] **Chen Ling**, Tanmoy Chowdhury, Jie Ji, Sirui Li, Andreas Züfle, Liang Zhao. Source Localization for Cross Network Information Diffusion. *The 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* Barcelona, Spain, 2024. [117]

13. [**ACL 2024**] Yifei Zhang, Bo Pan, **Chen Ling**, Yuntong Hu, Liang Zhao. ELAD: Explanation-Guided Large Language Models Active Distillation. *The 62nd Annual Meeting of the Association for Computational Linguistics.* Bangkok, Thailand, 2024. [210]

12. [**NAACL 2024**] **Chen Ling**, Xujiang Zhao, Wei Cheng, Yanchi Liu, Yiyou Sun, Xuchao Zhang, Takao Osaki, Katsushi Matsuda, Liang Zhao, Haifeng Chen. Uncertainty Decomposition and Quantification for In-Context Learning of Large Language Models. *2024 Annual Conference of the North American*

Chapter of the Association for Computational Linguistics. Mexico City, 2024. [118]

11. [**AISTATS 2024**] Nguyen Do, Tanmoy Chowdhury, **Chen Ling**, Liang Zhao, My T. Thai. MIM-Reasoner: Learning with Theoretical Guarantees for Multiplex Influence Maximization. *The 27th International Conference on Artificial Intelligence and Statistics.* Valencia, Spain, 2024. [40]

10. [**SDM 2024**] Junruo Gao, **Chen Ling**, Carl Yang, Liang Zhao. Helper Recommendation with Seniority Control in Online Health Community. *2024 SIAM International Conference on Data Mining.* Houston, TX, 2023. [50]

9. [**EMNLP 2023**] **Chen Ling**, Xuchao Zhang, Xujiang Zhao, Yanchi Liu, Wei Cheng, Takao Osaki, Haifeng Chen, Liang Zhao. Open-ended Commonsense Reasoning with Unrestricted Answer Candidates. *The 2023 Conference on Empirical Methods in Natural Language Processing.* Singapore, 2023. [113]

8. [**ICML 2023**] **Chen Ling**, Junji Jiang, Junxiang Wang, My Thai, Lukas Xue, James Song, Meikang Qiu, Liang Zhao. Deep Graph Representation Learning and Optimization for Influence Maximization. *Fortieth International Conference on Machine Learning.* Hawaii, 2023. [111]

7. [**ICLR 2023**] **Chen Ling***, Guangji Bai*, Liang Zhao. Temporal Domain Generalization with Drift-Aware Dynamic Neural Networks. *The Eleventh International Conference on Learning Representations.* Kigali, Rwanda, 2023. [**Oral Presentation**: Top-5% among all accepted papers.] [9]

6. [**SDM 2023**] Guangji Bai, **Chen Ling**, Yuyang Gao, Liang Zhao. Saliency-Augmented Memory Completion for Continual Learning. *2023 SIAM International Conference on Data Mining.* Minneapolis, MN, 2023. [10]

5. [**ICDM 2022**] **Chen Ling**, Tanmoy Chowdhury, Junji Jiang, Junxiang Wang, Xuchao Zhang, Haifeng Chen, and Liang Zhao. DeepGAR: Deep Graph Learning for Analogical Reasoning. *The 22nd IEEE International Conference on Data Mining.*. Orlando, FL, 2022. [105]

4. [**ECML-PKDD 2022**] **Chen Ling**, Henning Cao, and Liang Zhao. STGEN: Deep Continuous-time Spatiotemporal Graph Generation. *The 2022 European Conference on Machine Learning and Principles Discovery in Databases.*. Grenoble, France, 2022. [109]

3. [**KDD 2022**] **Chen Ling**, Junji Jiang, Junxiang Wang, and Liang Zhao. SL-VAE: Variational Autoencoder for Source Localization in Graph Information Diffusion. *The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* Washington, D.C., 2022. [110]

2. [**ICDM 2021**] **Chen Ling**, Carl Yang, and Liang Zhao. Deep Generation of Heterogeneous Networks. *The 21st IEEE International Conference on Data Mining.* Online, 2021. [**Best Paper Candidate**] [108]

1. [**WWW 2021**] Liming Zhang, Liang Zhao, Shan Qin, Dieter Pfoser, and **Chen Ling**. TG-GAN: Continuous-time Temporal Graph Deep Generative Models with Time-Validity Constraints. *The 30th International World Wide Web Conference.* Online, 2021. [205]

Journal Publications

4. [**Neural Network**] Tanmoy Chowdhury, **Chen Ling**, Junji Jiang, Junxiang Wang, My T. Thai, Liang Zhao. Deep graph representation learning influence maximization with accelerated inference. *Neural Network*, Volume 180, 2024, 106649. [32]

3. [**FBD**] Guanchen Wu, **Chen Ling**, Ilana Graetz, Liang Zhao. Ontology Extension by Online Clustering With Large Language Model Agents. *Frontiers in Big Data*, 2024, Accepted. [177]

2. [**FBD**] Junji Jiang, **Chen Ling**, Hongyi Li, Guangji Bai, Xujiang Zhao, Liang Zhao. Quantifying Uncertainty in Graph Neural Network Explanations. *Frontiers in Big Data 7* (2024): 1392662. [71]

1. [**KAIS**] **Chen Ling**, Carl Yang, Liang Zhao. Motif-guided Heterogeneous Graph Deep Generation. *Knowledge and Information Systems*, 65.7 (2023): 3099-3124. [112]

5.2.2 Publication before PhD Study

2. [**ICME 2021**] **Chen Ling**, Di Cui, Guangmo Tong, and Jianmin Zhu. On Forecasting Dynamics in Online Discussion Forums. *The 21st IEEE Multimedia and Expo.* Online, 2021.

1. [**HT 2020**] **Chen Ling**, Mozi Chen, and Guangmo Tong. NesTPP: Modeling Information Diffusion in Online Discussion Forum. *The 31st ACM Hypertext.* Online, 2020.

Appendix A

Enhancing Graph Data Mining by Exploiting Semantic Information on Networks

A.1 CNSL Technical Supplements

A.1.1 Derivation of Eq. (2.4)

$\log p_\psi(y_t|x_s, G_s, G_t) = \log[\sum_{x_t} p_{\psi_1}(x_t|x_s, G_s) \cdot p_{\psi_2}(y_t|x_t, G_t)]$, where x_t inherited infection probability from y_s. In practice, we assume $p_{\psi_1}(x_t|x_s, G_s)$ follows delta distribution, where only the x_t is 1 that corresponds to the x_s and the rest of x_t's are 0. This property is also assumed in many works [141] using VAE. Therefore, $\log p_\psi(y_t|x_s, G_s, G_t)$ is simplified as Eq. (2.4).

A.1.2 Graphical Model of CNSL

We provide the graphical model for the CNSL framework in Figure A.1. As shown in the figure, solid arrows indicate the variational approximation $q_{\phi_1}(z_s|x_s, G_s)$ and

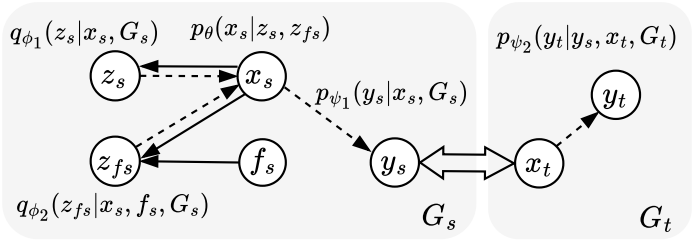


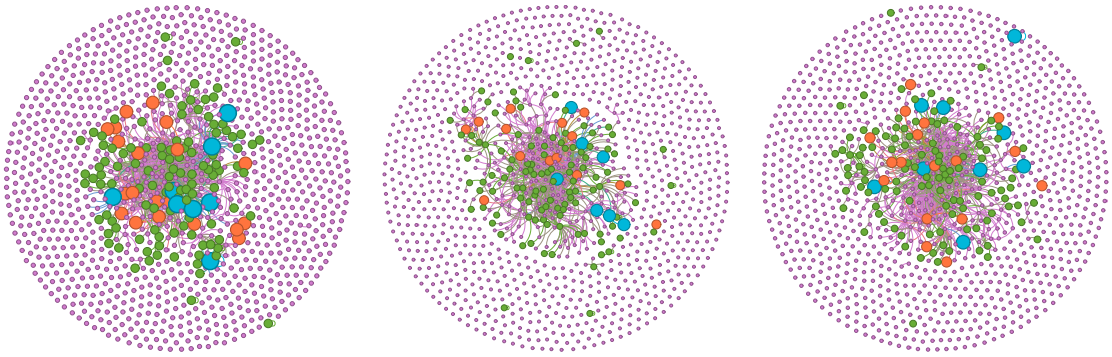
Figure A.1: The graphical model for CNSL, where the solid arrows indicate the variational approximation $q_{\phi_1}(z_s|x_s, G_s)$ and $q_{\phi_2}(z_{fs}|x_s, f_s, G_s)$ to the intractable posterior $p(Z|x_s, f_s, G_s)$. Dashed arrows denote the generative process that decodes x_s from $p_\theta(x_s|Z)$ and predicts the information diffusion $p_\psi(y_t|x_s, \mathcal{G})$.

$q_{\phi_2}(z_{fs}|x_s, f_s, G_s)$ to the intractable posterior $p(Z|x_s, f_s, G_s)$. Dashed arrows denote the generative process that decodes x_s from $p_\theta(x_s|Z)$ and predicts the information diffusion $p_\psi(y_t|x_s, \mathcal{G})$. The two directional arrows between y_s and x_t indicate x_t inherits the infection probability from the diffusion observation y_s through bridging nodes L.

