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:

_____ _____

Jiaying Lu Date

Learning Structured Knowledge from Real-World Data
without Excessive Annotations

By

Jiaying Lu
Doctor of Philosophy

Computer Science and Informatics

Carl Yang, Ph.D.
Advisor

Fei Liu, Ph.D.
Committee Member

Jiaming Shen, Ph.D.
Committee Member

Yuanzhe Xi, Ph.D.
Committee Member

Accepted:

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

Date

Learning Structured Knowledge from Real-World Data without Excessive Annotations

By

Jiaying Lu
B.Sc., Beijing University of Posts and Telecommunications, Beijing, China, 2014
M.Eng., Beijing University of Posts and Telecommunications, Beijing, China, 2017
M.Sc., Emory University, Atlanta, GA, USA, 2022

Advisor: Carl Yang, 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
in Computer Science and Informatics
2024

Abstract

Learning Structured Knowledge from Real-World Data
without Excessive Annotations
By Jiaying Lu

In a world where vast quantities of data are continually generated by humans every day, the majority of the data remains unstructured, posing a significant challenge to knowledge discovery and insight generation. Unleashing the full potential of these valuable information sources requires organizing the data with interconnections and contexts. This dissertation delves into the fundamental task of transforming unstructured real-world data into structured knowledge, all without an excessive reliance on manual annotations. Particularly, I investigate three areas of research, including: (1) Constructing concept maps from unstructured text data. We first develop an innovative unsupervised concept map construction method by utilizing syntactic parsing techniques [48]. Then we further study how to translate the initial parsing-based concept maps into more concise task-oriented concept maps under the guidance of weak supervision signal from downstream tasks [50]. (2) Aligning and completing taxonomic knowledge graphs (KGs). Given the widely available KGs scattered in different sites, it is urgent to integrate them into a comprehensive knowledge base to harness knowledge-centric applications. We propose a novel perspective to leverage expert-curated taxonomies as the backbone to aligning various KGs [52] under a few-shot manner. We further study how to complete taxonomic KGs after initial alignment between them [49]. (3) Empowering downstream applications with structured knowledge. Finally, we explore how to harness the performance of downstream applications with learned structured knowledge. For instance, we utilize similarity-based communities for multiclass classification [51]. Together, these works cover the whole life cycle of construction, integration, completion, and utilization of structured knowledge.

Learning Structured Knowledge from Real-World Data without Excessive Annotations

By

Jiaying Lu
B.Sc., Beijing University of Posts and Telecommunications, Beijing, China, 2014
M.Eng., Beijing University of Posts and Telecommunications, Beijing, China, 2017
M.Sc., Emory University, Atlanta, GA, USA, 2022

Advisor: Carl Yang, 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
in Computer Science and Informatics
2024

Acknowledgments

I would like to express my sincere gratitude to my advisor, Dr. Carl Yang, for his invaluable guidance and support throughout my PhD study. The past five years have been a memorable journey for me. I enjoy the moments of getting paper acceptance notifications, as well as the sleepless nights aiming for submission deadlines. I have learned a lot from Dr. Yang, especially the research taste regarding what is good research. I am also grateful to the Emory Graph Mining Group for its open-minded and inclusive culture. I would like to acknowledge our Department of Computer Science for providing the computing resources and support for completing my dissertation. Additionally, I'd like to recognize the generous funding support from Emory Laney Graduate School that financed my first two years of Ph.D. studies.

This endeavor would not have been possible without the mentorship of my committee members, Dr. Yuanzhe Xi, Dr. Fei Liu, and Dr. Jiaming Shen. Their insightful feedback and valuable suggestions have greatly improved the completeness and quality of this dissertation. I had the pleasure of collaborating with Dr. Jinho Choi and Dr. Han He for our research project on utilizing a series of coreNLP tools developed by Emory NLP lab.

Lastly, I'm extremely grateful to my family for their support and love in encouraging me to pursue my Ph.D. degree. Special thanks to my wife Dr. Wenjing Ma. Her research passion for computational biology has inspired me to explore AI for healthcare research. I'd also like to mention two important works that inspired me to quit my full-time job and pursue a Ph.D.: "Stoner" (novel) by Dr. John Williams and "La La Land" (movie) by Damien Chazelle.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Research Roadmap 2

1.3 Dissertation Outline 3

2 Learning to Construct Concept Maps **5**

2.1 Background 5

2.2 Unsupervised Syntactic Parsing-based Concept Map Construction . . 6

2.2.1 Preliminaries 6

2.2.2 Proposed Approach 8

2.2.3 Candidate Generation Module 8

2.2.4 Graph Construction Module 9

2.3 Weakly Supervised Concept Map Generation through Task-Guided Graph Translation 12

2.3.1 Problems Definition 14

2.3.2 *GT-D2G* 15

2.3.3 Experiments 26

3 Learn to Aligning and Completing Taxonomic Knowledge Graphs **37**

3.1 Background 37

3.2 HiPrompt: Few-Shot Biomedical Knowledge Fusion via Hierarchy-Oriented Prompting . . . 38
3.2.1 Problem Definition . . . 39
3.2.2 HiPrompt . . . 40
3.2.3 Experiments . . . 42
3.3 Open-World Taxonomy and Knowledge Graph Co-Learning . . . 47
3.3.1 Problem Definition . . . 48
3.3.2 TaxoKG-Bench: A Novel Benchmark with Six Datasets for TaxoKG . . . 49
3.3.3 HakeGCN: A Novel Method for Effective TaxoKG Completion 51
3.3.4 Experiments . . . 58

4 Applications of Structured Knowledge 63
4.1 Background . . . 63
4.2 MuGNet: A Sample-Similarity-Based Graph Neural Network for Multimodal Classification . . . 64
4.2.1 Problem Definition . . . 64
4.2.2 MuGNet . . . 65
4.2.3 Experiments . . . 68
4.3 Collaborative Projects on Structured Knowledge for Healthcare Applications . . . 73

5 Conclusion and Future Work 75
5.1 Conclusion . . . 75
5.2 Future Work . . . 76

Bibliography 78

List of Figures

1.1 A roadmap for this dissertation. 2

2.1 Dependency-based Candidate Generation 9

2.2 Dependency-based Graph with extra edges introduced. Dependency, Coreferential, and Adjacent sentence edges are denoted by blue, purple, and red edges, respectively. 10

2.3 Toy examples of concept maps on the topic *"Moon Landing"* generated by different methods. 12

2.4 Overview of proposed *GT-D2G* framework. 15

2.5 Graph Translator. Green rectangles denote RNN cells that take the previous time step chosen node q_{t-1} and generated adjacency vector $\boldsymbol{\theta_{t-1}}$ as input. The RNN state vector $\boldsymbol{h_t}$ is updated at every time step, and is initialized by the graph level representation of initial graph $\boldsymbol{Q_{g_{init}}}$. 18

2.6 Human evaluation results on (a) NYT, (b)AMiner, (c)Yelp based on four proposed metrics. 28

2.7 Concept maps generated by various models for case studies. 31

2.8 Test accuracies by varying the proportions of training data (ranging from 0.1%, 0.25%, ... to 10.00%). 34

2.9 Graph size distributions on different max graph sizes. 35

2.10 Concept maps generated by various models for case studies (*cont.*). . 36

3.1 A toy example of BKF to find entity-term alignment between KG and hierarchy. *Left*: A KG containing biomedical entities. *right*: A hierarchy containing biomedical terms. 39
3.2 Overview of our HiPrompt framework, with a zoom-in on the LLM-based re-ranker. 40
3.3 Case Studies on unlabeled data. Terms highlighted in violet denote the correct alignments for query entities. 46
3.4 Toy examples of existing KBs and TaxoKG. 48
3.5 HakeGCN model architecture. 51
3.6 In-depth analysis for different models. 60
3.7 In-depth analysis for neighbors impact. 60

4.1 Model architecture of MuGNet. 65
4.2 The critical difference diagrams show the mean ranks of each model for the test data of the eight datasets. The lower rank (further to the right) represents the better performance of a model. Groups of models that are not significantly different ($p < 0.05$) are connected by thick lines. 70
4.3 Training duration on all datasets. 71
4.4 Mean testing duration and mean normalized accuracy tradeoffs on all datasets. 71

List of Tables

2.1 Statistics of three datasets. 27

2.2 Correlation coefficients among the five peer annotators with manual responsiveness scores on a total of 300 documents of NYT, AMiner, Yelp (100 each). 29

2.3 Document classification accuracies(%). 31

3.1 Statistics of the KG-Hi-BKF benchmark. 43

3.2 Main experiment results (in percentages). 43

3.3 Retriever with various expansion strategies. 45

3.4 Re-ranker with various LLMs and prompts. 45

3.5 Statistics of the six datasets in TaxoKG-Bench. 50

3.6 TaxoKG completion results in different domains. For abbreviations, C-* indicates metrics for **c**oncept prediction, while R-* indicates metrics for **r**elation prediction. Underlined numbers denote the second runners, while bold numbers denote the winner. 59

3.7 Ablation study results on HakeGCN technical designs. 61

3.8 TaxoKG completion performance when presented with the separated data (SEMedical only or OPIEC only) v.s. the jointed data (SEMedical × OPIEC). 61

4.1 The statistics of the eight datasets in MuG. 68

4.2 Overall experimental results with explicit modality performance. The bold text represents the best performance and the underlined text represents the runner-up performance. 69

List of Algorithms

1 *GT-D2G* Training Algorithm 25

Chapter 1

Introduction

1.1 Motivation

Structured knowledge, in a general sense, refers to organized and well-structured information that is typically represented in a systematic, formal, and readable format. It is designed to capture and represent knowledge about a specific domain, subject, or field in a way that is easy for humans and machines to understand and use.

Structured knowledge is fundamental for both low-level AI models and high-level applications such as data interoperability and integration. Specifically, structured knowledge plays a crucial role in building safe, robust, and responsible AI models in several ways: training data enrichment, data quality assurance, fact validation and verification, and explainable models. Moreover, structured knowledge enables a wide range of knowledge-rich applications, including: semantic webs, factual question-answering systems, recommendation engines, etc.

While numerous of data is generated daily, most of the data remains unstructured. It is urgent to extract structured knowledge from the data in an automatic manner. In this dissertation, I introduce my efforts on the development and implementation of supervision-efficient methods to learn structured knowledge from real-world data.

These methods play a pivotal role in organizing, categorizing, and making sense of unstructured data, ultimately enabling meaningful insights and informed decision-making in data-driven applications.

1.2 Research Roadmap

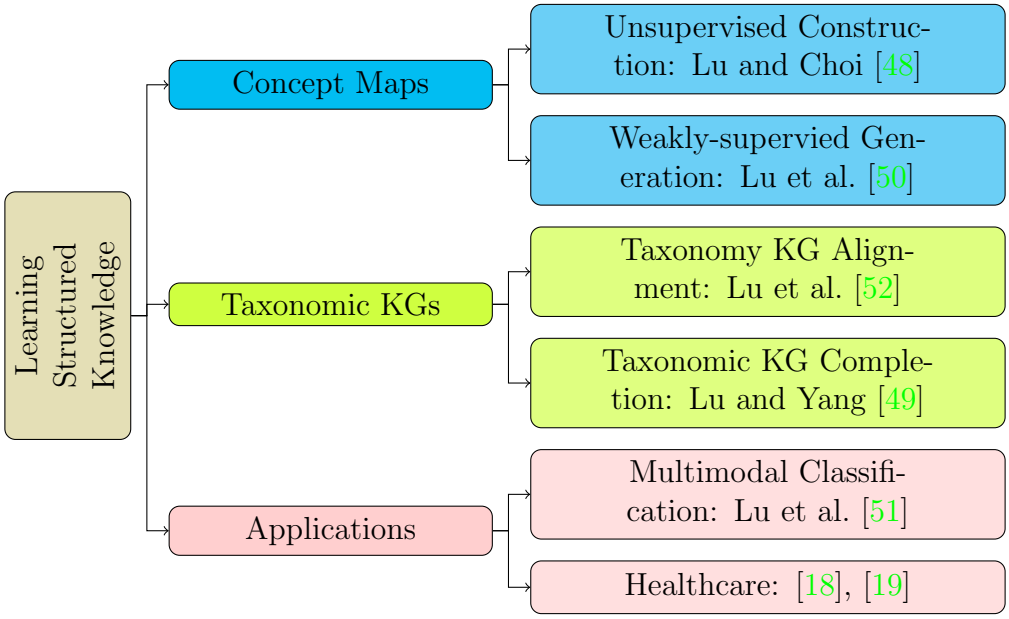


Figure 1.1: A roadmap for this dissertation.

Fig. 1.1 provides an overview of the roadmap for my dissertation. I investigate three aspects of "learning structured knowledge". They are:

Learning to Construct Concept Maps. I have conducted two projects on constructing concept maps without excessive annotation. In "Evaluation of Unsupervised Entity and Event Salience Estimation" [48] (accepted by the FLAIRS'21 as a conference paper), we propose an unsupervised syntactic parsing-based concept map generation algorithm. In "Weakly Supervised Concept Map Generation through Task-Guided Graph Translation" [50] (accepted by IEEE TKDE'23 as a journal paper), we propose a graph translation-based concept map generation framework.

Learning to Align and Complete Taxonomic Knowledge Graphs. I have

conducted two projects on aligning and completing taxonomic KGs. In "HiPrompt: Few-Shot Biomedical Knowledge Fusion via Hierarchy-Oriented Prompting" [52] (SIGIR'23), we explore utilizing few-shot prompting empowered by large language models (LLMs) to tackle the entity alignment between biomedical taxonomies and biomedical KGs. In "Open-World Taxonomy and Knowledge Graph Co-Learning", where we investigate utilizing taxonomies as loosely-defined schema to align open-world KGs and then complete the aligned TaxoKGs.

Applications of Structured Knowledge. I have conducted several projects on applying structured knowledge in various downstream applications. One major project "MuG: A Multimodal Classification Benchmark on Game Data with Tabular, Textual, and Visual Fields." [51] (Findings of EMNLP'23) is about applying a multimodal sample-similarity-based graph approach for multimodal classification task. Moreover, I also collaborate with researchers to explore the utility of structured knowledge in the healthcare domain. For instance, we leverage concept maps to help COVID-19 document retrival [18], and we write a survey on knowledge graphs for healthcare applications [19].

1.3 Dissertation Outline

In my dissertation, chapters are organized as follows.

1. Chapter 1 introduces the motivation for my dissertation, along with the organization of the whole dissertation.

2. Chapter 2 introduces two projects on concept map constructions.

3. Chapter 3 introduces two projects on taxonomic knowledge graph alignment and completion.

4. Chapter 4 introduces projects on utilizing structured knowledge on downstream

applications, along with my collaborative works on structured knowledge for healthcare.

5. Chapter 5 concludes the dissertation.

Chapter 2

Learning to Construct Concept Maps

2.1 Background

Standing out for the clear and concise structured knowledge representation, concept maps have been widely applied in knowledge management [44, 45], document summarization [26, 25], information retrieval [18] and educational science [61, 16]. Fig. 2.3 shows toy examples of concept maps derived from a document describing *"Moon Landing"*, where nodes in the graph indicate important concepts and links reflect interactions among concepts. Although concept maps are helpful in both providing interpretable representations of texts and boosting the performance of downstream tasks, the creation of concept maps is challenging and time-consuming.

In this chapter, two of my first-author papers are included. The first one is "Evaluation of Unsupervised Entity and Event Salience Estimation" [48] (FLAIRS'21), where we propose an unsupervised syntactic parsing-based concept map generation algorithm. The second one is "Weakly Supervised Concept Map Generation through Task-Guided Graph Translation" [50] (IEEE TKDE'23), where we propose a graph

translation-based concept map generation framework. GT-D2G takes our parsing-based concept maps as initial graphs, and translates them into more concise and task-oriented structures.

2.2 Unsupervised Syntactic Parsing-based Concept Map Construction

We propose unsupervised syntactic parsing-based concept map construction in [48]. This work utilizes entity and event salience estimation as the performance-indicating task. Our parsing-based concept map construction is flexible. Depending on the purpose, different variants can be utilized. For instance, in entity and event salience estimation, we explicitly allow noun phrases and verb phrases to be nodes in the concept maps, thus reflecting entities, events, and their interactions. In entity-centric downstream tasks such as document classification, our concept maps can be entity-only graphs, as used in GT-D2G [50].

We refer readers to our manuscript [48] for the details of entity and event salience estimation. In this section, we mainly introduce the syntactic parsing method used for concept map generation.

2.2.1 Preliminaries

Problem Definition

The unsupervised concept map generation task can be defined as follows: Given a text corpus $\mathcal{D}_l = (d_1, \ldots, d_{i_l})$, we aim at generating concept maps $\}_i = \{\mathcal{C}_i, \mathcal{M}_i\}$ for each document d_i.

Entity Definition

Inspired by definitions in previous work, we consider all **base Noun Phrases** (base NPs)[1] as entity candidates, excluding eventive nouns. Although some base NPs, such as *president* or *two weeks*, may refer to multiple real-world objects which violate the rigid definition, this simplification is beneficial to pseudo annotation. Another advantage of using base NPs instead of higher-level NPs is the more fine-grained entity annotation.

Event Definition

An event describes "*who did what to whom when and where*". Therefore, the event trigger (*what*) itself can not represent a complete event. In our definition, event triggers and core arguments are essential components of events. **Verbs**, **Eventive Nouns** (deverbal nouns, proper names referring to historically significant events), phrase constituted by **Light Verb + Noun** and predicative ***Adjectives*** are generally considered as **event triggers** in prior studies. For annotation simplicity, adjectives are not included in our definition. Since not all verbs and nouns are valid event triggers, we create a pre-defined vocabulary using following procedures:

1. We collect the candidate list of verbs and deverbal nouns from FrameNet [2] and NomBank [57] utilizing the same 569 frames generated by [47].

2. We then manually add valid head words (around 47) of proper names such as *epidemic, earthquake*, etc.

3. Similar to previous work, we remove auxiliary and copular verbs[2], light verbs[3], and report verbs[4], as they are rarely representative events.

[1] heuristics for base NP using context-free grammar: `NP->DT` $\bar{N}$`;` $\bar{N}$`->NN;` $\bar{N}$`->NN` $\bar{N}$`;` $\bar{N}$`->JJ` $\bar{N}$`;` $\bar{N}$`->`$\bar{N}$ $\bar{N}$

[2] Auxiliary and copular verbs include *appear, become, do, have, seem, be.*

[3] Light verbs include *do, get, give, go, have, keep, make, put, set, take.*

[4] Report verbs include *argue, claim, say, suggest, tell.*

This gives us a total of 2645 verb lemmas and 516 eventive nouns. Regarding core arguments, we consider entity and sub-event participants specifying *who* or *whom* is involved in the event.