A.2 Experiment Supplement

A.2.1 Case Study

In a case study depicted in Figure A.2, we illustrate the distribution of selected seed nodes. Here violet nodes represent the nodes that are not seeds. On the other hand, green nodes are the original seeds that were not selected by CNSL; orange nodes are wrongly identified as seeds by CNSL; and the Blue color nodes are correctly identified as seeds by CNSL.



(a) LT2LT (b) LT2IC (c) LT2SIS

Figure A.2: Visualization of the inferred Seed Nodes by CNSL.

A.2.2 Algorithm

Algorithm 2: CNSL Training Framework

Input : G_s, G_t, f_s, L, x_s, y_t

Output: Trained $q_\phi(\cdot)$, $p_\theta(\cdot)$, and $p_\psi(\cdot)$

repeat
  repeat
    $z_s = q_{\phi_1}(x_s, G_s)$;
    $z_{fs} = q_{\phi_2}(x_s, x_{fs}, G_s)$;
    $\hat{x}_s = p_\theta(z_s, z_{fs})$;
    $\hat{y}_s = p_{\psi_1}(\hat{x}_s, G_s)$;
    $\hat{x}_t \leftarrow \hat{y}_s$ based on the bridge links: $L = \{(v_s, v_t)|v_s \in V_s, v_t \in V_t\}$;
    $\hat{y}_t = p_{\psi_2}(\hat{x}_t, G_t)$;
    Calculate $\mathcal{L}_{\text{train}}$;
    Back-propagate loss and Update parameters in $q_\phi(\cdot)$, $p_\theta(\cdot)$, and $p_\psi(\cdot)$;
  until *each batch in train_set*;
until *epoch in* 1 *to num_epochs*;

We summarize the inference algorithm of CNSL in 2. We want to use observed x_s and y_t to learn the approximate posterior $q_\phi(Z|x_s, \mathcal{G})$, the decoding function $p_\theta(x_s|Z)$, and the cross-network diffusion prediction function $p_\psi(y_t|x_s, \mathcal{G})$. Specifically, we sep-

arately obtain two latent variables z_s and z_{fs} in Line 3-4. Both z_s and z_{fs} are fed to reconstruct $\hat{x}_s$ in Line 5. After the seed set reconstruction, we conduct cross-network diffusion prediction as shown in Line 6-8. The backpropagation is calculated based on Eq. (2.5) that consists of seed nodes reconstruction error, diffusion estimation error, as well as constraints of KL divergence and influence monotonicity.

A.2.3 Algorithm

Algorithm 3: CNSL Inference Framework

Input : $p_\theta(x_s|z_s, z_{fs})$; $p_{\psi_1}(y_s|x_s, G_s)$; $p_{\psi_2}(y_t|y_s, x_t, G_t)$; the number of iteration η; learning rate α.

Output: $\hat{x}_s$

1 $\bar{z}_s = \frac{1}{k}\sum_i^k q_{\phi_1}(z_s|\hat{x}_s^{(i)}, \mathcal{G})$;

2 $\bar{z}_{fs} = \frac{1}{k}\sum_i^k q_{\phi_2}(z_s|\hat{x}_s^{(i)}, \mathcal{G})$;

3 **repeat**

4 $\quad \hat{x}_s = p_\theta(\bar{z}_s, \bar{z}_{fs})$;

5 $\quad \hat{y}_s = p_{\psi_1}(\hat{x}_s, G_s)$;

6 $\quad \hat{x}_t \leftarrow \hat{y}_s$ based on $L = \{(v_s, v_t)|v_s \in V_s, v_t \in V_t\}$;

7 $\quad \hat{y}_t = p_{\psi_2}(\hat{x}_t, G_t)$;

8 $\quad \bar{z}_s \leftarrow \bar{z}_s - \alpha \cdot \nabla \mathcal{L}_{\text{pred}}(\hat{y}_t, \bar{z}_s, \bar{z}_{fs})$

9 **until** $i = 0, ..., \eta$;

10 $\hat{x}_s = p_\theta(\bar{z}_s, \bar{z}_{fs})$

We summarize the inference algorithm of CNSL in 3. For the seed set inference, we first sample k different $\hat{x}_s^{(i)}$ from the training set, and we marginalize them to obtain two latent variables $\bar{z}_s$ and $\bar{z}_{fs}$ (Line 1-2). For η iterations, we decode the predicted $\hat{x}_s$ based on $(\bar{z}_s, \bar{z}_{fs})$ (Line 4) and conduct cross-network information diffusion prediction (Line 5-7). The error between predicted $\hat{y}_t$ and the observed y_t is leveraged to update $\bar{z}_s$ based on Eq. (2.16).

A.3 DeepIM Technical Supplements

A.3.1 Proof of Theorem 2.3.1 and Corollary 2.3.2

The proof of Theorem 2.3.1 is demonstrated as follows.

Proof. $g_u(x, G; \theta) = \mathcal{A}^1 \circ (C^1 \circ \mathcal{A}^2 \circ C^2 \cdots \circ \mathcal{A}^K \circ C^K)$ via iterating Eq. (2.9) recursively. Because $\mathcal{A}^k$ and C^k are non-decreasing, so is $\mathcal{A}^1 \circ C^1 \cdots \circ \mathcal{A}^K \circ C^K$, which is g_u. Therefore, M is infection monotonic. Because g_u and g_r are non-decreasing, M is also non-decreasing and hence score monotonic. □

The proof of Corollary 2.3.2 is demonstrated as follows.

Proof. For the GAT model, $a^k_{i,j} = \mathcal{A}^k(h^k_i, h^k_j, \theta^k) = \theta^k_1(\theta^k_2 h^{k-1}_i || \theta^k_2 h^{k-1}_j)$ and $h^k_i = C^k(a^k, \theta^k) = \max(\sum_{j \in N(i)} softmax(LeakyReLU(a^k_{i,j}))\theta^k_1 h^{k-1}_j, 0)$ where $\theta^k = [\theta^k_1; \theta^k_2]$, and $||$ denotes the concatenation of two vectors. $\mathcal{A}^k$ are non-decreasing and non-negative because $\theta^k \geq 0$ (i.e. $\theta^k_1 \geq 0$ and $\theta^k_2 \geq 0$). In other words, $a^k_{i,j}$ is non-negative, and h^k_i, θ^k_1 and the softmax operator are non-negative. Therefore, the LeakyReLU operator and $\max(\bullet, 0)$ can be removed from C^k. That is, $h^k_i = C^k(a^k, \theta^k) = \sum_{j \in N(i)} softmax(a^k_{i,j})\theta^k_1 h^{k-1}_j$. Because the softmax operator is non-decreasing, and θ^k_1 is non-negative. C^k is non-decreasing. Hence, The GAT model satisfies the conditions in Theorem 2.3.1 and thus M is score and infection monotonic. □

A.3.2 Derivation of Equation 2.16

The derivation of Equation 2.16 is shown as follows.

$$\mathcal{L}_{\text{pred}} = \min_z \mathbb{E}\big[-\log p_\theta(y|x, G) - \log p_\psi(x|z) \big], \text{ s.t.} \sum\nolimits_{i=0}^{|V|} \mathcal{F}(v_i, G) \cdot x_i \leq k,$$

Since we assume the optimal $\tilde{y} = |V|$ and the predicted $y \in [0, |V|]$, the optimization target is to maximize the y until it reaches the fully infected status. Therefore, the first term in Eq. (2.16) is written as the Mean Squared Loss (MSE): $\|\tilde{y} - M(x, G; \theta)\|_2^2$. For the second term in Eq. (2.16), the value range of x after the autoencoder is $[0, 1]$, indicating the probability of each node being selected to the seed set. $x \in [0, 1]$ fits the binomial distribution so that minimizing the negative log-likelihood is equivalent to minimizing the probability mass function. Therefore, the second term of the above function can be written to $-\log\big[\prod_i^{|V|} f_\psi(z_i)^{x_i}(1 - f_\psi(z_i)^{1-x_i}\big]$. Adding both terms gives us the final expression as shown in Eq. (2.16):

$$\mathcal{L}_{\text{pred}} = \max_z \mathbb{E}\big[p_\theta(y|x, G) \cdot p_\psi(x|z)\big], \text{ s.t.} \sum\nolimits_{i=0}^{|V|} \mathcal{F}(v_i, G) \leq k.$$

A.3.3 More Experiments

Data. We provide a more detailed dataset description as follows. *1) Jazz* [143]. This dataset is a Jazz musicians collaboration network, where each node represents a musician and each edge represents two musicians who have played together in a band. *2) Cora-ML* [127]. This network contains computer science research papers, where each node represents a paper and each edge represents one paper cites the other one. *3) Power Grid* [143]. This is a topology network of the US Western States Power Grid. An edge represents a power supply line. A node is either a generator, a transformation, or a substation. *4) Network Science* [143]. This is a coauthorship network between scientists working on network theory, where nodes represent scientists and edges represent two scientists who have collaborated. *5) Digg* [135]. A directed network of social media where users follow each other and a vote to a post allows followers to see the post. *6) Weibo* [135]. A directed follower network where a cascade is defined by the first tweet and the list of retweets.

Hyperparameter Setting. For each baseline, we set hyperparameters according to their original papers and fine-tune them on each dataset. For the configuration of each diffusion model, we use a weighted cascade version of the IC model, i.e., the propagation probability $p_{u,v} = 1/d_v^{in}$ (d_v^{in} denotes the in-degree of node v) for each edge $e = (u, v)$ on graph G; For LT model, the threshold θ is set to be uniformly sampled from $[0.3, 0.6]$ for each node v; the infection probability and recovery probability are set to be 0.001 in the SIS model. For DeepIM, the 2-layer GAT-structured diffusion estimation model that each layer contains 4 attention heads and the dimension of each attention channel is 64. Both encoder and decoder are symmetric 4-layer MLP with hidden size 512, 1024, 1024 and 1024 for each layer, respectively. We choose Adam with learning rate 0.001 and 0.0001 for optimizing both Eq. (2.14) and Eq. (2.15), respectively.

Appendix B

Integrating Structured Knowledge and Quantifying Uncertainty in Natural Language Understanding

B.1 Open-ended Commonsense Reasoning

B.1.1 Experimental Details

Language Models. Our method is implemented with PyTorch and the Huggingface library. We do not train any new language models but finetune existing ones with the training procedure described in Section 3.1.2. the language model p_θ can be any language model either with the zero-shot setting or finetuned on the external knowledge base, and we leverage the masked language model RoBERTa-large [120] since it has larger representative power in commonsense ability with a less model size [214]. Specifically for GPT-3, we used the OpenAI API and specifically chose TEXT-DAVINCI-003 as the base model.

Commonsense Questions	Answer 1	Answer 2	Answer 3	Answer 4	Answer 5	Top 1	Top 3	Top 5
August needed money because he was afraid that he'd be kicked out of his house. What did he need money to do?	**needed for survival in urban center**	usual medium used to buy things	can buy things	gregorian calendar	root of all evil	1	1	1
The weasel was becoming a problem, it kept getting into the chicken eggs kept in the what?	found in grocery store	**hen house**	bird	roomful of junkies	farm	0	1	1
Where can you put a picture frame when it's not hung vertically?	picture frame	electro magnetic ibt	bounded surface	**table**	useful to convey idea	0	0	1
Unlike a spider and his many sight seers, people only have what?	heavier than sandwitches	go to mexican restaurants for dinner	optimistic dreams	find sound of bells mournful	watch movies at home on dvds	0	0	0

Table B.1: Examples of the rating criteria for assessing the model performance.

Knowledge Graph. We leverage ConceptNet, a general-domain knowledge graph, as our structured knowledge source G, which contains $799,273$ nodes and $2,487,810$ edges. We obtain the data from the repository[1] with version 5.6.0. ConceptNet contains 34 relations (edge types). In terms of achieving less noise and better inference time, we pre-process the ConceptNet by 1) merging similar relations into one unified relation to reduce ambiguity; 2) extracting English-only content and transforming all relations into an adjacency edge list; and 3) translating the relational edge between two concepts to a natural language by designed template: (related to, car, traffic) $\rightarrow$ "Car is related to traffic." An example of the transformation template can be found in Table B.2.

Data. *1) CommonsenseQA (CSQA)*: [156] is a multiple-choice QA dataset about common-world scenarios, which is constructed on ConceptNet and contains $1,140$ test cases. *2) QASC*: Khot et al. [83] is a multiple-choice QA dataset about grad-school science, which contains 917 test cases in total. We discard the provided answers and supporting arguments in both datasets.

Relation Groups	Merged Relations	Relation Text
antonym/distinctfrom	antonym	is the antonym of
atlocation/locatednear	atlocation	is at location of
causes/causesdesire/motivatedby	causes	causes
relatedto/similarto/synonym	relatedto	is related to
isa/instanceof/definedas	isa	is a

Table B.2: Examples of the ConceptNet edge relation transformation templates.

[1] https://github.com/commonsense/conceptnet5/wiki/

Implementation Details. Inferences are conducted on Nvidia Quadro RTX 6000 with approximately 100 GPU hours. We set the maximum length of the reasoning path to be 3, indicating our algorithm only searches for answers within 3-hop of neighbors from all the entities in the question. We generate $20,000$ logical sentences as the training corpus for each dataset as described in Section 3.1.2. We finetuned our model with 2 epochs and $1e-5$ learning rate by leveraging the training corpus.

Human Evaluation Criteria. We work with human annotators (students recruited from the college) to obtain the Top-N accuracy and the validity rate of the generated reasoning paths in Table 3.1 and Figure 3.6. There are three annotators in total to evaluate the generated answers. To be more specific, for each dataset, two annotators are individually assigned to score the Top-N accuracy. If there are discrepancies between two annotators' judgments, the third annotator is involved in making the final decision.

We provide guidance for evaluating each model's performance in Table B.1, and the Evaluation Criteria for the results in Figure 3.6 are self-contained. Specifically, we sample answers five times for each model on each commonsense question and rank their answers based on their model's confidence score (i.e., Equation (3.2)). Human annotators are responsible for evaluating whether there exists an answer that fits the semantic meaning of the question in the Top-1, Top-3, and Top-5 candidates, respectively.

B.1.2 More Test Cases

We illustrate more test cases of each model's performance on both datasets. (CSQA: Table B.3; QASC: Table B.4).

B.2 Uncertainty Quantification of In-context Learning for Large Language Model

B.2.1 Variance-based Decomposition

Alternatively, we can use the variance as a measure of uncertainty. Let $\sigma^2(\cdot)$ compute the variance of a probability distribution, and the total uncertainty is then $\sigma^2(\mathbf{y}_T|\mathbf{x}_{1:T})$. This quantity can then be decomposed using the law of total variance:

$$\sigma^2(\mathbf{y}_T|\mathbf{x}_{1:T}) = \sigma^2_{q(\Theta)}\left(\mathbb{E}[\mathbf{y}_T|\mathbf{x}_{1:T},\Theta]\right) + \mathbb{E}_{q(\Theta)}\left[\sigma^2(\mathbf{y}_T|\mathbf{x}_{1:T},\Theta)\right]. \tag{B.1}$$

where $\mathbb{E}[\mathbf{y}_T|\mathbf{x}_{1:T},\Theta]$ and $\sigma^2(\mathbf{y}_T|\mathbf{x}_{1:T},\Theta)$ are mean and variance of $\mathbf{y}_T$ given $p\left(\mathbf{y}_T|\mathbf{x}_{1:T},\Theta\right)$. $\sigma^2_{q(\Theta)}\left(\mathbb{E}[\mathbf{y}_T|\mathbf{x}_{1:T},\Theta]\right)$ represents the variance of $\mathbb{E}[\mathbf{y}_T|\mathbf{x}_{1:T},\Theta]$ when $\Theta \sim q(\Theta)$, which indicates the epistemic uncertainty since it ignores the contribution of z. In contrast, $\mathbb{E}_{q(\Theta)}\left[\sigma^2(\mathbf{y}_T|\mathbf{x}_{1:T},\Theta)\right]$ in Eq. (B.1) represents the aleatoric uncertainty since it denotes the average value of $\sigma^2(\mathbf{y}_T|\mathbf{x}_{1:T},\Theta)$ with $\Theta \sim p(\Theta)$ and ingores the contribution of Θ to $\mathbf{y}_T$. Note that variance-based uncertainty decomposition does not involve the probability of the generated tokens, which is applicable to black-box LLMs (e.g., GPT models).

Variance Approximation. In practice, when we are dealing with black-box LLMs (e.g., ChatGPT), there are multiple hyperparameters (e.g., `temperature` and `top_p`) allowing to return different responses. Specifically, $[\mathbf{y}_T^1,\ldots,\mathbf{y}_T^L]$ can be obtained through querying the LLM with different demonstrations $[\mathbf{x}_{1:T-1}^1,\ldots,\mathbf{x}_{1:T-1}^L]$ L times. The different sets of parameter configurations are denoted as $[\Theta_1,\ldots,\Theta_M]$. The $\mathbb{E}[\mathbf{y}_T|\mathbf{x}_{1:T},\Theta]$ can then be calculated as the expected model output given the input data and the model parameters Θ. Calculating the variance of this expectation with respect to a set of model configurations over all sets of demonstrations gives the epis-

temic uncertainty. The variance $\sigma^2(\mathbf{y}_T)$ can also be obtained given a set of few-shot demonstrations over all model parameters. Finally, average this variance over the certain model configuration to obtain the aleatoric uncertainty.

B.2.2 Dataset Description

Sentiment Analysis. 1) EMOTION [145] contains $2,000$ test cases, where LLMs are asked to classify a given sentence with six categories: *sadness*, *joy*, *love*, *anger*, *fear*, *surprise*. 2) Financial Phrasebank (Financial) [124] contains 850 test cases, where LLMs are asked to classify a given financial news with three categories: *negative*, *neutral*, *positive*. 3) Stanford Sentiment Treebank v2 (SST2) [151] consists of 872 sentences from movie reviews and human annotations of their sentiment, where the language model is asked to predict the sentiment from two classes: *positive* and *negative*.

Linguistic Acceptability. 1) The Corpus of Linguistic Acceptability (COLA) [174] is about English acceptability judgments drawn from books and journal articles on linguistic theory. Each example is a sequence of words annotated with whether it is a grammatical English sentence, and there are $1,040$ test cases in total.

Topic Classification. TC aims at categorizing the given sentence into predefined topics. We adopt AG_News [208] is a dataset that collects more than 1 million news articles, where LLMs are asked to classify a given news into four categories: *World*, *Sports*, *Business*, and *Sci/Tech*. There are $1,160$ test cases in total.

B.2.3 Experiment Setup

We conduct experiments primarily on LLAMA-2-7B-CHAT-HF, LLAMA-2-13B-CHAT-HF, and LLAMA-2-70B-CHAT-HF, where the model weights are downloaded from the website[2]. Since we cannot actually "sample" model weights as Bayesian Neural Net-

[2]https://ai.meta.com/resources/models-and-libraries/llama-downloads/

works, in order to let LLMs return different outputs, we leverage Beam Search since it considers multiple best options based on beam width using conditional probability, which is better than the sub-optimal Greedy search. The beam search is conducted with the beam size 10 and the max number of new tokens is set to be 16 uniformly across all datasets. We choose a different number of demonstrations (details are recorded in Table B.5) to allow the LLM to achieve the best performance on each dataset, and we sample demonstrations four times uniformly across all datasets.

B.2.4 Prompt Template

In this work, we uniformly apply the following prompt template for all datasets. Take the EMOTION dataset as an example, we summarize the prompt in Table B.6. Note that all datasets use the same template, small modifications are made on the actual label information and different demonstration numbers of different datasets.