2.2.2 Proposed Approach

Our approach consists of two modules: the *candidate generation module* and the *graph construction module*. The **candidate generation module** parses every sentence in the document d into the dependency tree and extracts *entity spans* $\mathcal{X}_{entity}$, and *event spans* $\mathcal{X}_{event}$ according to the POS taggers and syntactic relations. After all candidates are extracted, the graph $\mathcal{G}$ is then expanded with these spans as new nodes interchangeably denoted as $\mathcal{V}$. Then the **graph construction module** collects these generated entity and event nodes and connects them accordingly. Different from most previous work which considers the graph as a fully-connected graph, edges are added according to the dependency tree arcs which lead the plot graph $\mathcal{G}$ a partially-connected graph. Multiple types of edges including dependency edges, coreferential edges, and inter-sentence edges are added to better reflect semantic and syntactic relations between nodes.

2.2.3 Candidate Generation Module

Figure 2.1 shows an example of entity and event candidate spans generated by a dependency tree. For simplicity, each span in figure 2.1 is represented by the dependent head. The extraction result can be regarded as the remaining dependency tree after removing auxiliary words.

As mentioned before, entities and events have their own syntactic properties:

1) Entities are generally *noun phrases*.

2) Events are *verbal predicates* or *nominal predicates*.

It is then intuitively to derive context-free grammar rules that leverage the POS tagging and dependency relation information to extract entities and events. Since the extraction process is relatively deterministic, the result is reliable. Our implementation uses a BERT-based Dependency Parser from [35] and can be extended using any dependency parser.

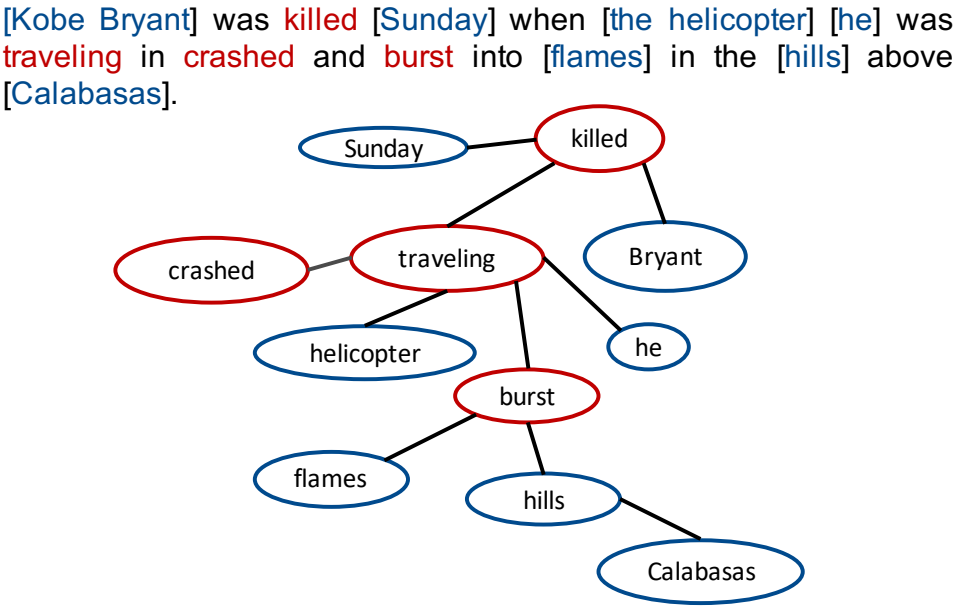


Figure 2.1: Dependency-based Candidate Generation

2.2.4 Graph Construction Module

After all entity and event candidates are extracted, it is natural to construct a graph to capture the relation structure of information. In addition to the intra-sentence dependency relations from dependency parsing, inter-sentence level relations such as adjacent sentence syntactic roots, and coreference resolution are also introduced in our graph construction module. In the following subsections, we will describe the proposed construction algorithm for the graph.

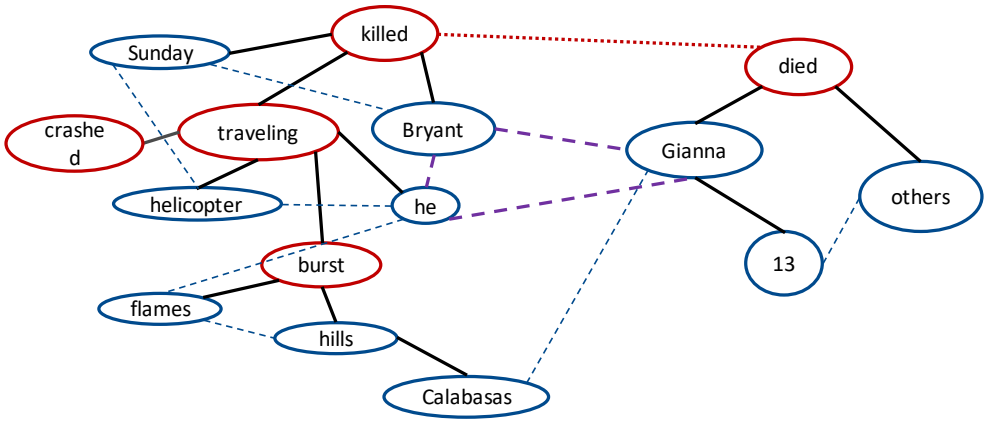


Figure 2.2: Dependency-based Graph with extra edges introduced. Dependency, Coreferential, and Adjacent sentence edges are denoted by blue, purple, and red edges, respectively.

Dependency-based Graph Construction

Figure 2.2 shows a dependency-based graph for two sentences about Kobe Bryant's helicopter crash. Adjacent noun phrase edges, coreferential edges, and adjacent sentence syntactic roots edges are further included in the graph to capture more information. First, the initial dependency-based graph is already established after the candidate extraction phase. The edge w_{ij}^{dep} between two nodes v_i and v_j are weighted according to the sum of the inverse tree distance between each span headword of hw_i and hw_j, as equation 2.1 denoted. The tree distance is defined as the minimum number of edges to be traversed to reach one node from another. It is also worth noting that in dependency-based graph nodes that share the same span have not been merged yet.

$$w_{ij}^{dep} = \frac{1}{tree\_dist(hw_i, hw_j)}, \ \forall e_{ij} \in E_{dep} \tag{2.1}$$

Next, adjacent NPs and syntactic roots are connected accordingly. Edges between Adjacent NPs(E_{NPs}) reflect the information propagation between neighbor entities and contain both intra- and inter-sentence level information. The edge weights w^{NPs} are set by the inverse of tree distances as well. On the other hand, edges between adjacent syntactic roots E_{roots} reflect the information propagation between neighbor

sentences. Since the dependency tree only provides within sentence distances, we can assign a constant to w^{roots} (in practice, we choose 1) as the virtual distance for each edge of E_{roots}. Therefore, equation 2.1 can be expanded to include edges in E_{NPS} and E_{roots}.

Moreover, spans in the same coreferential clusters are connected. The coreferential clusters can be obtained from either the ground truth of the corpus or system-generated result. The edge weights w^{coref} of E_{coref} can also be assigned to a constant (we still use 1 in experiments) because spans in one cluster are equally important. Finally, NP spans consisting of the same tokens within one article $\mathcal{D}$ would then be merged into one node x in the plot graph $\mathcal{G}$, except for single pronoun spans. Therefore,

$$x_i = \{s_{i,j}, ..., s_{i,k}\} \; where \; s_{i,j} = s_{i,k} \tag{2.2}$$

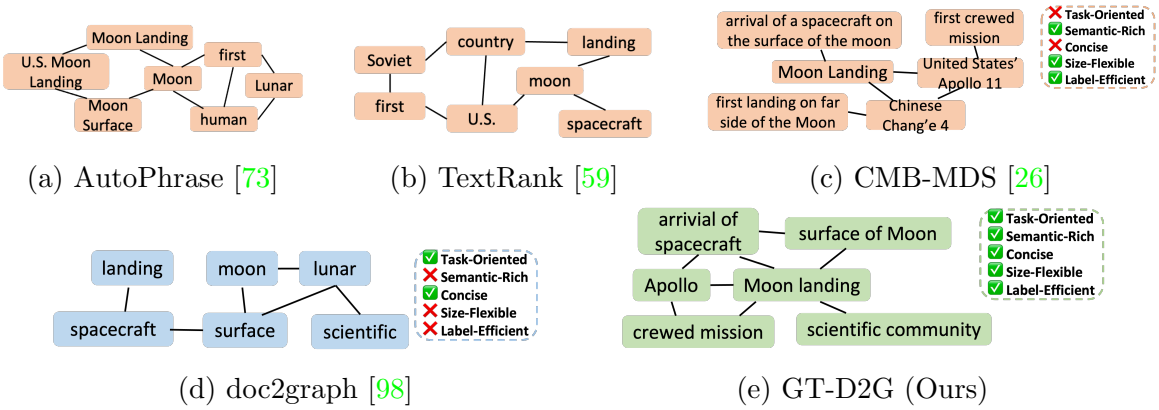
In equation 2.2, $s_{i,j}$ denotes the *jth span* of node x_i. The overall weights between node x_i and x_j then become the sum of different types of weights, which is shown in Equation 2.3. $w_{kl}^{dep}, w_{kl}^{NPs}, w_{kl}^{roots}, w_{kl}^{coref}$ represent edge weights from dependency tree, adjacent Nps, adjacent syntactic roots and coreference resolution clusters between $span_{i,k}$ and $span_{j,l}$, respectively.

$$w_{ij} = \sum_{s_{i,k} \in x_i} \sum_{s_{j,l} \in x_j} \frac{1}{w_{kl}^{dep} + w_{kl}^{NPs} + w_{kl}^{roots} + w_{kl}^{coref}} \tag{2.3}$$

After the edge weights are set, classic unsupervised graph ranking such as TextRank [58] algorithms can be directly employed. The importance of nodes in the graph is calculated and then used as indicators of whether entities or nodes are salient.

2.3 Weakly Supervised Concept Map Generation through Task-Guided Graph Translation

We propose GT-D2G (Graph Translation-based Document to Graph) [50], an automatic concept map generation framework that leverages syntactic parsing-based pipeline [48] proposed in Ch. 2.2 to derive semantic-rich initial graphs, and translates them into more concise structures under the weak supervision of downstream task labels. The concept maps generated by *GT-D2G* can provide interpretable summarization of structured knowledge for the input texts.



(a) AutoPhrase [73] (b) TextRank [59] (c) CMB-MDS [26]

(d) doc2graph [98] (e) GT-D2G (Ours)

Figure 2.3: Toy examples of concept maps on the topic *"Moon Landing"* generated by different methods.

Traditionally, concept map generation follows a multi-step pipeline including concept extraction, relation identification, and graph assembling [26, 1, 37], where auxiliary resources and carefully designed heuristics are often required. However, the separation of concept map construction and downstream tasks easily deviates the generated graphs from what the real task needs. For example, Figures 2.3a, 2.3b, 2.3c provide examples of concept maps constructed from such unsupervised ad hoc processes. Although the sample document has the label of *science*, the extracted concepts of "U.S. Moon Landing" (2.3a), "Soviet" (2.3b) and "Chinese Chang'e 4" (2.3c) are more related to the label of *politics*. As a consequence, these deviating

concepts will likely degrade the performance of document classification. Moreover, nodes chosen by these traditional methods often lack conciseness due to their heavy reliance on ad hoc pipelines. For instance, in Fig. 2.3a, the concept map contains redundant concepts such as "Moon" and "Moon Surface" as concepts mined by *AutoPhrase* are mainly based on frequency features; while in Fig. 2.3c, the concepts are rather verbose due to the OpenIE component for concept generation in *CMB-MDS*.

On the other hand, research efforts have been made to automatically generate concept maps from documents under weak supervision from text-related downstream tasks. *Doc2graph* [98] is one pioneering study that achieves this goal through a fully end-to-end neural network model. However, due to the lack of linguistic analysis, the generated concepts often suffer from semantic incompleteness and the links between concepts are often noisy. For example in Fig. 2.3d, one compound concept *"moon landing"* is preferable to two separated concepts *"landing"* and *"moon"* as the former carries more precise and complete semantic information. Moreover, while the weakly supervised training diagram enables *doc2graph* to generate concept maps at scale, we observe the downside of being not label-efficient. In other words, *doc2graph* is sensitive to training signals and it requires a significant amount of weak supervision to construct meaningful concept maps. Finally, the size of concept maps generated by *doc2graph* is fixed due to its rigid technical design, while the ideal size of graphs should vary according to the complexity of the documents being represented.

Inspired by both existing methods, we propose a graph translation-based neural concept map generation framework that simultaneously leverages existing NLP pipelines and receives weak supervision from downstream tasks, dubbed as **GT-D2G** (Graph Translation-based Document To Graph). The integration of NLP pipelines effectively assists *GT-D2G* in addressing the semantic incompleteness issue of *doc2graph* by introducing both words and phrases as concept candidates. Meanwhile, the initial semantic-rich graphs constructed by the NLP pipeline bring in *a priori* knowledge

from the linguistic side, thus alleviating the label inefficient issue of *doc2graph*. In *GT-D2G*, concepts and their interactions are generated iteratively through a sequence of nodes and adjacency vectors, which ensures deeper coupling between nodes and links for more meaningful results and resolves the fixed size issue of *doc2graph*. On the other hand, guided by the weak supervision from downstream tasks, *GT-D2G* is also able to generate task-oriented concept maps that provide preferable support to specific downstream tasks, while eliminating the redundancy issue of traditional unsupervised methods, specifically through the incorporation of a penalty over content coverage. To sum up, concept maps generated by our proposed *GT-D2G* method are task-oriented, semantic-rich, concise, size-flexible, and label-efficient, as illustrated in Fig. 2.3e.

In this work, an extensive suite of experiments has been conducted on text corpora from three domains: news, scientific papers, and customer reviews. Through experiments on the downstream task of document classification, we demonstrate that the proposed *GT-D2G* framework outperforms both traditional concept map generation baselines and the state-of-the-art neural method *doc2graph*, while a comprehensive ablation study shows the effectiveness of each of our novel designs. The quality and interpretability of generated graphs are supported by rigorous human evaluation and rich case studies. Finally, we specifically validate the labeling efficiency of *GT-D2G* in the label-efficient learning settings and the flexibility of generated graph sizes in controlled hyper-parameter studies.

2.3.1 Problems Definition

We focus on the novel problem of weakly supervised concept map generation. It can be defined as follows: Given a text corpus $\mathcal{D}_l = (d_1, \ldots, d_{i_l})$ with corresponding labels $\mathcal{Y} = (y_1, \ldots, y_{i_l})$ of certain downstream text-related tasks, we aim at generating concept maps $g_i = \{\mathcal{C}_i, \mathcal{M}_i\}$ for each document $d_i \in D_u$ where D_u is a set of

unlabeled documents. As can be seen from the definition, there are no ground-truth concept maps paired with the input text. Instead, weak or distant supervision from downstream tasks is provided. The downstream text-related tasks are very flexible, possibly ranging from document classification, retrieval, ranking, relation inference, *etc.* The major output is concept maps $\mathcal{G}$ for all documents $\mathcal{D} = \mathcal{D}_l \cup \mathcal{D}_u$. A document $d \in \mathcal{D}$ is indeed a sequence of words, *i.e.*, $d_i = (w_{i,1}, \ldots, w_{i,|d_i|})$. A concept map $g_i = \{\mathcal{C}_i, \mathcal{M}_i\}$ is an undirected graph which focuses on the concepts $\mathcal{C}_i$ and their interactions $\mathcal{M}_i$ in the span of d_i. $\mathcal{C}_i = (c_{i,1}, \ldots, c_{i,|c_i|})$ is a set of n concepts that can be words, phrases, or sentence fragments depending on the downstream tasks, and $\mathcal{M}_i \subseteq \mathbb{R}^{n \times n}$ indicates the interaction strength (*i.e.*, edge weight) among concepts in $\mathcal{C}_i$. Moreover, the auxiliary output is the predicted labels $\hat{\mathcal{Y}}$ for unlabeled documents $\mathcal{D}_u$.

2.3.2 *GT-D2G*

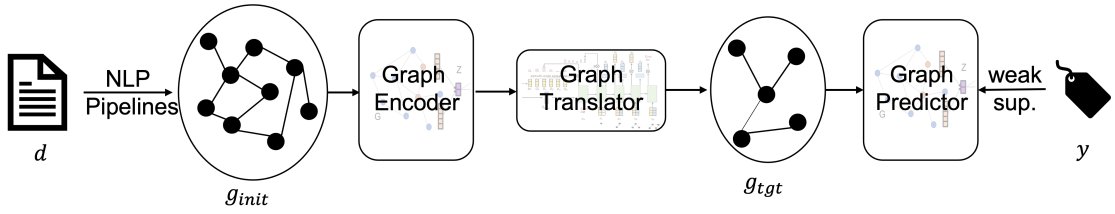


Figure 2.4: Overview of proposed *GT-D2G* framework.

Fig. 2.4 gives an overview of the proposed *GT-D2G* (Graph Translation based Document-To-Graph) framework: A proper NLP pipeline is used to extract salient phrases from document d and construct the initial semantic-rich concept map g_{init}. A Graph Encoder then encodes each node of g_{init} into a node-level embedding $\boldsymbol{Q_i}$, and also represents the whole g_{init} as a dense vector by aggregating all its node embeddings. A Graph Translator is responsible for identifying the nodes needed to be kept in the target graph g_{tgt} as well as proposing links among kept nodes iteratively. Once the nodes and links are generated, the target graph g_{tgt} is fed into

a Graph Predictor to produce a document label $\hat{y}$, which can be trained towards the ground-truth label y. The whole encoder-translator-predictor neural network is thus weakly supervised by the classification signal in an end-to-end fashion. In the following subsections, we expand with more technical details.

Enriching Concept Maps with Semantics

As we motivated before, one major drawback of doc2graph [98] is that single words are directly picked from the raw texts through a Pointer Network [84] and considered as nodes in the final concept map. However, words purely picked by a simple Pointer Network can easily be of low-quality [90]. Moreover, phrases are often preferable to represent concepts, especially noun phrases as semantically complete concepts [73]. For instance, extracting two nodes "deep", "learning" from a computer science paper is incomplete while "deep learning" as one concept node is semantically more meaningful and accurate. Some researchers propose to concatenate words that occur adjacently in the input document as extracted phrases to solve this issue, although potential heuristic post-processing is needed. In *GT-D2G*, we aim to enrich concept maps with semantics by leveraging our own syntactic parsing-based pipelines [48]. For simplicity and generalization concerns, we intentionally choose the most popular yet reliable NLP tools for initial concept map construction, which can be further extended according to application scenarios.

Node Generation. To avoid complicated pre-processing, we use multiple classic NLP tools in *GT-D2G* to extract noun phrases, verb phrases, and adjectives as node candidates in the initial concept map. Sentence segmentation, pos-tagging, lemmatization, and constituency parsing are conducted for every document. Since constituency parsing detects sub-phrases of given sentences, we then first extract basic noun phrases from constituency parsing results. The basic noun phrases extraction algorithm is deterministic so that any noun phrase not containing other noun phrases

is considered valid. After all basic noun phrases are identified, verb phrases and adjectives remaining in the text are extracted. Other discourse units such as adverbs and prepositions are discarded since they typically do not contain much knowledge or information. Due to the fact that multiple words can refer to the same concept, determinants such as "a", "an", "the" are removed from the node mentions, and words are replaced by their lemmas. Moreover, pronouns need to be merged into coreferent mentions to obtain a clean initial concept map. Thus, the coreference resolution technique is used to resolve all pronoun expressions in documents. We use the popular Stanford CoreNLP [54] for all the steps mentioned above.

Link Generation. For links between extracted nodes, we follow the sliding window idea introduced in keyphrase extraction studies [59]. Nodes that occur within a fixed-sized sliding window are connected to each other. Therefore, the initial concept maps are undirected graphs $g_{init} = \{C_{init}, M_{init}\}$. The link construction module is flexible in *GT-D2G* so that any algorithms can be applied to construct weighted links or directed links. For instance, we can directly use the whole parsing tree or filter out certain types of relations for link generation. The graph ensemble process is trivial once nodes and links are extracted.

Task Guided Graph Translation

Graph Encoder. Before graph translation, the model has to first learn to understand the initial graph. For this purpose, we adopt the recent successful graph representation learning model, *i.e.*, Graph Convolutional Network (GCN) [40] as our Encoder. The node embeddings $\boldsymbol{Q^{(k)}}$ are learned after the k-th layer of GCN by the following equation

$$\boldsymbol{Q^{(k)}} = \mathrm{ReLU}(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{M}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{Q^{(k-1)}}\boldsymbol{W_Q^{(k)}}), \tag{2.4}$$

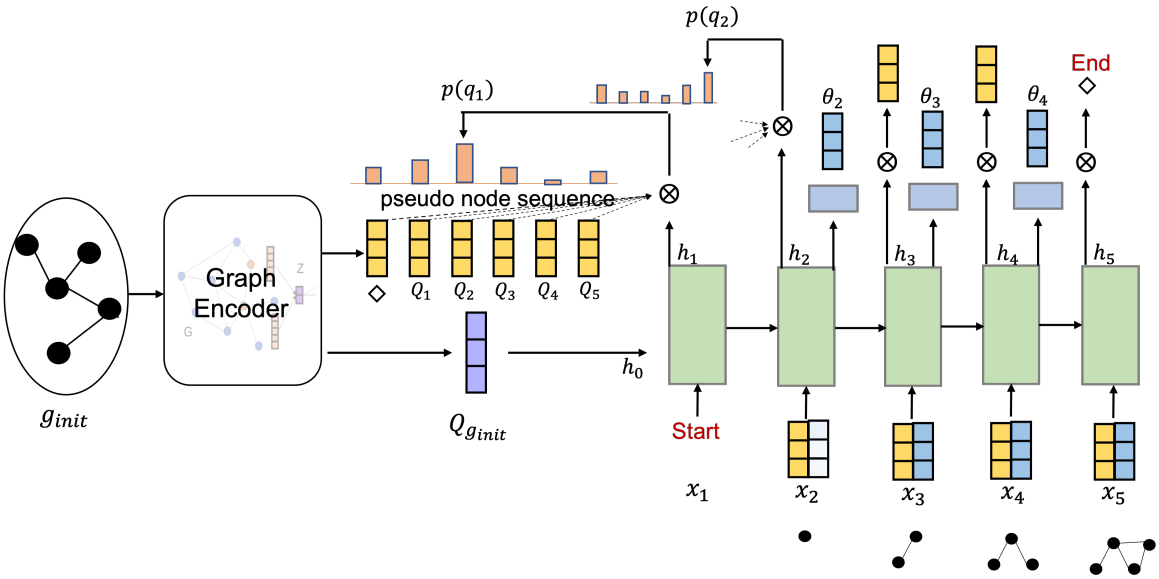


Figure 2.5: Graph Translator. Green rectangles denote RNN cells that take the previous time step chosen node q_{t-1} and generated adjacency vector $\boldsymbol{\theta_{t-1}}$ as input. The RNN state vector $\boldsymbol{h_t}$ is updated at every time step, and is initialized by the graph level representation of initial graph $\boldsymbol{Q_{g_{init}}}$.

where $\boldsymbol{W_Q^{(k)}}$ is learnable parameters in the k-th layer of GCN, $\tilde{\boldsymbol{M}} \subseteq \mathbb{R}^{n \times n}$ is the adjacency matrix $\boldsymbol{M}_{init}$ with additional self-connections and $\tilde{\boldsymbol{D}}_{ii} = \sum_j \tilde{\boldsymbol{M}}_{ij}$ is the diagonal degree matrix. The input node embeddings $\boldsymbol{Q^{(0)}}$ are the concatenations of phrase embeddings, normalized frequency feature, and normalized location feature. The phrase embedding of each node is the average of pre-trained word embeddings (in practice, we use GloVe [64]). The frequency feature and the location feature reflect the importance of the concept in the original text and are normalized by min-max scaling per graph. Besides the node-level embeddings, we also compute the graph-level embedding as $\boldsymbol{Q_{g_{init}}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{Q_i^k}$ to encode the global contextual information in the initial graph.

Graph Translator. Our graph translator aims to choose the most informative nodes that are also beneficial to downstream tasks from the initial graph, while proposing links among the chosen nodes accordingly. In particular, the Graph Translator generates a sequence of nodes and their corresponding adjacency vectors based on the

initial concept map g_{init}– to be specific, its node-level embeddings $\boldsymbol{Q_i}$ ($i \in [1, n]$) and graph-level embedding $\boldsymbol{Q_{g_{init}}}$ produced by the Graph Encoder. Since we expect to preserve the semantic rich and task-relevant concepts in the initial graph and only pick out a subset of nodes, we adopt the Pointer Network [84] from keyword selection and novelly extend it into a graph version to generate a sequence of pointers for the selection of the most important nodes from the initial concept map. After each node is selected, we get inspiration from GraphRNN [101] to also generate its corresponding adjacency vector which contains links to previously selected nodes. However, the original GraphRNN only works on the transductive learning setting when there is an actual graph as input to learning from. Therefore, we need to make several novel modifications to GraphRNN before seamlessly integrating it into our Graph Pointer Network (GPT) towards our novel setting of task-guided graph translation.

Graph Pointer Network. Since the original Pointer Network [84] works on sequential text data, we convert the non-sequential nodes in the initial concept map into a pseudo node sequence according to positions of node mentions in the source document, illustrated as the yellow bars in Fig. 2.5. The order of pseudo node sequence is flexible and can be replaced with any other order for proper reasons (e.g., node degree order). Here we just follow the most intuitive way and do not observe significant performance differences when using other orders. In our GPT, we use a one-directional RNN decoder to model the process of translating a sequence of nodes and links from an initial graph, denoted as the green rectangles in Fig. 2.5. In practice, we choose GRU [17] as the implementation. In order to start the translation from the whole initial graph, the hidden state of the RNN decoder is initialized by $\boldsymbol{h_0} = \boldsymbol{Q_{g_{init}}}$, and the input of the first step is $\boldsymbol{x_1} = (0, \ldots, 0)^\intercal$. Therefore, the hidden state that encodes the "graph translation state" is updated by

$$\boldsymbol{h_t} = \mathrm{RNN}(\boldsymbol{x_t}, \boldsymbol{h_{t-1}}), \tag{2.5}$$

where $\boldsymbol{h_{t-1}}$ denotes the hidden state from the last time step, and $\boldsymbol{x_t}$ denotes the input at the current time step. More specifically, we compute $\boldsymbol{x_t}$ as the representations of both nodes and links generated from the last time step, which can be denoted as

$$\boldsymbol{x_t} = [\boldsymbol{q_{t-1}}; \boldsymbol{\theta_{t-1}}], \tag{2.6}$$

where $[\cdot;\cdot]$ denotes vector concatenation. $\boldsymbol{q_{t-1}} = \boldsymbol{Q_i}$ is the node embedding from the Graph Encoder of the last selected node i, and we defer the explanation towards adjacency vector $\boldsymbol{\theta_{t-1}}$ to the later part of this subsection.

Deeply coupled node and link generation. Once we obtain the RNN decoder hidden state $\boldsymbol{h_t}$, the node selection process can be described by the following equations

$$e_{i,t} = \boldsymbol{v}^\intercal \tanh(\boldsymbol{W}[\boldsymbol{Q_i}; \boldsymbol{h_t}]), \tag{2.7}$$

$$\mathsf{p}_{t,i} = \mathsf{p}(\boldsymbol{q_t} = \boldsymbol{Q_i}) = \frac{\exp(e_{i,t})}{\sum_{j=1}^{n} \exp(e_{j,t})}, \tag{2.8}$$

where $\boldsymbol{v} \subseteq \mathbb{R}^{d_e}$ and $\boldsymbol{W} \subseteq \mathbb{R}^{d_h \times d_e}$ are learnable parameters for calculating the unnormalized node selection score $e_{\cdot,t}$ at time step t for every node in initial graph node set C_{init}. Our GPT then selects the i-th node with the maximum score by

$$i = \arg\max_{i}(\mathsf{p}_{t,i}), \tag{2.9}$$

adds the selected node into the translated target graph and feeds $\boldsymbol{q_t} = \boldsymbol{Q_i}$ into the RNN decoder at the next time step. To improve the semantic completeness of selected concept nodes, we also adapt the coverage loss in [81], by maintaining a coverage vector $\boldsymbol{c_t} = \sum_{t'=0}^{t-1} \mathsf{p}(\boldsymbol{q_{t'}})$ that accumulates the generated attention so far, while adding the following loss to enforce the model to pay more attention to nodes not covered yet:

$$L_{cov} = \sum_{d_i \in \mathcal{D}} \sum_{t_j \in d_i} \min(\mathsf{p}(\boldsymbol{q}_{\boldsymbol{t'}}), \boldsymbol{c}_{\boldsymbol{t}_j}). \tag{2.10}$$

To deeply couple the generation process of nodes and links so that the target graph (*i.e.*, final concept map) is meaningful, we get inspired by the recent deep graph generation model of GraphRNN [101]. Specifically, in our GPT, at each time step, after a new node is generated, we immediately generate its associated adjacency vector regarding all links between it and all previously generated nodes, as denoted by smaller blue rectangles in Fig. 2.5 and described in the following equation

$$\boldsymbol{\theta}_{\boldsymbol{t}} = f_{out}(\boldsymbol{h}_{\boldsymbol{t}}), \tag{2.11}$$

where $\boldsymbol{\theta}_{\boldsymbol{t}}$ is the length $t-1$ adjacency vector for the chosen node at time step t that is output by f_{out}. Based on slightly different goals for link generation, we design two variants of f_{out}: the *path* variant and the *neigh* variant. The former models the adjacency vector generation as generating a path connecting some previously picked nodes to the currently picked one, focusing on the higher-order sequential information among concepts. Hence, f_{out}^{path} is implemented as another RNN that connects to the hidden state of the RNN decoder. On the other hand, the *neigh* variant interprets the generation problem as generating all possible neighbors of the currently picked node from all previously picked nodes, focusing on the first-order neighborhood structures of concepts. Therefore, f_{out}^{neigh} is implemented as a multi-layer perceptron (MLP) with non-linear activation. The weights of f_{out} are shared across all time steps to reduce the number of parameters and alleviate overfitting. In our experiments, we find the *neigh* variant to be preferable over the *path* variant, which can be intuitively attributed to the fact that structural information is more important than sequential information among concepts.

Graph Predictor. After generating a sequence of nodes $q_1, \ldots, q_T$ and adjacency

vectors $\theta_1, \ldots, \theta_T$, we assemble the target graph as

$$g_{tgt} = \{C_{tgt}, M_{tgt}\} = \{(q_1, \ldots, q_T), (\theta_1, \ldots, \theta_T)\}. \tag{2.12}$$

For the downstream graph-level prediction, we adopt a *Graph Isomorphism Network* (GIN) [96] due to GIN's superior discriminative power to capture different graph structures. More specific, we adopt the sum operator as the neighborhood aggregation function, and an MLP as the center node and neighbor nodes combination function:

$$\boldsymbol{q^{(k)}} = \text{ReLU}((\boldsymbol{M_{tgt}} + (1 + \epsilon^{(k)})\boldsymbol{I})\boldsymbol{q^{(k-1)}}\boldsymbol{W_q^{(k)}}), \tag{2.13}$$

where $\boldsymbol{M_{tgt}} \subseteq \mathbb{R}^{T \times T}$ is the adjacency matrix of translated concept map, $\boldsymbol{I} \subseteq \mathbb{R}^{T \times T}$ is a identity matrix (*i.e.*, self connection), and $\epsilon^{(k)}, \boldsymbol{W_q^{(k)}}$ are learnable parameters for GIN's k-th layer. Furthermore, the graph label (*i.e.*, document category in our case) $\hat{y}$ is obtained by an additional two-layer MLP on the graph representation:

$$\hat{y} = \text{MLP}(\text{concat}(\text{sum}(\boldsymbol{q^{(k)}})|k = 1, \ldots, K)), \tag{2.14}$$

where the graph representation is achieved by summing all node embeddings from the same layer, and then concatenating summed embeddings across all layers.

Training Techniques

The whole model is trained in a weakly supervised end-to-end fashion, by computing the cross-entropy loss for the downstream task– document classification as we focus on in this work, and the coverage loss for the node selection in our GPT. Specifically,

we have

$$L_{cls} = -\sum_{d_i \in \mathcal{D}} \mathsf{p}(\hat{y}_i) \log \mathsf{p}(y_i), \tag{2.15}$$

$$L = L_{cls} + \lambda * L_{cov}, \tag{2.16}$$

where λ is a tunable hyper-parameter.

One technical challenge exists for the node selection operation that selects the node with maximum pointer attention $i = \arg\max_i(\mathsf{p}(\boldsymbol{q_t} = \boldsymbol{Q_i}))$ during the graph translation process in GPT. Firstly, the max value selection operation implemented as *argmax* is non-differentiable, thus leading to the lost gradient after node selection. Secondly, *argmax* is a deterministic sampling operator, thus making the GPT loses exploration ability. The exploration ability or stochastic sampling is important during the early training stages of GPT, because the predicted probability to select a node is not very reliable at that time. Inspired by the re-parameterization tricks for categorical variables sampling [4, 53, 38], we adopt a hard-version *Gumbel-Softmax* to sample one-hot vectors from the predicted probabilities, so that the node selection process in GPT is differentiable and stochastic. The sampled probability $P_{t,i}$ to choose node i at time step t then becomes:

$$P_{t,i} = \text{softmax}(\log(\mathsf{p}_{t,i}) + G_i, \tau), \tag{2.17}$$

where $\mathsf{p}_{t,i}$ is the predicted probability as defined in Eq (2.8), $G_i \sim \text{Gumbel}(0, 1)$ is the i-th random variable sampled from the Gumbel distribution, and τ is the temperature parameter for *softmax*. We set a relative large temperature to enforce $P_t = (P_{t,1}, P_{t,2}, \ldots, P_{t,n})$ has the one-hot vector shape. During training, we use $P_t \cdot Q$ to represent selecting one particular node for gradient backpropagation.

Moreover, to generate concept maps of flexible sizes, we incorporate the special

"EOS" node at the first position of pseudo node sequence, denoted as "◊" in Fig. 2.5. The end of an output node sequence is determined when the *"EOS"* is predicted. For the completeness of concept maps, we penalize node sequences that are too short, which can be implemented by applying a penalty to *"EOS"* node predicted at every time step as follows

$$L_{len} = \sum_{d_i \in \mathcal{D}} \sum_{t_j \in d_i} \text{Penalty}(t_j) \cdot \mathsf{p}(\boldsymbol{q_{t_j}} = \text{"}EOS\text{"}). \tag{2.18}$$

The function $\text{Penalty}(t) > 0$ defines a penalty curve depending on the current time step t. In our implementation, we choose the RBF kernel function $\Phi(t, t') = exp(-\frac{\|t-t'\|^2}{2\sigma^2})$ for the penalty curve [15]. Therefore, the overall loss function for *GT-D2G* is:

$$L = L_{cls} + \lambda_1 * L_{cov} + \lambda_2 * L_{len}. \tag{2.19}$$

To sum up, our whole framework is trained in an end-to-end fashion, while Graph Encoder, Graph Translator, and Graph Predictor are guided by the downstream task with the goal of reducing classification loss. In this way, each module is jointly learned and enhanced. Moreover, the translation process is regularized by the coverage loss and graph size loss, aiming to produce high-quality concept maps depending on the input documents' characteristics.

Complexity Analysis

To analyze the computational efficiency of the proposed model, we present the *GT-D2G* training algorithm for one input initial concept map (one input document). The actual implementation is based on mini-batch training, and is publicly available[5]. For obtaining graph representation of the initial concept map (L3-L4), the time complexity is $\mathcal{O}(Knd^2 + Kmd)$, where K is the number of GCN encoder layers, d is the

[5] *GT-D2G*: https://github.com/lujiaying/GT-doc2graph

Data: initial concept map $g_{init} = \{\mathcal{C}_{init}, \mathcal{M}_{init}\}$, input node embeddings $\{Q_v^0, \forall v \in \mathcal{C}_{init}\}$, ground truth graph label y
Result: translated concept map $g_{tgt} = \{\mathcal{C}_{tgt}, \mathcal{M}_{tgt}\}$, predicted graph label $\hat{y}$

Initialize GT-D2G parameters;
while *not converge* **do**
  `/* Obtain graph representation of` g_{init} `*/`
  Update node embedding $\boldsymbol{Q^{(k)}} = \mathrm{GCN}_{\mathrm{Enc}}(\boldsymbol{Q^{(0)}}; k)$ by Eq. 2.4;
  Update graph embedding $\boldsymbol{Q_{g_{init}}} = \mathrm{pooling}(\{\boldsymbol{Q_v^{(k)}}, \forall v \in \mathcal{C}_{init}\})$;
  `/* Translate` g_{init} `into` g_{tgt} `step-by-step */`
  while *not generate "EOS" node* **do**
    Prepare Graph Translator (RNN) input $(\boldsymbol{x_t}, \boldsymbol{h_{t-1}})$ by initilization or previous step results;
    Update hidden state of Graph Translator $\boldsymbol{h_t}$ by Eq. (2.5);
    Generate node q_t by Eq. (2.7), (2.8), (2.9);
    Generate adjacency vector θ_t by Eq (2.11);
  end
  `/* Predict graph label */`
  Assemble the translated concept map g_{tgt} by Eq. (2.12);
  Predict the graph label $\hat{y}$ by Eq. 2.13, 2.14;
  `/* Backpropagate the weak supervision */`
  Compute the overall loss L by Eq. (2.19);
  Update model parameters with the gradients of L.
end

Algorithm 1: *GT-D2G* Training Algorithm

embedding dimensions (128 in all layers), n is the number of nodes in g_{init} (tens of nodes in our experiments), m is the number of edges in g_{init} (*e.g.*, close to one hundred edges in our experiments). The time complexity can be further simplified into $\mathcal{O}(Knd^2)$ since $nd \gg m$. For graph translation (L5-L9), the time complexity is $\mathcal{O}(TKnd^2)$, where T is the size of the translated concept map, K is reused to represent the number of RNN decoder layers (*e.g.*, we set both GCN encoder, RNN decoder and GIN classifier layer sizes as 2), d is reused to represent the RNN embedding dimensions (*e.g.*, we set the hidden dimension to 128 for all modules). For the graph label prediction(L10-L11), the time complexity is $\mathcal{O}(KTd^2)$ which is similar to GCN encoder analysis. Therefore, the overall time complexity for proposed *GT-D2G* is $\mathcal{O}(Knd^2 + TKnd^2 + KTd^2) = \mathcal{O}(TKnd^2)$.