B.2.5 Case Study

Table B.7 demonstrates the actual changes in AU and EU when presenting LLMs with different sizes and different demonstrations. Given the test query is: *I had stated to her the reason I feel so fearful is that I feel unsafe* with the ground truth label is *(4: fear)*, which is a sentence with a negative feeling. For LLAMA-2-7B, by presenting LLMs with more diverse demonstrations (containing both positive and negative sentences), the results would be more diverse between different beam search returned sequences, leading to a relatively higher AU than EU. For LLAMA-2-70B with a larger parameter space and model capability, the EU and AU are significantly reduced, which indicates the model is more confident in the generated output and the variation of data may not influence much to the prediction.

CSQA Test Case 1	Generation Results
	Prompt: Sean was in a rush to get home, but the light turned yellow and he was forced to do what? →(Sean was in a rush to get home, but the light turned yellow and he was forced to [MASK].)
DeBERTa-v3-large	stop
RoBERTa-large	something
RelBERT	park
T5-3b	He got out of his car.
UnifiedQA	The driver didn't turn on the lights?
GPT-3	If the driver didn't turn on the lights, it could be dangerous to drive
KEEP (Ours)	**Stop cars**. **Reasoning Chain**: **Yellow light** is a type of **traffic light**, **traffic light** is capable of **stop cars**.
CSQA Test Case 2	Generation Results
	Prompt: Reading newspaper is one of many ways to practice your what? →(Reading newspaper is one of many ways to practice your [MASK].)
DeBERTa-v3-large	English
RoBERTa-large	Writing
RelBERT	Writing
T5-3b	Skills. You'll spend plenty of time reading news items.
UnifiedQA	Technique. You can use print, Internet, your phone, the internet, and even more online technologies.
GPT-3	Reading comprehension.
KEEP (Ours)	**Essential ingredient of education**. **Reasoning Chain**: **Newspaper** requires **Reading**, **Reading** is defined as an **Essential ingredient of education**.
CSQA Test Case 3	Generation Results
	Prompt: A child wants to play, what would they likely want? →(A child wants to play, they would likely want [MASK].)
DeBERTa-v3-large	toys
RoBERTa-large	toys
RelBERT	something
T5-3b	Their father's toys.
UnifiedQA	Other children.
GPT-3	A child likely would want toys, games, outdoor activities, or sports equipment.
KEEP (Ours)	**Toys**. **Reasoning Chain**: **A child** is related to **toys**.

Table B.3: Three representative test cases on the CSQA dataset.

QASC Test Case 1	Generation Results
	Prompt: Cystitis is an example of something that can be treated with? →(Cystitis is an example of something that can be treated with [MASK].)
DeBERTa-v3-large	surgery
RoBERTa-large	medicine
RelBERT	surgery
T5-3b	Illness treatment.
UnifiedQA	Antibiotics, but a small proportion of people with the disease are not.
GPT-3	Cystitis can be treated with antibiotics, anti-inflammatory medications, and pain relievers.
KEEP (Ours)	**Antibiotics**. **Reasoning Chain**: **Cystitis** is a type of **Diease**, **Antibiotics** is capable of handling **Diease**.
QASC Test Case 2	Generation Results
	Prompt: What allows fish to move through the water without slowing down too much? →([MASK] allows fish to move through the water without slowing down too much.)
DeBERTa-v3-large	It
RoBERTa-large	Water
RelBERT	Agility
T5-3b	In fact, they are the fastest of all living creatures on our planet.
UnifiedQA	Fishes all swim through the water and they all started swimming fast.
GPT-3	Fish have evolved a variety of features that help them move through the water with minimal resistance.
KEEP (Ours)	**Fins**. **Reasoning Chain**: **Fish** is capable of **Swimming**, **Swimming** requires **Fins**.
QASC Test Case 3	Generation Results
	Prompt: What made sharks excellent predators? →(Sharks are excellent predators because of [MASK].)
DeBERTa-v3-large	Camouflage
RoBERTa-large	this
RelBERT	something
T5-3b	They could not just eat their prey.
UnifiedQA	They have a streamlined body shape.
GPT-3	Sharks have many adaptations that make them excellent predators.
KEEP (Ours)	**Jaws**. **Reasoning Chain**: **Sharks** is related to **Jaws**.

Table B.4: Three representative test cases on the QASC dataset.

	Random	Class
Emotion	6	1 per class
Financial	6	2 per class
SST2	4	2 per class
COLA	2	1 per class
AG_News	4	1 per class

Table B.5: Number of demonstrations selected in each dataset.

System Prompt	### Below is an instruction that describes a task. Clearly follow the instruction and write a short response to answer it.
Task Description	### Instruction: Classify the sentiment in the following text based on the six categories: [0: Sadness; 1: Joy, 2: Love; 3: Anger; 4: Fear, 5: Surprise]. Provide the information in a structured format WITHOUT additional comments, I just want the numerical label for each text.
Demonstrations	### Here are some examples: Example 1: Sentence: {i didnt feel humiliated} Category: {0: Sadness} Example 2: Sentence: {im grabbing a minute to post i feel greedy wrong} Category: {3: anger} Example 3: Sentence: {i have the feeling she was amused and delighted} Category: {1: joy} Example 4: Sentence: {i feel more superior dead chicken or grieving child} Category: {1: joy} Example 5: Sentence: {i get giddy over feeling elegant in a pencil skirt} Category: {1: joy} ...
Test Query	### Test Sentence: {} Category:

Table B.6: Prompt Template consists of four parts: 1) *System Prompt* aims at providing a basic hint of the task; 2) *Task Description* provides some details of the task, e.g., if it is a sentiment analysis task or how many labels are there; 3) *Few-shot Demonstrations* are leveraged to give LLMs some basic formats of how the returned responses can be constructed; and 4) *Test Query* is the final test query that we want LLMs to classify/categorize, and the LLM is only expected to return an exact answer to solve the given question.

Testing Query: I had stated to her the reason I feel so fearful is because I feel unsafe (4: fear)		Extracted Predictions	EU	AU
LLaMA-2-7B	1. i felt anger when at the end of a telephone call (3: anger) 2. i feel a little mellow today (1: joy) 3. i don t feel particularly agitated (4: fear) 4. i hate it when i feel fearful for absolutely no reason (4: fear) 5. im updating my blog because i feel shitty (0: sadness)	0, 0, 0, 1, 3 4, 3, 4, 4, 4	0.171	0.372
	1. i am feeling outraged it shows everywhere (4: fear) 2. i do feel insecure sometimes but who doesnt (4: fear) 3. i start to feel emotional (0: sadness) 4. i feel so cold a href http irish (3: anger) 5. i feel i have to agree with her even though i can imagine some rather unpleasant possible cases (0: sadness)	4, 4, 1, 3, 4 4, 4, 4, 5, 4	0.163	0.189
LLaMA-2-70B	1. i felt anger when at the end of a telephone call (3: anger) 2. i feel a little mellow today (1: joy) 3. i don t feel particularly agitated (4: fear) 4. i hate it when i feel fearful for absolutely no reason (4: fear) 5. im updating my blog because i feel shitty (0: sadness)	4, 3, 4, 3, 4 4, 4, 2, 4, 4	0.012	0.079
	1. i am feeling outraged it shows everywhere (4: fear) 2. i do feel insecure sometimes but who doesnt (4: fear) 3. i start to feel emotional (0: sadness) 4. i feel so cold a href http irish (3: anger) 5. i feel i have to agree with her even though i can imagine some rather unpleasant possible cases (0: sadness)	4, 4, 4, 4, 4 4, 4, 4, 4, 4	0.004	0.009

Table B.7: Case study on the actual EU and AU decomposed from the predictive uncertainty

Appendix C

Representation Learning on Textual-edge Graphs for Link Prediction

C.1 Algorithm of Transition Document Composition

We provide the full procedure of producing the document $d_{(i,j)}$ based on the node pair's transition graph $G_{(s,t)}$ in Algorithm 15. From Line 1-2, we obtain the respective local structure of s and t based on their transition graph $G_{(s,t)}$ by BFS search. From Line 3-4, for both G_s and G_t, we extract hidden edges that are not covered in the BFS tree. In Line 5, we get the intersection of the node sets V_s and V_t for recording the cross-paragraph nodes. Next, we initiate the document with the initial prompt and traverse each node in G_s and G_t to assign document section indices and relations to the document. Finally, we add hidden edges and cross-paragraph references as denoted in Line 12-13.

Algorithm 4: Transition Document Composition

Data: The transition graph $G_{(s,t)}$, the diameter K of $G_{(s,t)}$.
Result: Composed document $d_{(s,t)}$ with hierarchical relation, hidden edge references, and cross-paragraph references.

1. $G_s \leftarrow BFS(\text{ROOT} = s, \text{GRAPH} = G_{(s,t)}, \text{DEPTH} = K//2)$;
2. $G_t \leftarrow BFS(\text{ROOT} = t, \text{GRAPH} = G_{(s,t)}, \text{DEPTH} = K//2)$;
; /* Obtaining local structure of s and t' neighbor by breadth-first search with depth $K//2$. */
3. $E_s^{\text{hidden}} \leftarrow \{e_{ij} | \forall\, v_i \in G_s, v_j \in G_s, e_{ij} \in G_{(s,t)}, e_{ij} \notin G_s\}$;
4. $E_t^{\text{hidden}} \leftarrow \{e_{ij} | \forall\, v_i \in G_t, v_j \in G_t, e_{ij} \in G_{(s,t)}, e_{ij} \notin G_t\}$;
; /* For both subgraphs G_s and G_t, we obtain hidden edges. */
5. $V^{\text{cross}} \leftarrow V_s \cup V_t$;
; /* Get common nodes shared by G_s and G_t. */
6. Initiate the document $d_{(i,j)}$ with an initial prompt: ''''We have two paragraphs that summarize the relation between s and t...'';
7. **for** *each node* $v_i \in G_s$ **do**
   8. Assign document sections (e.g., [Sec. 1.1]) following pre-order traversal of G_s;
9. **end**
10. Assign hidden edge following E_s^{hidden} to $d_{(i,j)}$;
11. **for** *each node* $v_i \in G_t$ **do**
    11. Assign document sections (e.g., [Sec. 1.1]) following pre-order traversal of G_t;
13. **end**
14. Assign hidden edges following E_t^{hidden} to $d_{(i,j)}$;
15. Assign cross-paragraph reference $\forall\, v_i \in V^{\text{cross}}$;

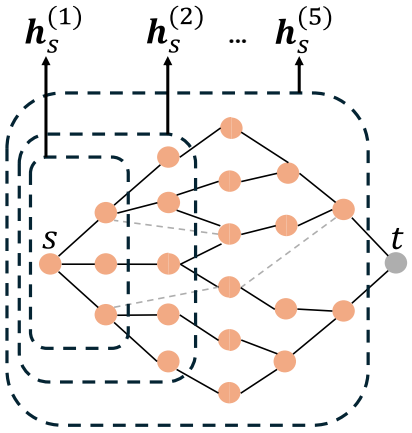


Figure C.1: The stratified representation learning of TGNN.

C.2 Transition Graph Neural Network

Considering a $(K-1)$-layer graph neural network, TGNN returns the embedding of s at each embedding updating layer of the GNN, as shown in Figure C.1. The strategy effectively minimizes duplicated computation by reusing convolution outputs across different cuts, resulting in a total computational cost equivalent to an K-layer GNN. While it shares computational complexity with a standard K-layer GNN, it is not strictly equivalent in terms of representation learning. By capturing and utilizing multi-scale representations at each layer n, the proposed approach offers potential advantages in expressiveness and performance for link prediction tasks.

C.3 Additional Experiments

Data. We run experiments on five real-world networks: Amazon-Movie [63], Amazon-Apps [63], GoodReads-Children [166], GoodReads-Crime [166], and StackOverflow. Amazon is a user-item interaction network, with reviews as textual content associated with the edges. Goodreads is a reader-book network, that utilizes readers' comments as textual information within the edges. StackOverflow is an expert-question network, and there will be an edge when an expert posts an answer to a question. The statistics of five datasets can be found in Table C.1.

Table C.1: Dataset Statistics

Dataset	# Node	# Edge
Goodreads-Children	192,036	734,640
Goodreads-Crime	385,203	1,849,236
Amazon-Apps	100,468	752,937
Amazon-Movie	173,986	1,697,533
Stack OverFlow	129,322	281,657

	Amazon-Apps		**Amazon-Movie**		**Goodreads-Children**		**Goodreads-Crime**		**StackOverflow**	
	AUC	**F1**	**AUC**	**F1**	**AUC**	**F1**	**AUC**	**F1**	**AUC**	**F1**
Single-Cut	0.7620	0.5880	0.7530	0.5530	0.9020	0.7050	0.9010	0.6520	0.9185	0.6841
Multi-Cut	**0.7697**	**0.5997**	**0.7731**	**0.5755**	**0.9146**	**0.7099**	**0.9124**	**0.6661**	**0.9374**	**0.6968**

Table C.2: Ablation study comparison between LINK2DOC with single cut versus multi-cut across five datasets, where the best values are bolded.

Ablation Study of Transition Graph Neural Network. We further demonstrate the effectiveness of considering multiple cuts to learn a better representation of $G_{(s,t)}$. As can be seen from Table C.2, Single-Cut denotes we only split $G_{(s,t)}$ in half, where each G_s and G_t have the depth of $K/2$.

The Multi-Cut strategy consistently outperforms the Single-Cut approach across all datasets. For example, in the Amazon-Apps dataset, the Multi-Cut method achieves an AUC of 0.7697 compared to 0.7620 with the Single-Cut method, and an F1 score of 0.5997 compared to 0.5880. This trend is observed across other datasets as well, such as Goodreads-Crime, where the Multi-Cut approach results in an AUC of 0.9124 versus 0.9010 for Single-Cut, and an F1 score of 0.6661 compared to 0.6520. The improvement is particularly notable in the StackOverflow dataset, where the AUC increases from 0.9185 with Single-Cut to 0.9374 with Multi-Cut, and the F1 score rises from 0.6841 to 0.6968.

Overall, the results clearly indicate that the Multi-Cut strategy leads to better performance in both AUC and F1 scores, suggesting that the model benefits from the multi-scale representation learning provided by the Multi-Cut approach. This likely enhances the model's ability to capture more comprehensive and diverse neighborhood information, leading to improved prediction accuracy.

Bibliography

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.

[2] Icek Ajzen. The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, 50:179–211, 12 1991. doi: 10.1016/0749-5978(91)90020-T.

[3] Khurshed Ali, Chih-Yu Wang, and Yi-Shin Chen. Boosting reinforcement learning in competitive influence maximization with transfer learning. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 395–400. IEEE, 2018.

[4] Alfonso Amayuelas, Liangming Pan, Wenhu Chen, and William Wang. Knowledge of knowledge: Exploring known-unknowns uncertainty with large language models. *arXiv preprint arXiv:2305.13712*, 2023.

[5] Hossein Amiri, Shiyang Ruan, Joon-Seok Kim, Hyunjee Jin, Hamdi Kavak, Andrew Crooks, Dieter Pfoser, Carola Wenk, and Andreas Zufle. Massive trajectory data based on patterns of life. In *Proceedings of the 31st ACM Interna-*

tional Conference on Advances in Geographic Information Systems, pages 1–4, 2023.

[6] Hossein Amiri, Will Kohn, Shiyang Ruan, Joon-Seok Kim, Hamdi Kavak, Andrew Crooks, Dieter Pfoser, Carola Wenk, and Andreas Züfle. The Patterns of Life Human Mobility Simulation. In *Proceedings of the 32nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2024)*, 2024.

[7] Hossein Amiri, Ruochen Kong, and Andreas Züfle. Urban anomalies: A simulated human mobility dataset with injected anomalies. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Geospatial Anomaly Detection (GeoAnomalies '24)*, page 1–11, 2024.

[8] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proc. of the ICML*, 2017.

[9] Guangji Bai, Chen Ling, and Liang Zhao. Temporal domain generalization with drift-aware dynamic neural networks. *arXiv preprint arXiv:2205.10664*, 2022.

[10] Guangji Bai, Chen Ling, Yuyang Gao, and Liang Zhao. Saliency-augmented memory completion for continual learning. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pages 244–252, 2023.

[11] Guangji Bai, Zheng Chai, Chen Ling, Shiyu Wang, Jiaying Lu, Nan Zhang, Tingwei Shi, Ziyang Yu, Mengdan Zhu, Yifei Zhang, et al. Beyond efficiency: A systematic survey of resource-efficient large language models. *arXiv preprint arXiv:2401.00625*, 2024.

[12] Guangji Bai, Yijiang Li, Chen Ling, Kibaek Kim, and Liang Zhao. Gradient-free adaptive global pruning for pre-trained language models. In *Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[13] Suman Banerjee, Mamata Jenamani, and Dilip Kumar Pratihar. A survey on influence maximization in a social network. *KAIS*, 62(9):3417–3455, 2020.

[14] Maria Becker, Katharina Korfhage, and Anette Frank. Coco-ex: A tool for linking concepts from texts to conceptnet. In *EACL*, 2021.

[15] Ning Bian, Xianpei Han, Bo Chen, and Le Sun. Benchmarking knowledge-enhanced commonsense question answering via knowledge-to-text transformation. In *AAAI*, 2021.

[16] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *Proc. of the ICML*, 2018.

[17] Antoine Bosselut, Ronan Le Bras, and Yejin Choi. Dynamic neuro-symbolic knowledge graph construction for zero-shot commonsense question answering. In *AAAI*, 2021.

[18] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020.

[19] Vinicius Caridá, Amir Jalilifard, Alex Mansano, and Rogers Cristo. Can netgan be improved on short random walks? In *Proc. of the BRACIS*, 2019.

[20] Ting-Yun Chang, Yang Liu, Karthik Gopalakrishnan, Behnam Hedayatnia, Pei Zhou, and Dilek Hakkani-Tur. Incorporating commonsense knowledge graph in pretrained models for social commonsense tasks. *arXiv preprint arXiv:2105.05457*, 2021.

[21] Pei Chen, Haotian Xu, Cheng Zhang, and Ruihong Huang. Crossroads, buildings and neighborhoods: A dataset for fine-grained location recognition. In

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3329–3339, 2022.

[22] Pei Chen, Soumajyoti Sarkar, Leonard Lausen, Balasubramaniam Srinivasan, Sheng Zha, Ruihong Huang, and George Karypis. Hytrel: Hypergraph-enhanced tabular data representation learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[23] Tiantian Chen, Siwen Yan, Jianxiong Guo, and Weili Wu. Touplegdd: A fine-designed solution of influence maximization by deep reinforcement learning. *arXiv preprint arXiv:2210.07500*, 2022.

[24] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[25] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proc. of the KDD*, pages 1029–1038, 2010.

[26] Xiaojun Chen, Shengbin Jia, and Yang Xiang. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, 141:112948, 2020.

[27] Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. Label-free node classification on graphs with large language models (llms). *arXiv preprint arXiv:2310.04668*, 2023.

[28] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61, 2024.

[29] Kamaljit Chowdhary and Paul Dupuis. Distinguishing and integrating aleatoric and epistemic variation in uncertainty quantification. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 47(3):635–662, 2013.

[30] Tanmoy Chowdhury, Chen Ling, Junji Jiang, Junxiang Wang, My T Thai, and Liang Zhao. Deep graph representation learning influence maximization with accelerated inference. *Available at SSRN 4663083*, 2023.