It is worth noting that the construction of initial concept maps is quite efficient, as the toolkit we employed (*e.g.*, JVM-based Stanford CoreNLP [54]) mainly utilize pre-trained models or rule-based annotators for the NLP pipelines. Moreover, *doc2graph*'s time complexity is $\mathcal{O}(TK\|\mathcal{D}\|d^2)$, where $\|\mathcal{D}\|$ denotes the number of words of input document. *GT-D2G* is more efficient than *doc2graph*, due to the fact that $\|\mathcal{D}\| \geq n$ in most cases. However, the advantage of *doc2graph* is that it does not require NLP pipelines to derive the initial concept maps.

2.3.3 Experiments

In this section, we evaluate our proposed *GT-D2G* framework focusing on the following four research questions:

RQ1: How is the quality of *GT-D2G* generated graphs?

RQ2: How do *GT-D2G* and its variants perform in comparison to other document classification methods?

RQ3: Is *GT-D2G* label efficient?

RQ4: Can *GT-D2G* generate flexible sizes of concept maps?

Experiment Settings

Datasets. Our experiments are conducted on three real-world text corpora [98]: *NYT*, *AMiner*, and *Yelp*. Different from [98], for the *Yelp* dataset, we re-grouped the 1-5 star reviews into negative, neutral and positive ratings. The statistics of the three datasets are listed in Table 2.1. For standard document classification, we follow the setting in [98] to randomly split the labeled documents into 80% for training, 10% for validation, and 10% for testing. We choose accuracy as the metric for document classification tasks. To get a stable result, we run each model three times and report the mean ± standard deviation.

Compared Methods. We compare *GT-D2G* with two sets of baselines described

Table 2.1: Statistics of three datasets.

Dataset	#doc	#word	#category	Init Concept Map		
				#node	#edge	#degree
NYT	13,081	88.64	5	34	84	4.9
Aminer	21,688	87.27	6	34	81	4.8
Yelp	25,357	71.59	3	28	76	5.4

as follows:

Graph-Based Methods as major competitors.

- ***AutoPhrase*** [73]: This is a Pos-Guided Phrasal Segmentation model for phrase mining. We use the top-n highest quality phrases mined from input text as concepts and connect concepts in same sentence. The edge weights is computed as $w_{ij} = 1 - e^{-c_{ij}}$, where c_{ij} denotes sentence-level co-occurring times of concept i and j.

- ***TextRank*** [59]: A word co-occurrence graph is first constructed using a sliding window that connects any two words within the window. We use words with top-n maximum PageRank values as concepts. The edge weights are computed in the same way as *AutoPhrase*.

- ***CMB-MDS*** [26]: We use its pipeline to construct concept map and filter out concepts with low importance scores to keep top-n concepts. The edge weights are set to 1 according to the *CMB-MDS* implementation.

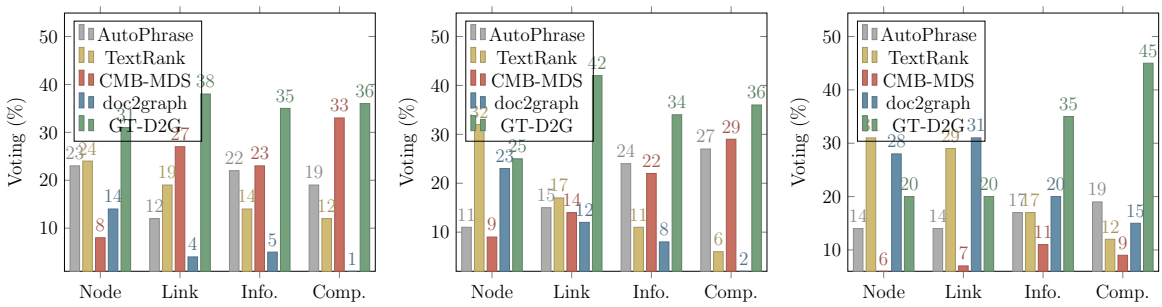
- ***doc2graph*** [98]: *doc2graph* is a neural concept map generation model that is capable of generating concept maps through distant document classification supervision. We follow their implementation to pre-define graph size as n.

Text-Based Methods as performance benchmarks.

- ***Bi-LSTM*** [31]: *Bi-LSTM* is a commonly used RNN model in text classification that learns the long-term dependencies in the document. We train

Bi-LSTM on the training set using the output from last time-step to predict document categories.

- ***BERT-base*** [21]: *BERT* has achieved excellent performance on a wide range of NLP tasks as a state-of-the-art language model. In our experiment, We fine-tune the pre-trained *BERT-base* model on the classification task.



(a) NYT (b) AMiner (c) Yelp

Figure 2.6: Human evaluation results on (a) NYT, (b)AMiner, (c)Yelp based on four proposed metrics.

Implementation Details. We implement *GT-D2G* using Pytorch [62] and DGL [87], with code publicly available[5]. Implementations of the compared baselines are either from open-source project (*BERT*[6]) or the original authors (*Bi-LSTM*/ *AutoPharse*/ *TextRank*/ *CMB-MDS*/ *doc2graph*[7]). We optimize *GT-D2G* through the Adam optimizer with learning rate to $3e-4$ and max epoch to 500. The temperature parameter τ for Gumbel-softmax starts from a big number (*e.g.* 3 or 5) and then anneals along with training epochs to encourage exploration on the later stage. To get a higher accuracy, we set batch size to 64 for training. The hidden layer dimension of GCN, RNN and MLP are set to 128, and the number of GNN layers in all GCN, GIN models are 2. For RBF kernel function used to penalize overlength node sequence, σ and t_{prime} are set to 4 and 0, respectively. We choose GRU for RNN

[6] *BERT*: https://github.com/huggingface/transformers
[7] *doc2graph*: https://github.com/JieyuZ2/doc2graph

used in generating nodes and edges for simplicity sake. All other hyper-parameters are tuned separately on the validation set.

Human Evaluation (*RQ1*)

Table 2.2: Correlation coefficients among the five peer annotators with manual responsiveness scores on a total of 300 documents of NYT, AMiner, Yelp (100 each).

Peer Scoring	Node	Link	Info.	Comp.
NYT	0.50	0.89	0.57	0.67
AMiner	0.76	0.80	0.75	0.93
Yelp	0.73	0.79	0.70	0.92

Human evaluation is critical to answer *RQ1*, *i.e.* evaluating the quality of generated concept maps, since there are no ground-truth concept maps on the three document classification datasets. Five expert annotators are hired to evaluate graphs generated from the text data by five methods: *AutoPhrase*, *TextRank*, *CMB-MDS*, *doc2graph*, and *GT-D2G*. More specifically, on each dataset, we randomly sample 100 document with associated graphs of each method. or each document, annotators are asked to rank the five concept maps in terms of four metrics:

Node: regardless of downstream tasks, whether nodes are semantic complete, in proper length and not redundant.

Link: whether links between nodes are consistent with the text and make sense.

Informativeness: whether the generated graph is helpful for the downstream task.

Completeness: whether the generated graph covers the most salient information of the original text from different aspects.

Correlation Coefficient is a widely used indicator to estimate the inter-annotator agreement (ITA). However, we observe that explicitly annotating the rank among all five concept maps leads to low inter-annotator agreement. Therefore, we allow annotators to pick k ($k \leq 3$) graphs for each metric as top graphs, as long as they think these k graphs are of the same best quality. That means, if an annotator thinks

two graphs by *doc2graph* and *GT-D2G* are competitive in Informativeness, she can mark both two as top graphs without distinguishing which is the best. The top max-k graph annotation guideline gives high Correlation Coefficient scores, as can be seen in Table 2.2.

The human evaluation results are shown in Fig. 2.6. The value on y-axis indicates the percentage of the data that the annotator think the method performs best under the corresponding metric. For the metrics of *Informativeness* and *Completeness*, annotators reached a high degree of consistency that our approach *GT-D2G* outperforms other baseline methods significantly. Moreover, *GT-D2G* performs best on *NYT* for *Node* metrics and *NYT* and *AMiner* for *Link* metrics.

Case Studies. The concept maps constructed by five methods are shown in Fig. 2.7 and 2.10. In general, *AutoPhrase* can represent meaningful concepts using phrases, but sometimes prone to generate duplicate nodes (*e.g.*, two *"mobile device"* in *AMiner* example). *TextRank* select meaningful concepts in word-level which are beneficial for the downstream tasks (*e.g.*, *"beethoven"* in *NYT*, *"mobile"* in *AMiner*, and *"amazing"* in *Yelp*), but the links among the selected concepts are not consistent with the original text. The nodes generated from *CMB-MDS* usually contain abundant information but are often in sentence-level, which are not concise and redundant. *doc2graph* can generate useful concepts with meaningful links, however, the nodes are mainly word-level (*e.g.*, *"mr."* instead of *"mr. haimovitz"* in *NYT*) and sometimes contain *"<unk>"* or *"-"* which indicate the limitation of this method. Our approach, *GT-D2G* can represent concepts in both word-level and phrase-level ways which are concise, semantic-rich, and beneficial for downstream tasks (*e.g.*, *"beethoven cello"* in *NYT*).

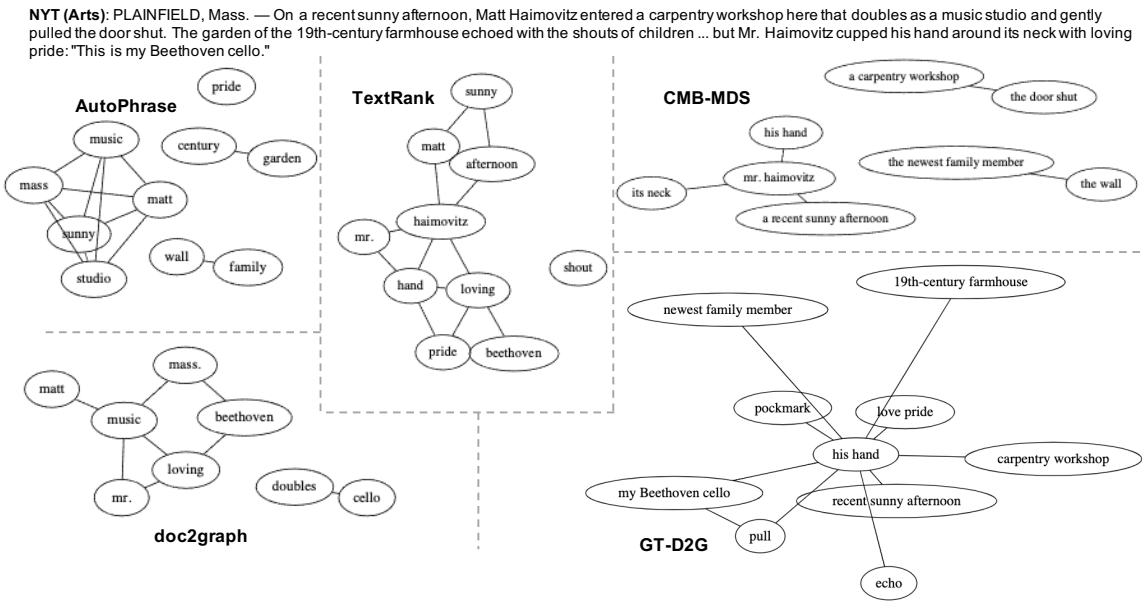


Figure 2.7: Concept maps generated by various models for case studies.

Table 2.3: Document classification accuracies(%).

Model	NYT	AMiner	Yelp
Bi-LSTM	87.52 ± 3.01	59.32 ± 2.71	78.46 ± 1.46
BERT-base	$\underline{97.54} \pm 0.16$	$\underline{73.62} \pm 0.06$	$\underline{85.34} \pm 0.08$
AutoPhrase	92.42 ± 0.65	59.63 ± 0.85	72.66 ± 0.33
TextRank	89.48 ± 0.07	57.47 ± 0.31	70.25 ± 0.61
CMB-MDS	87.68 ± 0.72	51.93 ± 2.02	65.63 ± 2.07
doc2graph	90.81 ± 1.00	67.06 ± 1.32	79.89 ± 0.52
GT-D2G-init	93.65 ± 0.86	66.76 ± 1.77	80.15 ± 0.80
GT-D2G-path	95.26 ± 0.13	68.23 ± 0.23	80.86 ± 0.97
GT-D2G-neigh	95.34 ± 0.33	$\mathbf{68.53} \pm 1.02$	80.92 ± 0.50
GT-D2G-var	$\mathbf{95.46} \pm 0.49$	68.37 ± 1.05	$\mathbf{80.98} \pm 0.51$

Classification Results (*RQ2*)

To answer *RQ2*, we conduct the document classification experiments on three text corpora. The generated concept maps have n concepts. To compare our methods with baseline methods conveniently, we set $n = 10$ for all graph-based baselines and non-flexible *GT-D2G* variants (*-path* and *-neigh*). For *GT-D2G-init*, n is equal to the total number of nodes of constructed initial graphs. For the flexible *GT-D2G* variant

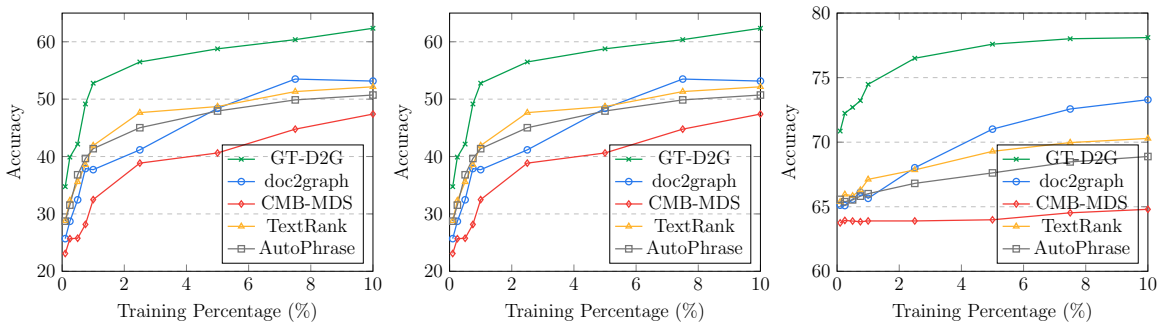
(*-var*), we set $n \leq 10$. Table 2.3 shows the classification performance of our methods and the compared methods. We observe that *GT-D2G* consistently outperforms all baseline methods except *BERT-base* on all three datasets, which indicates that the integration of semantic-rich initial concept maps from NLP pipelines and graph translation based on the weak supervision in our methods benefit the downstream tasks significantly. Notably, both *Bi-LSTM* and *BERT-base* are not capable of generating concept maps. As we mentioned before, the goal of *GT-D2G* is not to beat all SOTA document classification methods, but to achieve a competitive performance while providing interpretable structured knowledge representation. Consequently, in the following comparison elaborations, we exclude these two methods when we mention "baseline methods".

Compared with traditional graph-based approaches, *GT-D2G* gains 3%, 15%, 11% over the best results of traditional approaches on *NYT*, *AMiner*, and *Yelp*, respectively. Moreover, it surpasses the end-to-end *doc2graph* method by 5%, 2% and 1%, correspondingly. As mentioned in the toy example (Fig. 2.3) and Experiment Settings, both *AutoPhrase*, *TextRank* and *CMB-MDS* are existing unsupervised concept map generation models. These three models are capable of generating concept maps according to their own customized metrics (*e.g.*, frequency-based, connectivity-based, summarization-based), but they can not utilize the downstream task's signals to supervising the generation process. Consequently, concepts generated by these models are not task-oriented, thus leading to poor classification performance. On the other hand, *doc2graph* is the only compared model that is specifically designed for weakly-supervised concept map generation. As reflected in the experimental results, *doc2graph* is the major competitor of our *GT-D2G* (excluding the SOTA document classification models).

To better understand the effectiveness of our proposed techniques, we closely study the four variants of *GT-D2G* regarding the effectiveness of NLP pipelines (-

init), node-and-link iterative generation (*-path* and *-neigh*), and flexible-size graph generation (*-var*). In particular, to evaluate the effectiveness of incorporating NLP pipelines, we implement *GT-D2G-init* that directly encodes all nodes in the initial semantic-rich concept maps to make predictions. Table 2.3 show that *GT-D2G-init* outperforms all traditional graph-based baselines with 1.23 on *NYT*, 7.13% on *AMiner*, and 7.49% on *Yelp*. Comparing *GT-D2G-init* with *doc2graph*, *GT-D2G-init* achieves 1.23% and 0.26% gains on *NYT* and *Yelp*, while *GT-D2G-init* is worse by 0.3% on *AMiner*. Hence, the observed experimental results support the benefits of utilizing concept maps derived from NLP pipelines. Upon *GT-D2G-init*, the other three variants add the Graph Translator module to obtain a more concise concept map, since the initial concept maps often contain 20-40 nodes and the translated concept maps contain less than 10 nodes. According to the experimental results, the translated concept maps are preferable to initial concept maps, as they can further improve *GT-D2G-init* by 1.81% on *NYT*, 1.77% on *AMiner*, and 0.83% on *Yelp*.

To explore a proper way to generate edges, we implement and compare two methods, *GT-D2G-path* and *GT-D2G-neigh*. *GT-D2G-path* only generates edges based on the relations of concepts in text sequence while *GT-D2G-neigh* links each node with its all possible neighbors. As shown in Table 2.3, *GT-D2G-neigh* is consistently better than *GT-D2G-path* on all three datasets, which well supports our argument that generating edges among all possible neighbors is preferable to generating edges as a sequence of paths starting from the node. Furthermore, *GT-D2G-var* addresses the fixed size issue of *doc2graph* and the experiment results of *GT-D2G-var* illustrate the benefits of generating flexible size of concept maps. More discussion about generating size-flexible concept maps are in §2.3.3.



(a) NYT (b) AMiner (c) Yelp

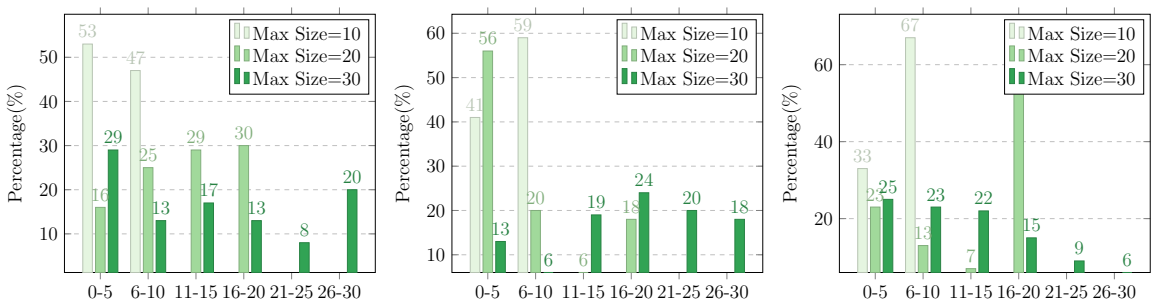
Figure 2.8: Test accuracies by varying the proportions of training data (ranging from 0.1%, 0.25%, ... to 10.00%).

Labeling Efficiency Evaluation (*RQ3*)

To demonstrate the labeling efficiency of *GT-D2G* over other concept map generation methods, we conduct experiments with different proportions (0.1%, 0.25%, 0.50%, 0.75%, 1.00%, 2.50%, 5.00%, 7.50%, 10.00%) of the training data. To get a stable test accuracy, we take the average value among three trials of each experiment by applying different random seeds. The average test accuracies of *NYTimes*, *AMiner*, and *Yelp* datasets were shown in Fig. 2.8 respectively, which answer *RQ3*.

We can observe that our approach *GT-D2G* has higher test accuracy than the other approaches from the beginning, with only 0.1% of the training data. In addition, with the increasing of the training data size, our model has steeper growth curves of test accuracy, which shows its effectiveness in exploiting limited supervision, and makes it maintain excellent performance during the whole label efficiency evaluation with limited labeled data. These results demonstrate the labeling efficiency of our model, which is enabled by the semantic-rich initial concept maps (§2.3.2) and the Gumbel-softmax training technique (§2.3.2). Therefore, *GT-D2G* can generate concept maps at scales not only without ground-truth training graphs but also without significant amounts of downstream task supervision.

Flexibility Evaluation (*RQ4*)



(a) NYT (b) AMiner (c) Yelp

Figure 2.9: Graph size distributions on different max graph sizes.

As discussed in RQ2, the $GT-D2G-var$ variant that is capable of generating flexible sizes of concept maps achieves the best document classification performance on two datasets (NYT and Yelp), while achieving the runner-up on the remaining dataset (AMiner). The observed experimental results justify the importance of the size-flexible property for concept map generation models.

To provide more insights, we further conduct experiments to explore the factors that impact the sizes of generated concept maps. As noted in the Training Techniques (§2.3.2), our framework is able to generate variable sizes of graphs by applying the RBF kernel-based graph size penalty and the content coverage penalty. These two penalties imply a trade-off between conciseness and completeness of generated concept maps. Fig. 2.9 shows the size distribution of the generated graphs on three datasets when the maximum graph size is set to be 10, 20, or 30 nodes. As can be seen, our *GT-D2G* can generate graphs with variable sizes as the size distribution varies according to the following two major factors: (a) input text complexity (across three datasets); (b) the preset hyperparameter "*max size*" (across different max sizes). For the input text complexity, we know that NYT and AMiner contain rather long and formal news articles and scientific reports, while Yelp contains short and informal online user-generated restaurant reviews. Consequently, concept maps derived from Yelp are

inclined to have small sizes, while concept maps from NYT and AMiner have more evenly size distributions (when the *max size* is set to 30). For the hyperparameter *max size*, we can clearly see the set value bounds the actual sizes of generated graphs.

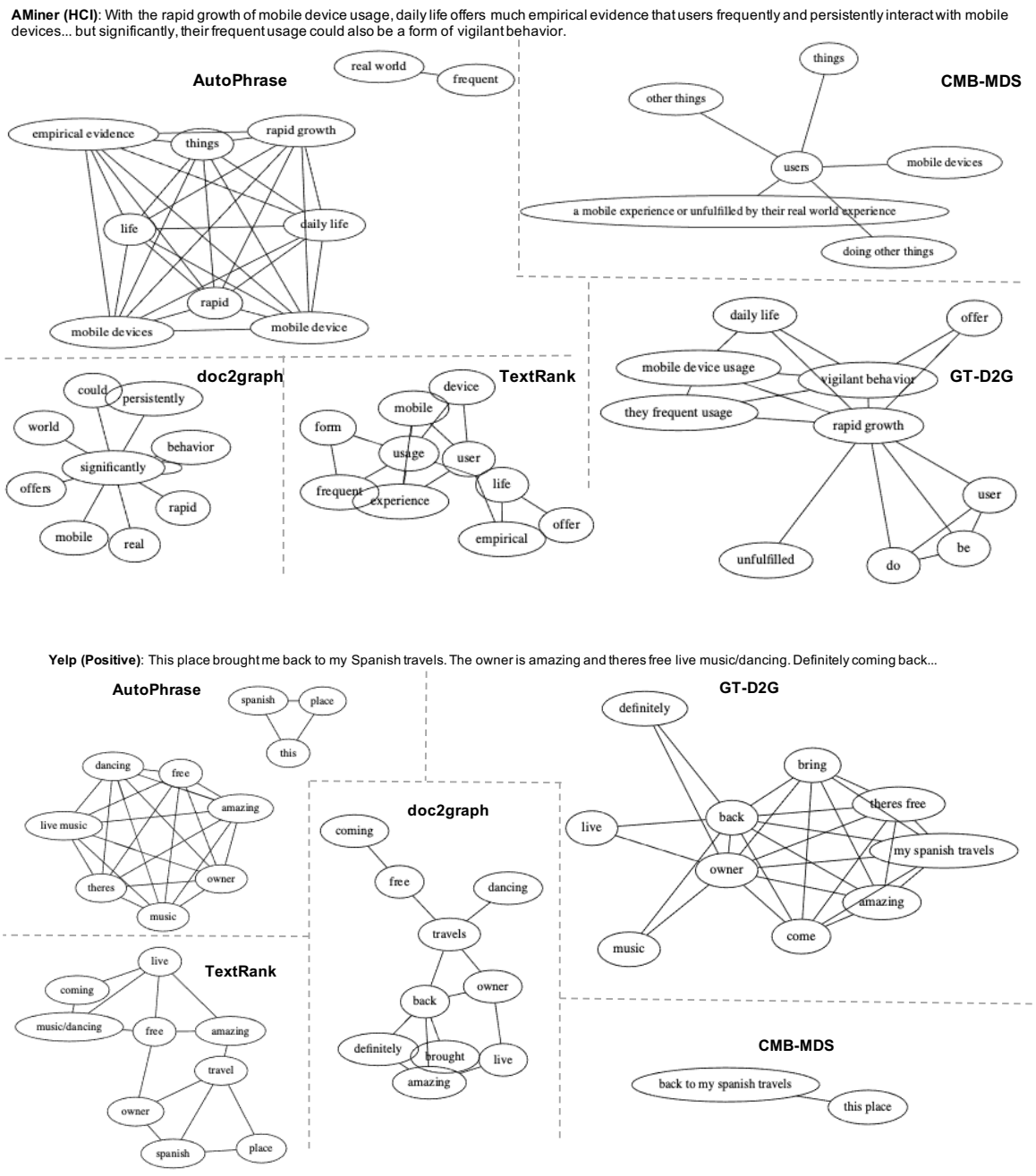


Figure 2.10: Concept maps generated by various models for case studies (*cont.*).

Chapter 3

Learn to Aligning and Completing Taxonomic Knowledge Graphs

3.1 Background

Knowledge bases (KBs) have incorporated large-scale multi-relational data and motivated many knowledge-driven applications such as online encyclopedia [85] and e-commerce product catalog [23]. Taxonomies and knowledge graphs (KGs), which represent real-world entities' abstract concepts and properties/behaviors/facts, constitute the essential information in knowledge bases (KBs). Taxonomies are useful tools to organize and index concepts of entities so that users can efficiently find the information of interest [74, 55]. On the other hand, KGs store human understanding of entities' properties, facts, or behaviors in a structured way, which are essential for knowledge representation and reasoning [22]. Extensive efforts have been made to construct KBs [8, 76] that include both taxonomies and KGs.

In this chapter, two of my first-author papers are included. The first one is "HiPrompt: Few-Shot Biomedical Knowledge Fusion via Hierarchy-Oriented Prompting" [52] (SIGIR'23), where we explore utilizing few-shot prompting empowered by

large language models (LLMs) to tackle the entity alignment between biomedical taxonomies and biomedical KGs. The second one is "Open-World Taxonomy and Knowledge Graph Co-Learning", where we investigate utilizing taxonomies as loosely-defined schema to align open-world KGs and then complete the aligned TaxoKGs.

3.2 HiPrompt: Few-Shot Biomedical Knowledge Fusion via Hierarchy-Oriented Prompting

We propose HiPrompt [52], a supervision-efficient knowledge fusion framework that elicits the few-shot reasoning ability of large language models through hierarchy-oriented prompts. Medical decision-making processes can be enhanced by comprehensive biomedical knowledge bases, which require fusing knowledge graphs constructed from different sources via a uniform index system. The index system often organizes biomedical terms in a hierarchy[1] to provide the aligned entities with fine-grained granularity. To address the challenge of scarce supervision in the biomedical knowledge fusion (BKF) task, researchers have proposed various unsupervised methods. However, these methods heavily rely on ad-hoc lexical and structural matching algorithms, which fail to capture the rich semantics conveyed by biomedical entities and terms. Recently, neural embedding models have proved effective in semantic-rich tasks, but they rely on sufficient labeled data to be adequately trained. HiPrompt bridges the gap between the scarce-labeled BKF and neural embedding models. Empirical results on the collected KG-Hi-BKF benchmark datasets demonstrate the effectiveness of HiPrompt.

In this work, we study the biomedical knowledge fusion (*BKF*) problem that aims to align entities from biomedical KGs into terms from the biomedical hierarchy.

[1]hierarchy: also mentioned as taxonomy. We will use these two terms interchangeably in this section.

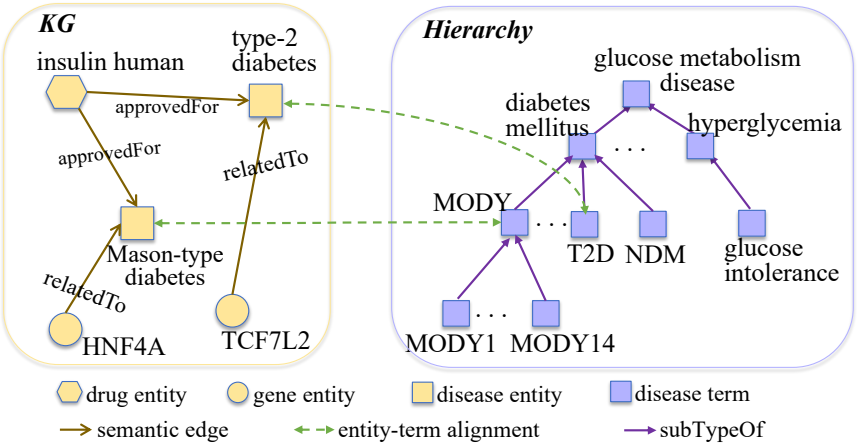


Figure 3.1: A toy example of BKF to find entity-term alignment between KG and hierarchy. *Left*: A KG containing biomedical entities. *right*: A hierarchy containing biomedical terms.

Figure 3.1 gives a toy example of the BKF task. The BKF task is challenging due to the following characteristics. First, inconsistent naming vocabularies are used in different resources, as they are developed independently by different groups of specialists. Second, unlike the existing KG entity alignment problem [78, 95] that contains many labeled entity-entity pairs as training samples, biomedical knowledge integration is supervision-scarce. Third, the topology of a KG and a hierarchy are very different, where the KG is a general graph, while the hierarchy is a directed acyclic graph.

3.2.1 Problem Definition

BKF aims at aligning existing specialized biomedical KGs into a uniform biomedical index system that can be represented by a hierarchy. We define the biomedical KG and hierarchy as follows: A biomedical KG is a multi-relation graph $\mathcal{G} = (E, R, RT)$, where E, R, RT are a set of various types of entities, a set of relation names, and $RT \in E \times R \times E$ is the set of relational triples, respectively. A biomedical hierarchy is a directed acyclic graph (DAG) $\mathcal{H} = (T, TP)$, where T is a set of terms, and $TP \in T \times T$ is a set of hypernym-hyponymy term pairs, respectively. The topology

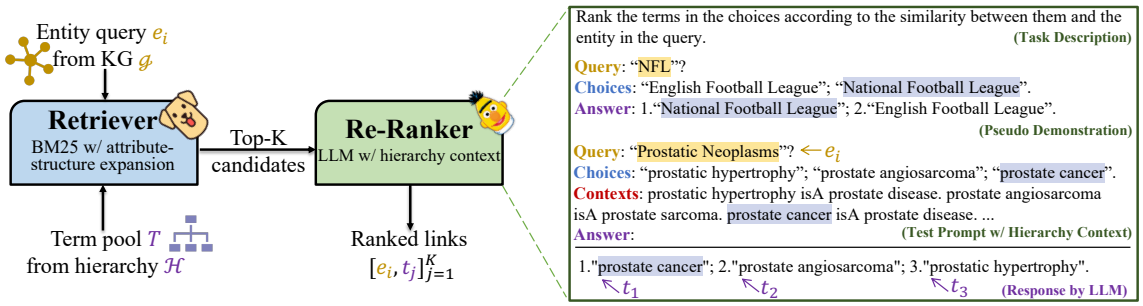


Figure 3.2: Overview of our HiPrompt framework, with a zoom-in on the LLM-based re-ranker.

differences between KG and hierarchy distinguish our BKF task from other related tasks (*e.g.*, entity alignment, KG integration). Moreover, both entities E and terms T contain rich associated semantic attributes (*e.g.*, definition, synonyms). Finally, we define our task as follows:

Definition 3.2.1 (biomedical knowledge fusion)**.** Given a biomedical KG $\mathcal{G}$, a biomedical hierarchy $\mathcal{H}$, a set of pre-aligned entity-term pairs $[e_a, t_a]_{a=1}^{M}$, and a set of unaligned entities $[e_1, e_2, \cdots, e_N] \in \mathcal{G}$. The goal is to link each unaligned entity to the hierarchy $LK = \{(e_i, t_j) | e_i \in \mathcal{G}, t_j \in \mathcal{H}\}$ such that t_j is the most specific term in the hierarchy for entity e_i in KG. In our work, we focus on the few-shot settings where the sample size M is very small to reflect the scarcity of labeled data that is ubiquitous in the biomedical field.

3.2.2 HiPrompt

Figure 3.2 shows the overall architecture of our proposed HiPrompt framework. To tackle the BKF task with limited training samples, our key insight is to utilize LLMs via hierarchy-oriented prompting. However, LLMs can not accommodate very lengthy input prompts (*e.g.*, GPT-3 only supports up to 4096 tokens) that contain all candidate terms along with their hierarchy contexts. A feasible workaround is to exhaustively examine each candidate term given the query entity, but the inference cost would be dramatic [63]. Therefore, we propose to use the *retrieve and re-rank* [86, 56, 30] approach to resolve the above challenges.

Retrieval Module. The retriever provides an efficient solution for coarse-grained candidate filtering, thus reducing the overall inference cost of HiPrompt. Given one entity query e_i from the KG $\mathcal{G}$ and all candidate terms T from the hierarchy $\mathcal{H}$, the retriever produces a coarsely ranked candidate list $(t'_1, t'_2, \cdots, t'_K)$, to avoid unnecessary computations for the LLM-based re-ranker. HiPrompt framework is flexible so that any unsupervised ranking function (*e.g.*, TF-IDF [69], LDA [6]) can be used to generate the ranked list. In practice, we choose the unsupervised BM25 [68] as the ranking function. Since entities and concepts have rich attributive and structural information, we further utilize these two types of information to expand [5] query entities and candidate terms.

Re-Ranking Module. Given the query entity e_i and the coarsely ranked candidate list $(t'_1, t'_2, \cdots, t'_K)$, we request the LLM to re-rank the list to $(t_1, t_2, \cdots, t_K)$ where t_1 is the most specific term of e_i via the gradient-free prompt-based learning. Figure 3.2 provides an example of the input prompt and the response of the re-ranker. The input prompt is composed of (1) curated textual ***task description***, (2) illustrative ***demonstration*** from few-show samples, and (3) the ***test prompt*** constructed from the query entity and the coarsely ranked list. The LLM-based re-ranker essentially tackles the BKF task by estimating the conditional probability: $P_{LLM}(w_1, w_2, \ldots, w_n|prompt)$, where $(w_1, \ldots, w_n)$ is the output word sequence with variable lengths. The desired re-ranked list can be converted from the output sequence by a simple mapping function $(t_1, t_2, \cdots, t_K) = f(w_1, w_2, \ldots, w_n)$.

For the template of demonstration, we use the query entity to form the question string "Query: $\{e_i\}$", the coarse candidate list to form the choice string "Choices: $\{t'_1$; t'_2; ... $t'_K\}$", and the ground truth to form the answer string "Answer: $\{t_1$; t_2; ..., $t_K\}$". While there is no such ground truth sample in the zero-shot setting, we propose the ***pseudo demonstration*** technique which adopts out-of-domain entity-term pairs to showcase what is the perspective format. Both real and pseudo demonstra-

tions are essential to generate output sequences in the consistent format [70, 41]. For the test prompt, we use the same template of the demonstration, while leaving the answer string as "Answer:" for LLM to predict what comes next. To further elicit LLMs with hierarchical constraints and dependencies of candidate terms, we propose the novel ***test prompt with hierarchy context*** where hypernyms of each candidate term are included in the context string. More specifically, we traverse the biomedical hierarchy $\mathcal{T}$ to locate the hypernym terms $t'_{i,p_1}, \cdots, t'_{i,p_j}$ of a candidate term t'_i. Therefore, the context string is formed as "Contexts: $\{t'_1$ isA $t'_{1,p}; \ldots; t'_K$ isA $t'_{K,p}\}$".

3.2.3 Experiments

Benchmark Datasets. We use the following data sources to create our KG-Hi-BKF benchmark[2]: (1) SDKG [106]: a disease-centric KG that covers five cancers and six non-cancer diseases. (2) repoDB [11]: we adopt their original triples, and generate entity attributes by querying DrugBank [91] and UMLS Metathesaurus [7]. (3) DzHi [72]: a hierarchy derived from the widely used Disease Ontology [72] which has a depth of 13. We first use the mapping existing in the resources themselves, which leads to many-to-many linkages between two KBs. We further manually verify the correctness of the many-to-many linkages and curate the datasets to the correct stage. Table 3.1 shows the statistics of the created benchmark. As can be seen, the linkages follow the one-to-one assumption [78], and the scale of labeled entity-term pairs is very small.

Compared Models. We compare HiPrompt to the following two sets of baselines: (a) *Non-neural conventional models*: (a.1) **Edit Dist** [67] that quantifies the distance between entities and terms by the edit distance of their names. (a.2) **BM25** [68] that ranks a set of documents based on the query tokens appearing in each docu-

[2]KG-Hi-BKF benchmark is available at https://doi.org/10.6084/m9.figshare.21950282.