[31] Tanmoy Chowdhury, Chen Ling, Xuchao Zhang, Xujiang Zhao, Guangji Bai, Jian Pei, Haifeng Chen, and Liang Zhao. Knowledge-enhanced neural machine reasoning: A review. *arXiv preprint arXiv:2302.02093*, 2023.

[32] Tanmoy Chowdhury, Chen Ling, Junji Jiang, Junxiang Wang, My T Thai, and Liang Zhao. Deep graph representation learning for influence maximization with accelerated inference. *Neural Networks*, 180:106649, 2024.

[33] Hejie Cui, Jiaying Lu, Shiyu Wang, Ran Xu, Wenjing Ma, Shaojun Yu, Yue Yu, Xuan Kan, Tianfan Fu, Chen Ling, et al. A survey on knowledge graphs for healthcare: Resources, application progress, and promise. In *ICML 3rd Workshop on Interpretable Machine Learning in Healthcare (IMLH)*, 2023.

[34] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[35] Chunhui Deng, Huifang Deng, and Zhipeng Fu. Modeling and study of information transfer in complex network. In *2018 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*, pages 668–675, 2018.

[36] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Uncertainty decomposition in bayesian neural networks with latent variables. *arXiv preprint arXiv:1706.08495*, 2017.

[37] Shrey Desai and Greg Durrett. Calibration of pre-trained transformers. *arXiv preprint arXiv:2003.07892*, 2020.

[38] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*, 2018.

[39] Swapnil Dhamal, KJ Prabuchandran, and Y Narahari. Information diffusion in social networks in two phases. *IEEE TNSE*, 3(4):197–210, 2016.

[40] Nguyen Hoang Khoi Do, Tanmoy Chowdhury, Chen Ling, Liang Zhao, and My T Thai. Mim-reasoner: Learning with theoretical guarantees for multiplex influence maximization. In *International Conference on Artificial Intelligence and Statistics*, pages 2296–2304, 2024.

[41] Brian W Dolhansky and Jeff A Bilmes. Deep submodular functions: Definitions and learning. *Advances in Neural Information Processing Systems*, 29, 2016.

[42] Ming Dong, Bolong Zheng, Nguyen Quoc Viet Hung, Han Su, and Guohui Li. Multiple rumor source detection with graph convolutional networks. In *CIKM*, pages 569–578, 2019.

[43] Yuxiao Dong, Jie Tang, Sen Wu, Jilei Tian, Nitesh V Chawla, Jinghai Rao, and Huanhuan Cao. Link prediction and recommendation across heterogeneous social networks. In *Proc. of the ICDM*, pages 181–190, 2012.

[44] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proc. of the KDD*, 2017.

[45] Nan Du, Yingyu Liang, Maria Balcan, and Le Song. Influence function learning in information diffusion networks. In *ICML*, pages 2016–2024, 2014.

[46] Paul Erdős, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[47] Ekaterina Fadeeva, Roman Vashurin, Akim Tsvigun, Artem Vazhentsev, Sergey Petrakov, Kirill Fedyanin, Daniil Vasilev, Elizaveta Goncharova, Alexander Panchenko, Maxim Panov, et al. Lm-polygraph: Uncertainty estimation for language models. *arXiv preprint arXiv:2311.07383*, 2023.

[48] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*, 2023.

[49] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proc. of the CIKM*, 2017.

[50] Junruo Gao, Chen Ling, Carl Yang, and Liang Zhao. Helper recommendation with seniority control in online health community. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*, pages 253–261, 2024.

[51] Matthew Gerber, Andrew Gordon, and Kenji Sagae. Open-domain commonsense reasoning using discourse relations from a corpus of weblog stories. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 43–51, 2010.

[52] Abdorasoul Ghasemi and Hermann de Meer. Robustness of interdependent power grid and communication networks to cascading failures. *IEEE Transactions on Network Science and Engineering*, 2023.

[53] Liyu Gong and Qiang Cheng. Exploiting edge features for graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[54] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. of the NeurIPS*, 2014.

[55] Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. Graphgen: A scalable approach to domain-agnostic labeled graph generation. In *Proc. of the WebConf*, pages 1253–1263, 2020.

[56] Jiayan Guo, Lun Du, and Hengyu Liu. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*, 2023.

[57] Qintian Guo, Sibo Wang, Zhewei Wei, and Ming Chen. Influence maximization revisited: Efficient reverse reachable set generation with bound tightened. In *Proc. of the SIGMOD*, pages 2167–2181, 2020.

[58] Xiaojie Guo and Liang Zhao. A systematic survey on deep generative models for graph generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5370–5390, 2022.

[59] Xiaojie Guo, Liang Zhao, Cameron Nowzari, Setareh Rafatirad, Houman Homayoun, and Sai Manoj Pudukotai Dinakarrao. Deep multi-attributed graph translation with node-edge co-evolution. In *Proc. of the ICDM*, pages 250–259, 2019.

[60] Amarnath Gupta. Generating large-scale heterogeneous graphs for benchmarking. In *Specifying Big Data Benchmarks*, pages 113–128. 2012.

[61] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1024–1034, 2017.

[62] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *ICLR*, 2021.

[63] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517, 2016.

[64] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[65] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proc. of the WebConf*, pages 2704–2710, 2020.

[66] Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. Can llms effectively leverage graph structural information: when and why. *arXiv preprint arXiv:2309.16595*, 2023.

[67] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[68] Haozhe Ji, Pei Ke, Shaohan Huang, Furu Wei, and Minlie Huang. Generating commonsense explanation by extracting bridge concepts from reasoning paths. *arXiv preprint arXiv:2009.11753*, 2020.

[69] Jiaojiao Jiang, Sheng Wen, Shui Yu, Yang Xiang, and Wanlei Zhou. Identifying propagation sources in networks: State-of-the-art and comparative studies. *IEEE Commun. Surv. Tutor.*, 19(1):465–481, 2016.

[70] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. Structgpt: A general framework for large language model to reason

over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, 2023.

[71] Junji Jiang, Chen Ling, Hongyi Li, Guangji Bai, Xujiang Zhao, and Liang Zhao. Quantifying uncertainty in graph neural network explanations. *Frontiers in big Data*, 7:1392662, 2024.

[72] Xiaodong Jiang, Pengsheng Ji, and Sheng Li. Censnet: Convolution with edge-node switching in graph neural networks. In *IJCAI*, pages 2656–2662, 2019.

[73] Yanjie Jiang, Hui Liu, Nan Niu, Lu Zhang, and Yamin Hu. Extracting concise bug-fixing patches from human-written patches in version control systems. In *IEEE/ACM 43rd International Conference on Software Engineering (ICSE 2021)*, pages 686–698, 2021.

[74] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9: 962–977, 2021.

[75] Bowen Jin, Yu Zhang, Yu Meng, and Jiawei Han. Edgeformers: Graph-empowered transformers for representation learning on textual-edge networks. In *The Eleventh International Conference on Learning Representations*, 2022.

[76] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. Large language models on graphs: A comprehensive survey. *arXiv preprint arXiv:2312.02783*, 2023.

[77] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[78] Mihir Kale and Abhinav Rastogi. Text-to-text pre-training for data-to-text tasks. *arXiv preprint*, 2020.

[79] Harshavardhan Kamarthi, Priyesh Vijayan, Bryan Wilder, Balaraman Ravindran, and Milind Tambe. Influence maximization in unknown social networks: Learning policies for effective graph sampling. *arXiv preprint arXiv:1907.11625*, 2019.

[80] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proc. of the KDD*, 2003.

[81] Hana Khamfroush, Novella Bartolini, Thomas F La Porta, Ananthram Swami, and Justin Dillman. On propagation of phenomena in interdependent networks. *IEEE Transactions on network science and engineering*, 3(4):225–239, 2016.

[82] Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv*, 2020.

[83] Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. Qasc: A dataset for question answering via sentence composition. In *AAAI*, 2020.

[84] Joon-Seok Kim, Hyunjee Jin, Hamdi Kavak, Ovi Chris Rouly, Andrew Crooks, Dieter Pfoser, Carola Wenk, and Andreas Züfle. Location-based social network data generation based on patterns of life. In *2020 21st IEEE International Conference on Mobile Data Management (MDM)*, pages 158–167, 2020. doi: 10.1109/MDM48529.2020.00038.

[85] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[86] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[87] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

[88] Sanjay Kumar, Abhishek Mallik, Anavi Khetarpal, and BS Panda. Influence maximization in social networks using graph embedding and graph neural network. *Information Sciences*, 607:1617–1636, 2022.

[89] Yong Kyu Lee, Seong-Joon Yoo, Kyoungro Yoon, and P Bruce Berra. Index structures for structured documents. In *Proceedings of the first ACM international conference on Digital libraries*, pages 91–99, 1996.

[90] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proc. of the KDD*, 2007.

[91] Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pages 6437–6449. PMLR, 2021.

[92] Hui Li, Mengting Xu, Sourav S Bhowmick, Changsheng Sun, Zhongyuan Jiang, and Jiangtao Cui. Disco: Influence maximization meets network embedding and deep learning. *arXiv preprint arXiv:1906.07378*, 2019.

[93] Hui Li, Mengting Xu, Sourav S Bhowmick, Joty Shafiq Rayhan, Changsheng Sun, and Jiangtao Cui. Piano: Influence maximization meets deep reinforcement learning. *IEEE Transactions on Computational Social Systems*, 2022.

[94] Mingchen Li, Chen Ling, Rui Zhang, and Liang Zhao. A condensed transition graph framework for zero-shot link prediction with large language models. In *International Conference on Data Mining*, 2024.

[95] Xiang Li, J David Smith, Thang N Dinh, and My T Thai. Tiptop:(almost) exact solutions for influence maximization in billion-scale networks. *IEEE/ACM Transactions on Networking*, 27(2):649–661, 2019.