Dataset	Source	#Disease	#Entities	#Links
SDKG-DzHi	SDKG	841	19,416	635
	DzHi	11,159	11,159	635
repoDB-DzHi	repoDB	2,074	3,646	709
	DzHi	11,159	11,159	709

Table 3.1: Statistics of the KG-Hi-BKF benchmark.

ment. (a.3) **LogMap** [39] that matches entities and terms via logical constraints and semantical features. (a.4) **PARIS** [77] that provides a off-the-shelf fusion tool empowered by a parameter tuning-free probabilistic model. (a.5) **AML** [27] that is based on non-literal string comparison algorithms. is a probabilistic matching system based on probability estimates. (b) *Neural embedding models*: (b.1) **SapBERT** [43] that learns to self-align synonymous biomedical entities through a Transformer. (b.2) **MTransE** [14] that extends the translational KG embedding method TransE [9] to multi-language system entity alignment by axis calibration and linear transformations. (b.3) **SelfKG** [46] that designs a self-negative sampling strategy to push sampled negative pairs far away from each other when no labeled positive pairs are available.

Setting	Model	SDKG-DzHi						repoDB-DzHi					
		Hits@1	Hits@3	nDCG@1	nDCG@3	WuP	MRR	Hits@1	Hits@3	nDCG@1	nDCG@3	WuP	MRR
Zero-shot	Edit Dist	65.51	70.39	68.08	50.82	85.53	68.69	68.69	71.37	71.71	54.15	85.21	70.71
	BM25	73.07	87.40	77.56	63.01	91.97	81.06	59.38	74.75	70.33	64.51	90.71	68.84
	LogMap	75.75	79.06	76.97	54.82	85.06	77.38	86.60	87.73	87.38	60.79	91.68	87.09
	PARIS	22.68	22.68	23.15	16.13	43.85	22.68	6.35	6.35	6.42	4.44	32.28	6.35
	AML	OOM	OOM	OOM	OOM	OOM	OOM	78.00	78.56	78.67	54.90	86.02	78.26
	SapBERT	69.61	87.24	76.38	63.86	93.78	78.97	75.04	90.69	81.24	73.51	94.25	83.61
	SelfKG	57.95	69.45	58.98	47.29	74.25	64.70	72.78	81.10	75.95	63.78	88.41	77.71
	HiPrompt	**90.79**	**93.08**	**91.57**	**77.00**	**96.74**	**92.13**	**88.01**	**91.26**	**90.70**	**82.85**	**97.06**	**90.64**
One-shot	SapBERT	69.56	87.22	76.34	63.84	93.29	78.93	75.00	90.68	81.21	73.51	94.13	83.59
	MTransE	0.0	0.16	0.0	0.05	35.09	0.16	0.0	0.28	0.14	0.27	28.89	0.37
	HiPrompt	**92.11**	**95.11**	**93.53**	**77.63**	**97.25**	**93.91**	**88.28**	**91.53**	**90.61**	**81.31**	**96.39**	**90.28**

Table 3.2: Main experiment results (in percentages).

Quantitative evaluations. We mainly focus on zero-shot and one-shot settings, and utilize the remaining labeled samples as the test set to report quantitative results. Several *strict* and *lenient* evaluation metrics are used. For strict metrics that appreciate only the exact correct prediction, we adopt **Hits@k** and mean reciprocal

rank (**MRR**). For lenient metrics that also reward near-hits, we adopt **nDCG@k** with exponential decay [3] and hierarchy-based term relatedness score **WuP** [94]. All compared baselines are executed with their recommended hyperparameters. For all non-neural conventional models, we only report the zero-shot results as they are unsupervised methods. For neural embedding methods, we report the zero-shot results utilizing released model weights (SapBERT) or conducting self-supervised training (SelfKG), while reporting the one-shot results by fine-tuning these models (SapBERT, MTransE) on the one demonstrative training sample. For our HiPrompt, we use GPT-3 [12] as the LLM for re-ranker and set its temperature hyperparameters as 0 to lower the completion randomness. Using a single prompt template is sufficient since initial exploration shows that various templates do not have a significant impact on model performance. We exclude the use of automatic prompt generation techniques [75, 105] due to the limited availability of training data.

Main Results. Table 3.2 shows the quantitative results for zero-shot and one-shot settings. HiPrompt largely outperforms all other methods in all evaluation metrics under both settings, which demonstrates the effectiveness of the proposed hierarchy-oriented prompting. Under the zero-shot setting, the non-neural unsupervised baseline LogMap achieves the second-best performance. All examined models can successfully generate predictions except AML throws out-of-memory (OOM) errors on the SDKG-DzHi dataset. PARIS performs worst in the zero-shot setting because it can not predict aligned terms for each query entity. Instead, PARIS produces the alignment based on its own ad-hoc threshold. MTransE performs worst in the one-shot setting since it is underfitting using just one training sample. Comparing the same models (SapBERT, HiPrompt) between zero-shot and one-shot settings, we observe the performance differences are negligible, thus indicating that effectively eliciting the adaptive reasoning ability is one of the key factors to tackling supervision-scarce BKF problem.

Expan.	*SDKG-DzHi*			*repoDB-DzHi*		
	Hits@5	Hits@10	Hits@20	Hits@5	Hits@10	Hits@20
Name	88.66	89.61	90.55	85.05	88.72	90.27
+Atr.	94.96	96.85	98.11	89.00	92.52	95.20
+Str.	90.08	90.71	91.81	88.15	90.27	92.24
+Atr.+Str.	**96.85**	**97.64**	**98.74**	**91.11**	**93.65**	**95.63**

Table 3.3: Retriever with various expansion strategies.

LLMs	SDKG-DzTaxo			repoDB-DzTaxo		
	Hits@1	Hits@3	MRR	Hits@1	Hits@3	MRR
	One-shot (prompt w/o Hi. Context)					
GPT-3	**91.80**	**94.32**	**93.45**	**87.85**	**91.24**	**89.92**
GPT-JT	75.08	86.44	81.80	58.33	69.77	66.42
OPT-6.7B	68.93	80.44	76.38	60.73	73.59	69.33
	One-shot (prompt w/ Hi. Context)					
GPT-3	**92.11**	**95.11**	**93.91**	**88.28**	**91.53**	**90.28**
GPT-JT	80.76	93.69	87.45	69.07	82.91	77.24
OPT-6.7B	72.40	84.86	79.64	63.70	77.68	72.41

Table 3.4: Re-ranker with various LLMs and prompts.

Ablation Studies. We further conduct ablation studies to evaluate the impact of our hierarchy-oriented techniques. Table 3.3 compares the different expansion strategies for HiPrompt's retrieval module. As can be seen, if expanding the KG entities and hierarchy terms with both attributive and structural features ("*+Atr.+Str.*" variant), the retriever can achieve the best Hits@K performance. Table 3.4 compares different LLMs and different prompts for HiPrompt's re-ranking module. Among the examined LLMs, GPT-3 with 175 billion parameters surpasses GPT-JT [80] with 6B parameters and OPT-6.7B [103] with 6.7B parameters due to its large parameter space. When adding the proposed hierarchy context to the name-only prompts, every LLM achieves better performance on all metrics, thus demonstrating the importance of explicit hierarchy-oriented information. We also observe that improvements for GPT-JT and OPT-6.7B are more significant than GPT-3, since GPT-3 may already have such hierarchical information encoded.

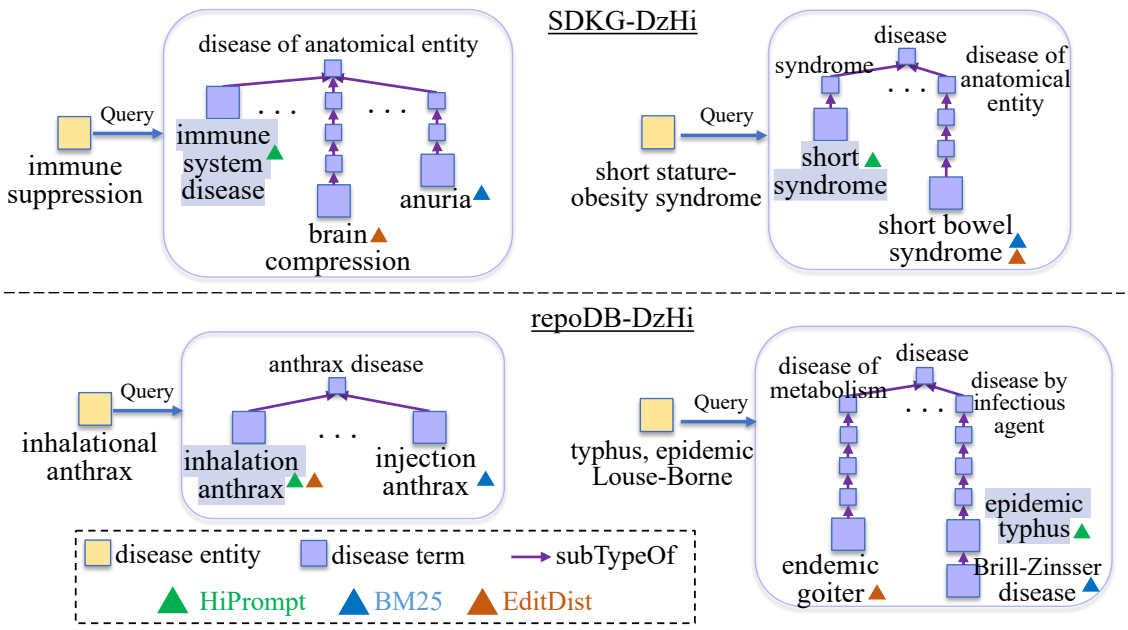


Figure 3.3: Case Studies on unlabeled data. Terms highlighted in violet denote the correct alignments for query entities.

Case Studies. Figure 3.3 shows the fusion results from BM25, EditDist, and HiPrompt. In general, HiPrompt can find the most specific terms in the hierarchy for the query entities, by satisfying the semantic similarities and hierarchical constraints simultaneously. For instance, HiPrompt recognizes that "*immune system disease*" is the most appropriate for the query "*immune suppression*", rather than its hypernym "*disease of anatomical entity*" that is too general, or hyponyms such as "*immune system cancer*" or "*allergic disease*" that are too specific. On the other hand, EditDist only considers lexical matching, thereby ignoring the different naming conventions of the same biomedical concepts. BM25 also mainly relies on lexical matching, but it incorporates the names, definitions, and synonyms of biomedical terms during the matching, resulting in better performance in handling various names. However, BM25 ignores the hierarchical information, which leads to the inappropriate granularity of aligned terms (*e.g.*, the term "*epidemic typhus*" is too broad for the query entity "*typhus, epidemic Louse-Borne*").

3.3 Open-World Taxonomy and Knowledge Graph Co-Learning

We propose to create open-world TaxoKGs based on existing automatically constructed taxonomies and open KGs, to empower KBs towards easy accommodation of emerging entities and relations. Most existing KBs are constructed under the closed-world assumption, which often corresponds to a fixed schema and requires ad-hoc canonicalization to integrate new knowledge. We observe that taxonomies can serve to provide a loosely defined schema and mitigate the reliance on ad-hoc canonicalization. To further improve the completeness of TaxoKG, we collect several new benchmark datasets towards the development of HakeGCN, an innovative hierarchy-aware graph-friendly model for TaxoKG completion.

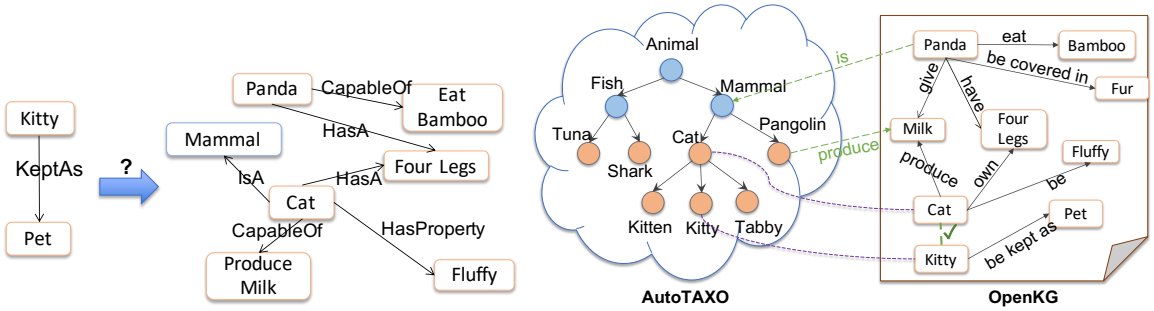
The knowledge stored in KBs can be categorized into two types:

1. The taxonomic knowledge that contains hierarchical *IsA* relations between *entities* and *abstract concepts*, which are stored in *taxonomies* (*e.g.*, "*(Cat, IsA, Mammal)*" in Fig. 3.4a);

2. The non-taxonomic knowledge that contains graph-structured interactions between *entities* and attributes of *entities*, which are stored in *knowledge graphs* (KGs) (*e.g.*, "*(Cat, HasProperty, Fluffy)*" in Fig. 3.4a).

Taxonomies are useful tools to organize and index concepts of entities so that users can efficiently find the information of interest [74, 55]. On the other hand, KGs store human understanding of entities' properties, facts, or behaviors in a structured way, which are essential for knowledge representation and reasoning [22]. Extensive efforts have been made to construct KBs [8, 76] that include both taxonomies and KGs. However, most existing KBs are in closed domains, and the creation process highly relies on pre-defined schema [66] and exhaustive entity/relation canonicalization [92]. Although with guaranteed precision, closed-world KBs are limited in coverage and

freshness. For example, if a KB is defined with a curated evolutionary biology schema that focuses on taxon and related characteristics of organisms, it is hard to incorporate knowledge triplets such as "(*Cat*, *KeptAs*, *Pet*) and (*German Shepherd*, *TrainedAs*, *Detection Dog*)". On the other hand, when a new triplet "(*Kitty*, *KeptAs*, *Pet*)" is introduced, although as humans we know kitty is a synonym of cat, the closed-world KB cannot easily incorporate the new knowledge unless the canonicalization tool can identify *Kitty* as *Cat*. Therefore, closed-world KB is most suitable for fixed or slowly evolving knowledge-enhanced applications.



(a) Closed-world KB (b) TaxoKG

Figure 3.4: Toy examples of existing KBs and TaxoKG.

3.3.1 Problem Definition

The TaxoKG completion task is a variant of the general open-world KB completion:

Definition 3.3.1 (Open-world KB Completion)**.** Given the incomplete KB $\mathcal{B} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ where $\mathcal{V}$, $\mathcal{R}$ and $\mathcal{E}$ are entity set, relation set and triplet set, open-world KB completion aims at inferring the missing triplets $\{(s, r, o)|(s, r, o) \notin \mathcal{E}, s \in \mathcal{V}^s, r \in \mathcal{R}^s, o \in \mathcal{V}^s\}$, where $\mathcal{V}^s$ and $\mathcal{R}^s$ are entity superset and relation superset, respectively.

More specifically, TaxoKG $\mathcal{B}$ contains the taxonomy $\mathcal{T}$ and the knowledge graph $\mathcal{G}$. An AutoTAXO $\mathcal{T} = (\mathcal{V}_e, \mathcal{V}_c, \mathcal{E}_\mathcal{T})$ is a collection of entity-concept pairs, where $\mathcal{V}_e$ and

$\mathcal{V}_c$ are entity and concept sets, and $\mathcal{E}_\mathcal{T} = \{(e, c)\} \subseteq \mathcal{V}_e \times \mathcal{V}_c$ is the set of taxonomic edges, all of which carry the uniform *IsA* relation. An OpenKG $\mathcal{G} = (\mathcal{V}_e, \mathcal{R}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ is a collection of subject-relation-object triplets, where $\mathcal{V}_e$ is the entity set shared with $\mathcal{T}$, $\mathcal{R}_\mathcal{G}$ is the relation set that contains all other relations except for the taxonomic ones, and $\mathcal{E}_\mathcal{G} = \{(s, r, o)\} \subseteq \mathcal{V}_e \times \mathcal{R}_\mathcal{G} \times \mathcal{V}_e$ is the edge set connecting entities with associated relations. Hence, there exist two sub-tasks for TaxoKG completion: (1) the AutoTAXO concept prediction task and (2) the OpenKG relation prediction task. The former is to assign a set of concepts $C_e = \{c_1, c_2, \ldots, c_m\}$ for each entity $e \in \mathcal{V}_e$, whereas the latter aims to predict missing facts in the form of $q_s = (?, r_k, o_j)$ or $q_o = (s_i, r_k, ?)$. It is worth noting that $e, s, o \in \mathcal{V}_e^s$, $c \in \mathcal{V}_c^s$, and $r \in \mathcal{R}_\mathcal{G}{}^s$, which means we need to handle unseen entities, concepts, and relations.

3.3.2 TaxoKG-Bench: A Novel Benchmark with Six Datasets for TaxoKG

To the best of our knowledge, our work is the first to study the open-world taxonomy and knowledge graph co-learning problem. Hence, we create and release TaxoKG-Bench with six datasets of large-scale TaxoKG to the community for future studies[3].

Creation Process

The goal of building TaxoKG-Bench is to provide a benchmark to evaluate models on TaxoKG-based tasks such as its completion and applications. TaxoKG completion involves the ability to predict new-emerging concepts and novel facts for unseen entities. TaxoKG-Bench integrates the following data sources:

- Three AutoTAXOs: MS Concept Graph (MSCG) [93], SemEval-2018 Task 9 2A:Medical (SEMedical) and 2B:Music (SEMusic) [13];
- Two OpenKGs: ReVerb [24] and OPIEC [28].

[3]TaxoKG-Bench: https://figshare.com/articles/dataset/Taxo-KG-Bench/16415727

Statistics of TaxoKG-Bench

MSCG × *ReVerb* and *MSCG* × *OPIEC* are two large-scale TaxoKGs containing billions knowledge triplets of before filtering. Therefore, we set high thresholds for them. In particular, concepts with at least 20 grounded entities are kept in both MSCG × ReVerb and *MSCG* × *OPIEC* datasets, while entities with frequency greater than or equal to $40, 25$ are kept in *MSCG* × *ReVerb* and *MSCG* × *OPIEC*, respectively. For relation, frequencies greater than or equal to $35, 3$ are kept. Nevertheless, the remaining knowledge triplets are still in million scales, which makes the evaluation on these two *Taxo-KG*s very slow. We then conduct further down-samplings to build lightweight yet diverse testbeds. Similarly, we set the concept threshold, entity threshold, and relation threshold for *SEMedical* aligned and *SEMusic* aligned *Taxo-KG*s as $\{3, 2, 2\}$ and $\{3, 4, 3\}$, respectively.

Table 3.5: Statistics of the six datasets in TaxoKG-Bench.

Dataset	# entity	# concept	# pair	# mention	# predicate	# triplet
MSCG × ReVerb	5.6/1.0/3.6(K)	1.8/0.5/1.4(K)	6.4/1.2/4.0(K)	12.8/3.8/7.0(K)	10.3/2.2/4.8(K)	59.7/3.7/11.2(K)
SEMedical × ReVerb	256/48/163	261/131/219	256/48/163	7.3/1.3/2.9(K)	6.1/0.9/2.3(K)	21.3/1.3/4.0(K)
SEMusic × ReVerb	412/76/262	335/229/283	412/76/262	7.5/2.1/4.1(K)	8.9/1.7/3.7(K)	41.2/2.6/7.7(K)
MSCG × OPIEC	6.3/1.1/4.0(K)	1.8/0.6/1.4(K)	7.6/1.4/4.8(K)	5.5/1.8/3.2(K)	3.2/0.4/0.9(K)	51.2/3.2/9.6(K)
SEMedical × OPIEC	238/44/151	256/136/209	238/44/151	1432/255/564	508/75/199	2239/176/499
SEMusic × OPIEC	443/81/282	363/256/305	443/82/282	3.6/1.2/2.3(K)	1.4/0.3/0.6(K)	15.9/1.5/3.9(K)

After the downsampling process mentioned above, we then split the six TaxoKGs into training, validation, and testing sets for setting up a reproducible benchmark. On the AutoTAXOs side, we split the entity-concept pairs by randomly assigning 55%, 5%, and 35% entities into training, validation, and testing sets. On the OpenKG side, we split subject-relation-object triplets by randomly assigning 80%, 5%, and 15% triplets into training, validation, and testing sets. In other words, each split set is the union of the assigned ontology-relation set and instance-relation set.

For more details about TaxoKG-Bench, we encourage interested readers to refer to our paper [49].

3.3.3 HakeGCN: A Novel Method for Effective TaxoKG Completion

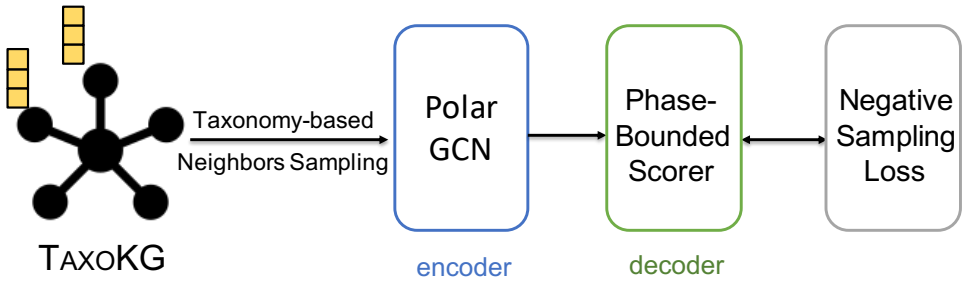


Figure 3.5: HakeGCN model architecture.

To tackle the TaxoKG completion task, our key insight is to leverage the mutual enhancement between taxonomy and KG. Hence, we propose a novel model with the learn-to-conceptualize and learn-to-generalize abilities via combining **H**ierarchy-**A**ware **K**nowledge base **E**mbedding and **G**raph **C**onvolutional neural **N**etworks, namely HakeGCN. As Fig. 3.5 shows, it can be regarded as an encoder-decoder model. HakeGCN includes a series of essential technical designs for TaxoKG completion: (1) The polar coordinates-based GCN encoder which joins the power of GCNs in modeling multi-relations in KGs and polar coordinates in modeling hierarchical relations. (2) The taxonomy-based sampling strategy to improve the GCN encoder in learning from less-noisy neighbors. (3) The GCN-oriented phased bounded decoder that modify the value boundary of the original phase coordinate score function in HAKE [104], making it easier for the decoder to differentiate entities at the same taxonomy level.

Handling Unseen Entities, Concepts and Relations

As §3.3.2 states, there are numerous unseen entities, concepts, and relations in the TaxoKG completion task. Unfortunately, most existing KB completion models [9, 104, 71] are developed under the closed-world assumption, therefore their solution to

embed entities/concepts/relations is to treat them as phrases and assign a look-up embedding table for phrases seen in the training set. Consequently, these models cannot handle new emerging phrases in the open-world setting. In HAKEGCN, we opt to create entity, concept, and relation representations from the tokens of the surface mentions [10]. The entity and concept representations are then fed into the GCN encoder as initial embeddings of vertices $\boldsymbol{h_v^0}$, and relation representations as initial embeddings of edges $\boldsymbol{h_r^0}$. Therefore, for any vertex or edge h that is in the form of a sequence of tokens $\{t_1, t_2, \ldots, t_L\}$, the representation is calculated by

$$\boldsymbol{h} = f(h) = f_{phr}(f_{tok}(t_1), f_{tok}(t_2), \ldots, f_{tok}(t_L)), \tag{3.1}$$

where the lowercase letter h denotes vertex or edge phrase, the boldface lowercase letter $\boldsymbol{h}$ denotes the phrase embedding of vertex or edge, $f_{tok} : \mathbb{V}^{Tok} \to \mathbb{R}^d$ denotes the token embedding look-up mapping function, and $f_{phr} : \mathbb{R}^{L \times d} \to \mathbb{R}^{d'}$ denotes the phrase composition function. The choice of composition functions is flexible, which includes *average*, *sum*, *max*, *RNN* and even *Transformer*. In HAKEGCN, we choose *average* for the sake of simplicity. The token embedding look-up table is shared among vertices and edges.

After taking the average of token embeddings, we apply different single-layer perceptrons on $\boldsymbol{h_v}, \boldsymbol{h_r}$ to obtain the vertex and edge embeddings:

$$\boldsymbol{h_v^0} = \mathrm{PReLU}(\boldsymbol{W_v h_v} + \boldsymbol{b_v}) \quad \text{and} \quad \boldsymbol{h_r^0} = \mathrm{PReLU}(\boldsymbol{W_r h_r} + \boldsymbol{b_r}). \tag{3.2}$$

Here, we use v to represent any entity $e \in \mathcal{V}$ and concept $c \in \mathcal{V}$ that can be viewed as the vertex of knowledge base $\mathcal{B} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$. Similarly, we use r to represent the IsA relation $\mathcal{R}_{\mathrm{IsA}} \in \mathcal{R}$ of AutoTaxo and any relation $r \in \mathcal{R}$ of OpenKG that can be viewed as the edge of $\mathcal{B}$. For the non-linear activation, we opt for PReLU [36]. The superscript 0 denotes that we use them as the input of the GCN encoder.

GCN Encoder with Polar Convolution and Taxo-based Neighbor Sampling

Our novel encoder is a generalization of inductive GCN encoders in polar coordinates that benefits from the expressiveness of both graph neural networks and hierarchy-aware polar embeddings. First, since the input vertex and edge embeddings are Cartesian coordinate embeddings, we derive the relational neighbor aggregation and embedding updating in the Cartesian system. Next, we derive a mapping from Cartesian coordinates to polar coordinates. We finally use the mapped polar embeddings of vertices and edges as the input of the decoder.

Updating Embeddings in Cartesian Coordinate. Since most existing KG embedding methods consider the input features or initial embeddings of entities and relations in the Cartesian coordinate system [88], we first adopt the widely-studied relational-GCNs in the Cartesian coordinate system. The choice of the GCN encoder is flexible, as long as it takes both vertex and edge representations into account. We propose our own GCN encoder, which is a generalized form of existing relation-GCNs:

$$\boldsymbol{m}_v^{k+1} = \textsc{Agg}(\{\boldsymbol{W}_{\boldsymbol{dir(r)}}^{k}\ \phi(\boldsymbol{h}_u^k, \boldsymbol{h}_r^k), \forall (u,r) \in \mathcal{N}(v)\}), \tag{3.3}$$

$$\boldsymbol{h}_v^{k+1} = \text{PReLU}(\boldsymbol{W}_v^k[\boldsymbol{h}_v^k \parallel \boldsymbol{m}_v^{k+1}] + \boldsymbol{b}_v^k). \tag{3.4}$$

The message $\boldsymbol{m}_v^{k+1}$ on vertex v is collected from the neighbors $\mathcal{N}(v)$. The composition function $\phi(\boldsymbol{h_u}, \boldsymbol{h_r})$ can be either $\boldsymbol{h_u} - \boldsymbol{h_r}$, $\boldsymbol{h_u} * \boldsymbol{h_r}$ or $\boldsymbol{h_u} \star \boldsymbol{h_r}$ [60]. The aggregation operator $\textsc{Agg}(\cdot)$ can be chosen from *average*, *sum*, *max* or other functions. In practice, we select $\phi(\boldsymbol{h_u}, \boldsymbol{h_r})$ and $\textsc{Agg}(\cdot)$ through hyperparameter tuning. Moreover, the relation-specific learnable parameter $\boldsymbol{W}_{\boldsymbol{dir(r)}}$ [82] in Eq. (3.3) is

$$\boldsymbol{W}_{\boldsymbol{dir(r)}} = \begin{cases} \boldsymbol{W_o}, & (u,r,v) \in \mathcal{E}, \\ \boldsymbol{W_I}, & (u,r,v) \in \mathcal{E}_{inv}, \end{cases} \tag{3.5}$$

where $\mathcal{E}_{inv}$ denotes invert edges introduced to $\mathcal{B}$ for better vertex and edge representations. In Eq. (3.4), $[\boldsymbol{h}_v^k \parallel \boldsymbol{m}_v^{k+1}]$ denotes concatenation of the node and the message representations. Our novel design is that we do not introduce self-loops during message aggregation, but concatenate the self node embeddings with aggregated neighborhood embedding during node representation updating. Moreover, the edge updating rule is:

$$\boldsymbol{h}_r^{k+1} = \text{PReLU}(\boldsymbol{W}_r^k \boldsymbol{h}_r^k + \boldsymbol{b}_r^k). \tag{3.6}$$

Eq. (3.6) is only used to update edges in the training graph during the encoding phase, the representation of relation r for predicting knowledge triplet (s, r, o) is calculated through another similar transformation in the decoder.

Mapping from Cartesian to Polar Representations. The polar coordinate-based embedding have shown promising results in closed-world KB completion [79, 104], as it utilizes the modulus dimension information to reflect depth of the taxonomy hierarchy and the phase dimension to represent the entities' surrounding non-taxonomic relations. The neighborhood aggregation and updating operations in the HakeGCN encoder are defined in the Cartesian coordinate system, while it is ideal for the decoder to consider both hierarchical and other relations in TaxoKG in the polar coordinate system [104]. To bridge the gap between the Cartesian coordinate embeddings from HakeGCN encoder and the polar coordinate embeddings used by decoder, we conduct the following representation mapping:

$$\rho = \sqrt{x^2 + y^2} \quad \text{and} \quad \theta = \text{atan2}(y, x), \tag{3.7}$$

where $x, y \in \mathbb{R}$, $\rho \in \mathbb{R}_+$, and $\theta \in [-\pi, +\pi]$. The atan2 function is a variation of the *arctangent* function. During the polar convolution process above, vertex and edge

embeddings in Cartesian coordinate can be denoted as $\boldsymbol{h} = [\boldsymbol{x} \parallel \boldsymbol{y}]$. Assuming $\boldsymbol{h}$'s dimension is $2d$, then $\boldsymbol{h}$ stores d pairs of Cartesian coordinates. Therefore, using Eq. (3.7), $\boldsymbol{h}$ can be mapped into $\boldsymbol{h} = [\boldsymbol{\rho} \parallel \boldsymbol{\theta}]$ containing d pairs of polar coordinates.

Inductive GCN for open-world TaxoKG. Since the TaxoKG completion task is under the open-world setting, the GCN encoder needs to generalize to unseen vertices and edges. Therefore, we construct the polar encoder as an inductive GCN to efficiently generate vertex and edge embeddings for previously unseen data [33]. For the training set and testing set, two different graphs are built where the training graph contains all seen entities, concepts, and relations, while the testing graph is the union of the training graph and unseen entities, concepts (introduced as new disconnected vertices) from testing sets. For relation representations, as the relations may not show during training, we use the relation transformation perceptron defined in Eq. (3.2) to obtain them.

Taxonomy-based Neighborhood Sampling. We propose a taxonomy-based neighbor sampling strategy that intentionally keeps useful neighbors and discards noisy ones, which is an advancement of existing uniform neighbor sampling [71]. The intuition is to allow the GCN encoder to see more neighbors close on the taxonomy, which contains less noise. Although the neighborhood information is helpful for KB completion tasks, many existing GCN-based models keep all neighbors during training which introduces noisy and even hazardous information [100, 82]. For instance, presented "platypus is a mammal but lays eggs", GCN-based models may induct that laying eggs is a positive factor to judge an animal belongs to the mammal category. To relieve the noisy, RGCN [71] proposes to apply uniform random edge dropout on its encoder, which may discard useful neighborhood information. Therefore, we propose a taxonomy-based neighbor sampling strategy that intentionally keeps useful neighbors and discards noisy ones. Taxonomy-based sampling assigns a higher probability for edges between the entity of interest and the neighbors connected by both

entity-entity and entity-concept edges. The intuition is to allow the GCN to see more neighbors on the taxonomy, which contains less noise. The value of higher chance is chosen through hyper-parameter tuning

GCN-Oriented Phase Bounded Decoder

After getting the representations from the GCN-based encoder, the decoder scores "(subject, relation, object)" triplets through a function $f(s,r,o) : \mathbb{R}^d \times \mathbb{R}^{d'} \times \mathbb{R}^d \to \mathbb{R}$. We adopt the polar coordinate score function [104] with a GCN-oriented boundary:

$$f(s,r,o) = -d(s,r,o) = -\lambda_m d_m(s,r,o) - \lambda_p d_p(s,r,o), \tag{3.8}$$

where (s,r,o) denotes both entity-concept pairs (with the associated relation of "IsA") and entity-relation-entity triplets in TaxoKG, and $d(s,r,o)$ denotes the distance function. In particular, $\lambda_m, \lambda_p \in \mathbb{R}$ are two learnable parameters to balance the modulus distance $d_m(s,r,o)$ and the phase distance $d_p(s,r,o)$. We also propose a GCN-oriented boundary for d_p for effective optimization. Similar to HAKE [104], the modulus and phase distance functions in Eq. (3.8) $f(s,r,o) = -\lambda_m d_m(s,r,o) - \lambda_p d_p(s,r,o)$, where (s,r,o) denotes both entity-concept pairs (with the associated relation of "IsA") and entity-relation-entity triplets in TaxoKG, and $d(s,r,o)$ denotes the distance function. In particular, $\lambda_m, \lambda_p \in \mathbb{R}$ are two learnable parameters to balance the modulus distance $d_m(s,r,o)$ and the phase distance $d_p(s,r,o)$. The modulus and phase distance functions are defined by the following equations:

$$d_m(s,r,o) = \left\| \boldsymbol{h_{s,m}} \circ \boldsymbol{h_{r,m}} - \boldsymbol{h_{o,m}} \right\|_2, \tag{3.9}$$

$$d_p(s,r,o) = \left\| \sin(\boldsymbol{h_{s,p}} + \boldsymbol{h_{r,p}} - \boldsymbol{h_{o,p}}) \right\|_1, \tag{3.10}$$

where $\boldsymbol{h_s}, \boldsymbol{h_o}$ denote the subject, object embeddings obtained from the GCN encoder

production $\boldsymbol{h_u}$ in Eq. (3.4), and $\boldsymbol{h_r}$ denotes the relation embedding obtained from a separate transformation in decoder using a similar process as in Eq. (3.6). For the polar coordinate, $\boldsymbol{h_{*,m}}, \boldsymbol{h_{*,p}}$ denote the embeddings in the modulus and phase part. In Eq. (3.9), the operator $\circ : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ denotes the Hadamard product between two vectors. Let $\Delta\boldsymbol{\theta} = \boldsymbol{h}_{s,p} + \boldsymbol{h}_{r,p} - \boldsymbol{h}_{o,p}$. In the original phase distance function of HAKE, there is a denominator 2 for $\Delta\boldsymbol{\theta}$, which leads Eq. (3.10) to $\left\|\sin(\frac{\Delta\boldsymbol{\theta}}{2})\right\|$. This is due to $\boldsymbol{h}_{*,p} \in [0, 2\pi)^d$, and thus $(\boldsymbol{h}_{s,p} + \boldsymbol{h}_{r,p} - \boldsymbol{h}_{o,p}) \in [0, 4\pi)^d$. In our own version of the phase part distance function, we remove the denominator. Therefore, the $\boldsymbol{h}_{*,p}$ produced by atan2 is bounded in $[-\frac{\pi}{2}, +\frac{\pi}{2}]$. This modification is essential because the phase boundary amplifies triplets' phase distances, thus making it easier for the decoder to distinguish entities at the same level of the taxonomy.

Loss Function. We adopt the widely used negative sampling loss function [9, 97, 60, 104] with self-adversarial training [79]:

$$L = -\log \sigma(\gamma - d(s, r, o)) - \sum_{i=1}^{n} p(s'_i, r, o'_i) \log \sigma(d(s'_i, r, o'_i) - \gamma), \tag{3.11}$$

where σ is the sigmoid function, γ is a fixed margin that can be chosen by hyper-parameter tuning, and (s'_i, r, o'_i) represents the ith sampled negative triplet of (s, r, o). The term $p(s'_i, r, o'_i)$ is the sampling probability of the particular negative triplet, which can be calculated by:

$$p(s'_i, r, o'_i) = \frac{\exp(\alpha f_{samp}(s'_i, r, o'_i))}{\sum_j \exp(\alpha f_{samp}(s'_j, r, o'_j))}, \tag{3.12}$$

where α is another hyper-parameter that represents the temperature of negative sampling.

3.3.4 Experiments

In this section, we evaluate our proposed HAKEGCN through performance comparisons, in-depth analysis, and ablation studies.

Experiment Settings

Evaluation Protocols. For the AutoTAXO concept prediction subtask of TAXOKG completion, we choose *Mean Average Precision* (MAP) and *Precision at N* (P@N) as evaluation metrics [13]. MAP is based on top-15 predicted concepts. For the other OpenKG relation prediction subtask, we follow previous KB completion studies [9] to rank candidate entities under the "filtered" protocol, and we choose *Mean Reciprocal Rank* (MRR) and *Hits at N* (H@N) as metrics.

Compared Methods. We adopt the following representative methods as baselines:

- Translation-based: TransE [9], HAKE [104];
- Semantic matching-based: DistMult [97], HolE [60];
- GCN-based: R-GCN [71], CompGCN [82];
- Mutual enhancement-based: LTCAG.

We integrate the same techniques introduced in §3.3.3 to mitigate unseen entities, concepts, and relations for baselines. The detailed introduction of baseline methods and choices of hyperparameters are described in the Appendix Section of paper [49].

Performance Comparisons

Tables 3.6a, 3.6b and 3.6c show the performance of compared models on TAXOKG-BENCH. Our naïve LTCAG model, which requires no training, surprisingly achieves competitive performance to all complicated models except for HAKE in AutoTAXO concept prediction metrics (MAP, P@10,30,50) on all six datasets. Our HAKEGCN consistently outperforms SOTA models on all datasets on both tasks, which demon-

Table 3.6: TaxoKG completion results in different domains. For abbreviations, C-* indicates metrics for **c**oncept prediction, while R-* indicates metrics for **r**elation prediction. Underlined numbers denote the second runners, while bold numbers denote the winner.

(a) General domain.