[96] Yichuan Li, Kaize Ding, and Kyumin Lee. Grenade: Graph-centric language model for self-supervised representation learning on text-attributed graphs. *EMNLP*, 2023.

[97] Yuchen Li, Ju Fan, Yanhao Wang, and Kian-Lee Tan. Influence maximization on social graphs: A survey. *TKDE*, 30(10):1852–1872, 2018.

[98] Zhuofeng Li, Zixing Gou, Xiangnan Zhang, Zhongyuan Liu, Sirui Li, Yuntong Hu, Chen Ling, Zheng Zhang, and Liang Zhao. Teg-db: A comprehensive dataset and benchmark of textual-edge graphs. In *Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[99] Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *arXiv preprint*, 2019.

[100] Bill Yuchen Lin, Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Xiang Ren, and William Cohen. Differentiable open-ended commonsense reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4611–4625, 2021.

[101] Su-Chen Lin, Shou-De Lin, and Ming-Syan Chen. A learning-based framework to handle multi-round multi-party influence maximization on social networks. In

Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 695–704, 2015.

[102] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[103] Yishi Lin, Wei Chen, and John CS Lui. Boosting information spread: An algorithmic approach. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 883–894, 2017.

[104] Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. Generating with confidence: Uncertainty quantification for black-box large language models. *arXiv preprint arXiv:2305.19187*, 2023.

[105] C. Ling, T. Chowdhury, J. Jiang, J. Wang, X. Zhang, H. Chen, and L. Zhao. Deepgar: Deep graph learning for analogical reasoning. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1065–1070, 2022.

[106] Chen Ling, Guangmo Tong, and Mozi Chen. Nestpp: Modeling thread dynamics in online discussion forums. In *HT*, pages 251–260, 2020.

[107] Chen Ling, Di Cui, Guangmo Tong, and Jianming Zhu. On forecasting dynamics in online discussion forums. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2021.

[108] Chen Ling, Carl Yang, and Liang Zhao. Deep generation of heterogeneous networks. In *2021 IEEE international conference on data mining (ICDM)*, pages 379–388. IEEE, 2021.

[109] Chen Ling, Hengning Cao, and Liang Zhao. Stgen: Deep continuous-time

spatiotemporal graph generation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 340–356, 2022.

[110] Chen Ling, Junji Jiang, Junxiang Wang, and Zhao Liang. Source localization of graph diffusion via variational autoencoders for graph inverse problems. In *Proceedings of the 28th ACM SIGKDD*, pages 1010–1020, 2022.

[111] Chen Ling, Junji Jiang, Junxiang Wang, My T Thai, Renhao Xue, James Song, Meikang Qiu, and Liang Zhao. Deep graph representation learning and optimization for influence maximization. In *International Conference on Machine Learning*, pages 21350–21361, 2023.

[112] Chen Ling, Carl Yang, and Liang Zhao. Motif-guided heterogeneous graph deep generation. *Knowledge and Information Systems*, 65(7):3099–3124, 2023.

[113] Chen Ling, Xuchao Zhang, Xujiang Zhao, Yanchi Liu, Wei Cheng, Mika Oishi, Takao Osaki, Katsushi Matsuda, Haifeng Chen, and Liang Zhao. Open-ended commonsense reasoning with unrestricted answer candidates. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8035–8047, 2023.

[114] Chen Ling, Xuchao Zhang, Xujiang Zhao, Yifeng Wu, Yanchi Liu, Wei Cheng, Haifeng Chen, and Liang Zhao. Knowledge-enhanced prompt for open-domain commonsense reasoning. *1st AAAI Workshop on Uncertainty Reasoning and Quantification in Decision Making*, 2023.

[115] Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Xuchao Zhang, Tianjiao Zhao, et al. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *arXiv preprint arXiv:2305.18703*, 2023.

[116] Chen Ling, Xujiang Zhao, Xuchao Zhang, Yanchi Liu, Wei Cheng, Haoyu Wang, Zhengzhang Chen, Takao Osaki, Katsushi Matsuda, Haifeng Chen, et al. Improving open information extraction with large language models: A study on demonstration uncertainty. *arXiv preprint arXiv:2309.03433*, 2023.

[117] Chen Ling, Tanmoy Chowdhury, Jie Ji, Sirui Li, Andreas Züfle, and Liang Zhao. Source localization for cross network information diffusion. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5419–5429, 2024.

[118] Chen Ling, Xujiang Zhao, Xuchao Zhang, Wei Cheng, Yanchi Liu, Yiyou Sun, Mika Oishi, Takao Osaki, Katsushi Matsuda, Jie Ji, et al. Uncertainty quantification for in-context learning of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: HLT*, pages 3357–3370, 2024.

[119] Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. Generated knowledge prompting for commonsense reasoning. *arXiv preprint*, 2021.

[120] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint*, 2019.

[121] Jiaying Lu, Jiaming Shen, Bo Xiong, Wenjing Ma, Steffen Staab, and Carl Yang. Hiprompt: Few-shot biomedical knowledge fusion via hierarchy-oriented prompting. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2052–2056, 2023.

[122] Kaixin Ma, Filip Ilievski, Jonathan Francis, Yonatan Bisk, Eric Nyberg, and

Alessandro Oltramari. Knowledge-driven data construction for zero-shot evaluation in commonsense question answering. In *AAAI*, 2021.

[123] Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. *arXiv preprint arXiv:2002.07650*, 2020.

[124] P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65, 2014.

[125] Sahil Manchanda, Akash Mittal, Anuj Dhawan, Sourav Medya, Sayan Ranu, and Ambuj Singh. Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs. *Advances in Neural Information Processing Systems*, 33:20000–20011, 2020.

[126] Abraham H Maslow. A theory of human motivation. *Psychological review*, 50 (4):370, 1943.

[127] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[128] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[129] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

[130] Mohamed Mokbel, Mahmoud Sakr, Li Xiong, Andreas Züfle, Jussara Almeida, Taylor Anderson, Walid Aref, Gennady Andrienko, Natalia Andrienko, Yang

Cao, et al. Mobility data science: Perspectives and challenges. *ACM Transactions on Spatial Algorithms and Systems*.

[131] Mohamed Mokbel, Mahmoud Sakr, Li Xiong, Andreas Züfle, Jussara Almeida, Taylor Anderson, Walid Aref, Gennady Andrienko, Natalia Andrienko, Yang Cao, et al. Mobility data science (dagstuhl seminar 22021). In *Dagstuhl reports*, volume 12. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[132] Hung T Nguyen, My T Thai, and Thang N Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *Proc. of the SIGMOD*, 2016.

[133] Hung T Nguyen, My T Thai, and Thang N Dinh. A billion-scale approximation algorithm for maximizing benefit in viral marketing. *IEEE/ACM Transactions On Networking*, 25(4):2419–2429, 2017.

[134] Bo Pan, Zheng Zhang, Yifei Zhang, Yuntong Hu, and Liang Zhao. Distilling large language models for text-attributed graph learning. *arXiv preprint arXiv:2402.12022*, 2024.

[135] George Panagopoulos, Fragkiskos Malliaros, and Michalis Vazirgiannis. Multi-task learning for influence estimation and maximization. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[136] Bhargavi Paranjape, Julian Michael, Marjan Ghazvininejad, Hannaneh Hajishirzi, and Luke Zettlemoyer. Prompting contrastive explanations for commonsense reasoning tasks. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4179–4192, 2021.

[137] B Aditya Prakash, Jilles Vreeken, and Christos Faloutsos. Spotting culprits in epidemics: How many and which ones? In *ICDM*, pages 11–20, 2012.

[138] Ruizhong Qiu, Dingsu Wang, Lei Ying, H Vincent Poor, Yifang Zhang, and Hanghang Tong. Reconstructing graph diffusion history from a single snapshot. *arXiv preprint arXiv:2306.00488*, 2023.

[139] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[140] Vipula Rawte, Amit Sheth, and Amitava Das. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922*, 2023.

[141] Ali Razavi, Aaron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-VAEs. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BJe0Gn0cY7`.

[142] Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, pages 90–95, 2011.

[143] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[144] Kazumi Saito, Masahiro Kimura, Kouzou Ohara, and Hiroshi Motoda. Efficient discovery of influential nodes for sis models in social networks. *Knowledge and information systems*, 30(3):613–635, 2012.

[145] Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, 2018.

[146] Sushila Shelke and Vahida Attar. Source detection of rumor in social network–a review. *Online Social Networks and Media*, 9:30–42, 2019.

[147] Chuan Shi, Zhiqiang Zhang, Ping Luo, Philip S Yu, Yading Yue, and Bin Wu. Semantic path based personalized recommendation on weighted heterogeneous information networks. In *Proc. of the CIKM*, pages 453–462, 2015.

[148] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. A survey of heterogeneous information network analysis. *TKDE*, 29(1):17–37, 2016.

[149] Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Unsupervised commonsense question answering with self-talk. *arXiv preprint*, 2020.

[150] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *Proc. of the ICANN*, pages 412–422, 2018.

[151] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[152] Mingming Sun and Ping Li. Graph to graph: a topology aware approach for graph structures learning and generation. In *Proc. of the AISTATS*, 2019.

[153] Yizhou Sun and Jiawei Han. Meta-path-based search and mining in heterogeneous information networks. *Tsinghua Science and Technology*, 2013.

[154] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. Pathsim:

Meta path-based top-k similarity search in heterogeneous information networks. *Proc. of the VLDB Endowment*, 4(11):992–1003, 2011.

[155] Yizhou Sun, Brandon Norick, Jiawei Han, Xifeng Yan, Philip S Yu, and Xiao Yu. Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. *TKDD*, 2013.

[156] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of NAACL-HLT*, pages 4149–4158, 2019.

[157] Jing Tang, Xueyan Tang, Xiaokui Xiao, and Junsong Yuan. Online processing algorithms for influence maximization. In *Proc. of the SIGMOD*, pages 991–1005, 2018.

[158] Youze Tang, Xiaokui Xiao, and Yanchen Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proc. of the SIGMOD*, pages 75–86, 2014.

[159] Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *Proc. of the SIGMOD*, 2015.

[160] Shan Tian, Songsong Mo, Liwei Wang, and Zhiyong Peng. Deep reinforcement learning-based approach to tackle topic-aware influence maximization. *Data Science and Engineering*, 5(1):1–11, 2020.

[161] Diman Zad Tootaghaj, Novella Bartolini, Hana Khamfroush, Ting He, Nilanjan Ray Chaudhuri, and Thomas La Porta. Mitigation and recovery from cascading failures in interdependent networks under uncertainty. *IEEE Transactions on Control of Network Systems*, 6(2):501–514, 2018.

[162] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[163] Asahi Ushio, Steven Schockaert, and Jose Camacho-Collados. Distilling Relation Embeddings from Pre-trained Language Models. In *EMNLP 2021*, 2021.

[164] Sharan Vaswani, Branislav Kveton, Zheng Wen, Mohammad Ghavamzadeh, Laks VS Lakshmanan, and Mark Schmidt. Model-independent online learning for influence maximization. In *ICML*, 2017.

[165] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv*, 2017.

[166] Mengting Wan, Rishabh Misra, Ndapa Nakashole, and Julian McAuley. Fine-grained spoiler detection from large-scale review corpora. *arXiv preprint arXiv:1905.13416*, 2019.

[167] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Proc. of the AAAI*, 2018.

[168] Junxiang Wang, Junji Jiang, and Liang Zhao. An invertible graph diffusion neural network for source localization. In *Proceedings of the ACM Web Conference 2022*, pages 1058–1069, 2022.

[169] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *Proc. of the WebConf*, 2019.

[170] Yanhao Wang, Qi Fan, Yuchen Li, and Kian-Lee Tan. Real-time influence maximization on dynamic social streams. *arXiv preprint arXiv:1702.01586*, 2017.

[171] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5):1–12, 2019.

[172] Zhen Wang, Dongpeng Hou, Chao Gao, Xiaoyu Li, and Xuelong Li. Lightweight source localization for large-scale social networks. In *Proceedings of the ACM Web Conference 2023*, pages 286–294, 2023.

[173] Zheng Wang, Chaokun Wang, Jisheng Pei, and Xiaojun Ye. Multiple source detection without knowing the underlying propagation model. In *AAAI*, 2017.

[174] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

[175] Zhihao Wen and Yuan Fang. Augmenting low-resource text classification with graph-grounded pre-training and prompting. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 506–516, 2023.

[176] Lisa Wimmer, Yusuf Sale, Paul Hofman, Bernd Bischl, and Eyke Hüllermeier. Quantifying aleatoric and epistemic uncertainty in machine learning: Are conditional entropy and mutual information appropriate measures? In *Uncertainty in Artificial Intelligence*, pages 2282–2292, 2023.

[177] Guanchen Wu, Chen Ling, Ilana Graetz, and Liang Zhao. Ontology extension by online clustering with large language model agents. *Frontiers in Big Data*, 7:1463543, 2024.

[178] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[179] Yijun Xiao and William Yang Wang. Quantifying uncertainties in natural language processing tasks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7322–7329, 2019.

[180] Yuxin Xiao, Paul Pu Liang, Umang Bhatt, Willie Neiswanger, Ruslan Salakhutdinov, and Louis-Philippe Morency. Uncertainty quantification with pre-trained language models: A large-scale empirical analysis. *arXiv preprint arXiv:2210.04714*, 2022.

[181] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

[182] Yiqing Xie, Sha Li, Carl Yang, Raymond Chi-Wing Wong, and Jiawei Han. When do gnns work: Understanding and improving neighborhood aggregation. In *IJCAI'20: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence,{IJCAI} 2020*, volume 2020, 2020.

[183] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[184] Xovee Xu, Tangjiang Qian, Zhe Xiao, Ni Zhang, Jin Wu, and Fan Zhou. Pgsl: A probabilistic graph diffusion model for source localization. *Expert Systems with Applications*, 238:122028, 2024.

[185] Qi Xuan, Xincheng Shu, Zhongyuan Ruan, Jinbao Wang, Chenbo Fu, and Guanrong Chen. A self-learning information diffusion model for smart social

networks. *IEEE Transactions on Network Science and Engineering*, 7(3):1466–1480, 2019.

[186] Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, et al. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 36:17238–17264, 2023.

[187] Xin Yan, Hui Fang, and Qiang He. Diffusion model for graph inverse problems: Towards effective source localization on complex networks. *Advances in Neural Information Processing Systems*, 36, 2024.

[188] Carl Yang, Yichen Feng, Pan Li, Yu Shi, and Jiawei Han. Meta-graph based hin spectral embedding: Methods, analyses, and insights. In *Proc. of the ICDM*, 2018.

[189] Carl Yang, Mengxiong Liu, Frank He, Xikun Zhang, Jian Peng, and Jiawei Han. Similarity modeling on heterogeneous networks via automatic path discovery. In *Proc. of the ECML-PKDD*, 2018.

[190] Carl Yang, Jieyu Zhang, and Jiawei Han. Neural embedding propagation on heterogeneous networks. In *Proc. of the ICDM*, 2019.

[191] Carl Yang, Peiye Zhuang, Wenhan Shi, Alan Luu, and Pan Li. Conditional structure generation through graph variational generative adversarial nets. In *Proc. of the NIPS*, pages 1340–1351, 2019.

[192] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. Heterogeneous network representation learning: A unified framework with survey and benchmark. *IEEE Transactions on Knowledge and Data Engineering*, 34(10): 4854–4873, 2020.

[193] Carl Yang, Haonan Wang, Ke Zhang, Liang Chen, and Lichao Sun. Secure deep graph generation with link differential privacy. In *Proc. of the IJCAI*, 2021.

[194] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *IJCAI*, volume 2015, pages 2111–2117, 2015.

[195] Lan Yang, Zhiwu Li, and Alessandro Giua. Containment of rumor spread in complex social networks. *Information Sciences*, 506:113–130, 2020.

[196] Yulei Yang and Dongsheng Li. Nenn: Incorporate node and edge features in graph neural networks. In *Asian conference on machine learning*, pages 593–608. PMLR, 2020.

[197] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint*, 2021.

[198] Mao Ye, Xingjie Liu, and Wang-Chien Lee. Exploring social influence for recommendation: a generative model approach. In *Proc. of the SIGIR*, pages 671–680, 2012.

[199] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134*, 2023.

[200] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *Proc. of the ICML*, 2018.

[201] Jialin Yu, Alexandra I Cristea, Anoushka Harit, Zhongtian Sun, Olanrewaju Tahir Aduragba, Lei Shi, and Noura Al Moubayed. Efficient uncertainty

quantification for multilabel text classification. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022.

[202] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. In *Proc. of the NIPS*, pages 11983–11993, 2019.

[203] Wenyu Zang, Peng Zhang, Chuan Zhou, and Li Guo. Locating multiple sources in social networks under the sir model: A divide-and-conquer approach. *Journal of Computational Science*, 10:278–287, 2015.

[204] Cai Zhang, Weimin Li, Dingmei Wei, Yanxia Liu, and Zheng Li. Network dynamic gcn influence maximization algorithm with leader fake labeling mechanism. *IEEE Transactions on Computational Social Systems*, 2022.

[205] Liming Zhang, Liang Zhao, Shan Qin, Dieter Pfoser, and Chen Ling. Tg-gan: Continuous-time temporal graph deep generative models with time-validity constraints. In *theWebConf*, pages 2104–2116, 2021.

[206] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

[207] Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. Iteratively learning embeddings and rules for knowledge graph reasoning. In *Proc. of the WebConf*, pages 2366–2377, 2019.

[208] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

[209] Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. Greaselm: Graph reasoning enhanced language models. In *International Conference on Learning Representations*, 2021.

[210] Yifei Zhang, Bo Pan, Chen Ling, Yuntong Hu, and Liang Zhao. Elad: Explanation-guided large language models active distillation. In *The 62nd Annual Meeting of the Association for Computational Linguistics (ACL-Findings)*, 2024.

[211] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational inference. *ICLR*, 2023.

[212] Xujiang Zhao, Feng Chen, Shu Hu, and Jin-Hee Cho. Uncertainty aware semi-supervised learning on graph data. *Advances in Neural Information Processing Systems*, 33:12827–12836, 2020.

[213] Ding Zhou, Sergey A Orshanskiy, Hongyuan Zha, and C Lee Giles. Co-ranking authors and documents in a heterogeneous network. In *Proc. of the ICDM*, 2007.

[214] Xuhui Zhou, Yue Zhang, Leyang Cui, and Dandan Huang. Evaluating commonsense in pre-trained language models. In *AAAI*, 2020.

[215] Kai Zhu and Lei Ying. Information source detection in the sir model: A sample-path-based approach. *IEEE/ACM TON*, 24(1):408–421, 2016.

[216] Kai Zhu, Zhen Chen, and Lei Ying. Catch'em all: Locating multiple diffusion sources in networks with partial observations. In *AAAI*, 2017.

[217] Shichao Zhu, Chuan Zhou, Shirui Pan, Xingquan Zhu, and Bin Wang. Relation structure-aware heterogeneous graph neural network. In *2019 IEEE international conference on data mining*, pages 1534–1539, 2019.

[218] Tao Zou, Le Yu, Yifei Huang, Leilei Sun, and Bowen Du. Pretraining language models with text-attributed heterogeneous graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

[219] Andreas Züfle, Carola Wenk, Dieter Pfoser, Andrew Crooks, Joon-Seok Kim, Hamdi Kavak, Umar Manzoor, and Hyunjee Jin. Urban life: a model of people and places. *Computational and Mathematical Organization Theory*, 29(1):20–51, 2023.