	MSCG × ReVerb				MSCG × OPIEC			
	C-MAP	C-P@1, 3, 10	R-MRR	R-H@10, 30, 50	C-MAP	C-P@1, 3, 10	R-MRR	R-H@10, 30, 50
TransE	.007	.001, .003, .002	7e-4	8e-4, .002, .004	.006	.004, .002, .001	.002	.001, .004, .008
HAKE	.034	.013, .013, .010	.029	**.065**, **.120**, **.153**	.031	.014, .011, .010	.539	**.787**, **.821**, **.837**
DistMult	.004	.004, .001, 5e-4	.001	3e-4, .004, .006	.001	9e-4, 3e-4, 3e-4	.080	.131, .159, .176
HolE	.007	.003, .003, .002	7e-4	7e-4, .002, .004	.006	.004, .002, .001	.002	.001, .004, .008
R-GCN	.003	5e-4, .001, 8e-4	.001	8e-4, .003, .007	.044	.044, .017, .006	.017	.031, .121, .179
CompGCN	.014	.008, .005, .004	4e-4	2e-4, 6e-4, 8e-4	.004	.003, .002, .001	.011	.025, .051, .067
LtCaG	.005	.003, .002, .002	.001	.002, .003, .004	.003	.002, .001, .001	.002	.002, .006, .009
HakeGCN	**.069**	**.033**, **.028**, **.017**	**.031**	.058, .113, .150	**.070**	**.052**, **.027**, **.014**	**.675**	.756, .805, .832

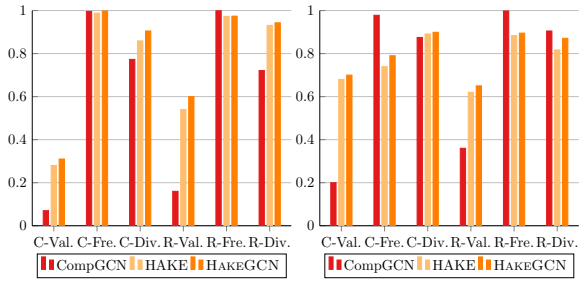
(b) Medical domain.

	SEMedical × ReVerb				SEMedical × OPIEC			
	C-MAP	C-P@1, 3, 10	R-MRR	R-H@10, 30, 50	C-MAP	C-P@1, 3, 10	O-MRR	R-H@10, 30, 50
TransE	.036	.104, .083, .050	.002	.002, .009, .012	.025	.045, .061, .030	.005	.007, .019, .030
HAKE	.203	.307, **.286**, **.216**	.170	.343, .430, .459	.262	.371, .309, **.256**	.352	.450, .509, .544
DistMult	.065	.188, .069, .033	.023	.070, .135, .187	.022	.159, .068, .032	.032	.061, .158, .218
HolE	.029	.063, .063, .044	.002	.002, .005, .009	.024	.091, .030, .027	.006	.007, .018, .032
R-GCN	.024	.018, .041, .052	.001	.001, .003, .004	.036	.159, .062, .037	.004	.003, .016, .026
CompGCN	.119	.191, .184, .150	.003	.005, .012, .017	.041	.060, .044, .032	.009	.013, .023, .034
LtCaG	.186	.245, .247, .172	.004	.005, .006, .008	.126	.166, .157, .122	.013	.021, .041, .051
HakeGCN	**.233**	**.331**, .278, .204	**.275**	**.424**, **.545**, **.603**	**.271**	**.377**, **.366**, .251	**.412**	**.508**, **.600**, **.652**

(c) Music domain.

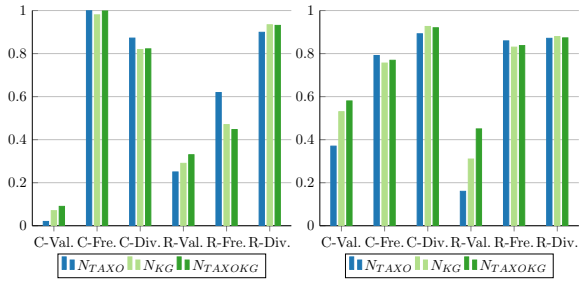
	SEMusic × ReVerb				SEMusic × OPIEC			
	C-MAP	C-P@1, 3, 10	R-MRR	R-H@10, 30, 50	C-MAP	C-P@1, 3, 10	R-MRR	R-H@10, 30, 50
TransE	.012	.053, .035, .028	.002	.002, .006, .009	.041	.123, .082, .064	.002	.003, .008, .013
HAKE	.201	.275, .270, .210	.131	.258, .344, .382	.284	.379, .363, .294	.321	.497, .612, .669
DistMult	.035	.118, .092, .066	.019	.039, .123, .188	.047	.086, .078, .081	.017	.044, .092, .124
HolE	.038	.118, .092, .066	.002	.002, .004, .007	.028	.062, .066, .043	.003	.003, .008, .015
R-GCN	.005	.011, .010, .013	8e-4	7e-4, .002, .003	.014	.021, .039, .034	.002	.001, .005, .008
CompGCN	.063	.092, .111, .095	.009	.019, .034, .042	.082	.199, .161, .112	.005	.012, .023, .036
LtCaG	182	.286, .251, .172	.003	.004, .006, .009	.287	**.426**, .378, .251	.025	.040, .055, .063
HakeGCN	**.238**	**.301**, **.307**, **.221**	**.178**	**.286**, **.412**, **.481**	**.328**	**.426**, **.417**, **.310**	**.421**	**.572**, **.694**, **.746**

strates the substantial advantages of integrating taxonomy and KG to mutually complete each other. There is an obvious pattern when entity and relation numbers grow from hundreds in SEMedical × OPIEC to ten-thousands in MSCG × ReVerb, where all baseline performances drop significantly due to the incapability of unseen entities and relations. HAKE is the second-best model that beats HakeGCN on some metrics in medical and general domain datasets.



(a) MSCG × ReVerb (b) SEMusic × ReVerb

Figure 3.6: In-depth analysis for different models.



(a) MSCG × OPIEC (b) SEMedical × ReVerb

Figure 3.7: In-depth analysis for neighbors impact.

In-depth Analysis

We conduct in-depth analysis on knowledge triplets generated by two strongest baseline models CompGCN, HAKE, and HAKEGCN, using the following manual (Validity) and automated (Freshness, Diversity) evaluating metrics:

- Validity (Val.): whether generated triplets are valid to humans[4].
- Freshness (Fre.): the percentage of generated knowledge triplets that are novel[5].
- Diversity (Div.): Pielou's evenness index[6] which is popular in environment science to represent how equal the phrases in overall produced knowledge triplets is.

We collect results and compute the three metrics on the AutoTAXO concept prediction task by the top-5 predicted concepts, given 100 entities from MSCG × ReVerb and SEMusic × ReVerb. Similarly, we collect results on OpenKG link computed from the top-5 predicted subject or object entities, given 100 triplet queries. In Figure 3.6, the left three grouped bars (C-Val./Fresh./Div.) represent evaluation results of concepts assigned to entities of interest, and the right three stacked bars (R-Val./Fresh./Div.) represent results of generated open knowledge triplets. We observe that HAKEGCN produces the highest quality knowledge triplets. In particular,

[4]The validity scores are annotated by two graduate students, Zishan Gu and Jiaying Lu, and three undergraduate students, Jacob Choi, Leisheng Yu, and Dheep Dalamal.

[5]A triplet not present in original TAXOKG is considered as fresh. Align with the open-world assumption, we treat each unique mention as a unique entity(concept, relation).

[6]Pielou's eveness index: https://en.wikipedia.org/wiki/Species_evenness.

HakeGCN outperforms the two baseline models in both taxonomy and KG validity, with competitive freshness and diversity.

Ablation Studies

Table 3.7: Ablation study results on HakeGCN technical designs.

	SEMedical × ReVerb		SEMdical × OPIEC	
	C-MAP	R-MRR	C-MAP	R-MRR
HakeGCN	**.233**	**.275**	**.271**	**.412**
w/o. taxo_graph_sampling	.154	.268	.151	.376
w/o. polar_conv	.155	.254	.196	.331
w/o. phase_bounded_scorer	.152	.239	.216	.311

Do our technical designs contribute to performance boost? To better understand our proposed techniques, we closely study the key components of HakeGCN. The three components are: taxonomy-based neighbor sampling, polar GCN, and GCN-oriented phase bounded decoder. Table 3.7 presents the results on two medical TaxoKG's with the major metrics for both the AutoTAXO concept prediction (C-MAP) and the OpenKG relation prediction (R-MRR) tasks. For row "w/o. taxo_graph_sampling", we use the uniform neighbor sampling; for "w/o. polar_conv", we use the Cartesian coordinate-based graph convolution; for "w/o. phase_bounded_scorer", we use the existing unbounded score function from HAKE. Table 3.7 supports the effectiveness of proposed techniques, since all three components improve the performance of HakeGCN.

Table 3.8: TaxoKG completion performance when presented with the separated data (SEMedical only or OPIEC only) v.s. the jointed data (SEMedical × OPIEC).

(a) Concept prediction results.

Model	Data	C-MAP	C-P@10, 30, 50
HAKE	AutoTaxo	.186	.344, **.355**, .177
HAKE	TaxoKG	**.262**	**.371**, .309, **.256**
CompGCN	AutoTaxo	**.075**	**.284**, **.117**, **.109**
CompGCN	TaxoKG	.041	.060, .044, .032
HakeGCN	AutoTaxo	.105	.093, .093, .123
HakeGCN	TaxoKG	**.271**	**.377**, **.366**, **.251**

(b) Relation prediction results.

Model	Data	R-MRR	R-H@10, 30, 50
HAKE	OKG	.350	**.454**, **.517**, **.545**
HAKE	TaxoKG	**.352**	.450, .509, .544
CompGCN	OKG	.006	.012, **.030**, **.049**
CompGCN	TaxoKG	**.009**	**.013**, .023, .034
HakeGCN	OKG	.375	.478, .555, .607
HakeGCN	TaxoKG	**.412**	**.508**, **.600**, **.652**

Can taxonomy and KG mutually enhance each other? To support the utility of TaxoKG integration, we further conduct ablation study on the taxonomy completion (concept prediction task) and KG completion (relation prediction task) performance when models are presented with only separated data instead of the jointed data of TaxoKG. The results clearly show the significant benefit of jointly modeling existing TAXOs and KGs. Specifically, our HakeGCN is the most effective one in leveraging such joined data of TaxoKGs (consistently achieving the most gains running on TaxoKGs over KGs and TAXOs only).

How do taxonomic and non-taxonomic neighbors impact the experiments? We further analyze the impact of neighbor information from AutoTAXOs and OpenKGs. In Figure 3.7, we plot the in-depth evaluation results of HakeGCN when using neighbors on AutoTAXOs alone (N_{TAXO}), OpenKGs alone (N_{KG}), and both AutoTAXOs and OpenKGs (N_{TAXOKG}). For the GCN encoder, N_{TAXO} is implemented by removing all taxonomic relation edges in the input graph, and N_{KG} by removing all non-taxonomic relation edges. The metrics and notations are the same as Figure 3.6. As can be seen from Figure 3.7, using only one type of neighbors does not significantly impact the freshness and diversity. In contrast, using both types of neighbors from taxonomy and KG can produce more valid knowledge triplets (*e.g.* improving from 0.02/0.07 to 0.09 in MSCG $\times$ OPIEC and from 0.37/0.53 to 0.58 in SEMedical $\times$ ReVerb). Such results clearly demonstrate the substantial mutual enhancement between the taxonomy and KG towards the completion of TaxoKG.

Chapter 4

Applications of Structured Knowledge

4.1 Background

So far, I have covered my research endeavors in learning to construct two specific types of structured knowledge: concept maps (Ch. 2) and taxonomic knowledge graphs (Ch. 3). In this chapter, I further introduce structured knowledge on diverse novel applications. I consider the term "structured knowledge" in the broadest way. Therefore, it includes but is not limited to:

- semantic graph, abstract meaning graph, syntactic parsing graph;
- concept map;
- taxonomy, ontology, hierarchy;
- knowledge graph, knowledge base, (structured) database;
- social network, community network;
- brain network, biological network.

One of my first-author works are included. It is "MuG: A Multimodal Classification Benchmark on Game Data with Tabular, Textual, and Visual Fields." [51] (Findings of EMNLP'23), where we propose a multimodal sample-similarity-based graph approach for multimodal classification task. Moreover, I have collaborated with other researchers on leveraging concept maps to help COVID-19 document retrival [18], and a survey on knowledge graphs for healthcare applications [19].

4.2 MuGNet: A Sample-Similarity-Based Graph Neural Network for Multimodal Classification

We propose MuGNet [51], a sample-similarity-based graph neural network (GNN) for multimodal classification. MuGNet dynamically constructs community graphs based on sample similarity and effectively combines graphical representation learning with multimodal fusion. In our novel multimodal classification benchmark MuG, MuGNet achieve computability performance with state-of-the-art transformer-based multimodal classifiers with more efficient throughput.

4.2.1 Problem Definition

Given a finite set of categories $\mathcal{Y}$ and labeled training pairs $(x_i, y_i) \in \mathcal{X}_L \times \mathcal{Y}$, multimodal classification aims at finding a classifier $\hat{f} : \mathcal{X}_L \rightarrow \mathcal{Y}$ such that $\hat{y}_j = \hat{f}(x_j)$ is a good approxmiation of the unknown label y_j for unseen sample $x_j \in \mathcal{X}_U$. It is worth noting that the each multimodal sample $x \in \mathcal{X}_L \cup \mathcal{X}_U$ consists of tabular fields t, textual fields s, and image fields i (*i.e.*, $x = \{t, s, i\}$).

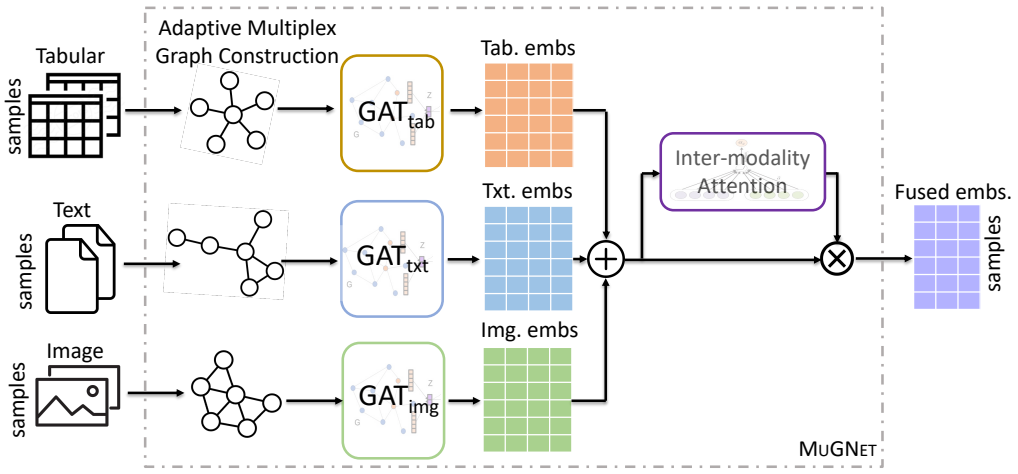


Figure 4.1: Model architecture of MuGNet.

4.2.2 MuGNet

MuGNet is our own multimodal classifier which is further proposed as a competitor to existing models. We propose three key components to make MuGNet a powerful graph neural network for the multimodal classification task. They are adaptive multiplex graph construction module, GAT encoder module, and attention-based fusion module, as shown in Figure 4.1. Firstly, adaptive multiplex graphs are constructed to reflect sample-wise similarity within each modality. Then, separate GAT encoders [83] are employed to obtain dense embeddings of samples, by propagating information between neighbors. Finally, tabular, text and image embeddings are combined by inter-modality attention to obtaining the fused embedding for multimodal classification. GNNs [99, 32] show great capability to leverage the graph structure, propagate information, integrate features, and capture higher-order relationships. This leads to accurate and robust classification performance across various domains. In this work, we propose to regard the whole samples as a correlation network [89, 29] that represents sample-to-sample similarities, while existing multimodal classifiers rarely consider this before.

Adaptive multiplex graph construction module. Following the notation defined in Ch.4.2.1, the adaptive multiplex graph construction module first utilizes

pre-processing pipelines (*e.g.*, monotonically increasing integer mapping for categorical inputs, no alteration for numerical inputs) or pre-trained feature extractors (*e.g.*, CLIP [65] for text and image inputs) to obtain dense multimodal features $\mathcal{F} = f(\mathcal{X}_L) \in \mathbb{R}^{N\times(d^t+d^s+d^i)}$, where $\mathcal{F} = \{\mathcal{F}^t, \mathcal{F}^s, \mathcal{F}^i\}$ denotes feature matrices for tabular, text, and image modalities. The adaptive multiplex graph construction module then derives multiplex sample-wise similarity graph $\mathcal{G} = \{\mathcal{G}^t, \mathcal{G}^s, \mathcal{G}^i\} = \{(\mathcal{A}^t, \mathcal{F}^t), (\mathcal{A}^s, \mathcal{F}^s), (\mathcal{A}^i, \mathcal{F}^i)\}$, where each modality-specific adjacency matrix $\mathcal{A}^m \in \mathbb{R}^{N\times N}, \forall m \in \{t, s, i\}$ is calculated based on the multimodal features

$$\mathcal{A}^m_{i,j} = \text{sim}(\mathcal{F}^m_i, \mathcal{F}^m_j). \tag{4.1}$$

It is worth noting that the sample-wise similarity function sim is adaptive, and is chosen from *cosine similarity*, *radial basis function (RBF) kernel*, or *k-nearest neighbor*. For these modality-specific graphs, we use separate hyperparameters (*e.g.*, threshold for score-based functions, or the value of k for k-nearest neighbor) to control their sparsity properties. The similarity function and its associated hyperparameters are determined through hyperparameter tuning [42] on the held-out validation set, so that the multiplex graph construction is adaptive to any downstream task.

GAT encoder module. We use the powerful multi-head graph attention neural network (GAT) [83] as the encoder to obtain structure-aware representations of samples. Separate GATs are employed for each view of the multiple graph, so that $\mathcal{H}^m = GAT(\mathcal{A}^m, \mathcal{F}^m; \theta)$, where $\mathcal{H}^m \in \mathbb{R}^{N\times d^m_h}$, and θ represents the learnable parameters of the GAT encoder. We want to state there is no information leakage in MuGNet, because we follow the inductive learning setting of GNNs [34] where the GAT encoder is trained on the multiplex graph $\mathcal{G}$ derived from labeled training samples $\mathcal{X}_L$, and new unseen multiplex graph is derived from all samples $\mathcal{X}_L \cup \mathcal{X}_U$ at the inference stage. Furthermore, we adopt a graph sampling technique (Graph-

SAINT [102]) during the GAT training process, to improve the efficiency and generalization. The graph sampling technique essentially samples a subgraph by random walks for each training step, thus the "neighbor explosion" issue is alleviated with a constrained number of neighbors per node and the variance of GAT is reduced with fewer outliers or noise in the sampled graph.

Attention-based fusion module. After we obtain the structure-aware embeddings of samples from the tabular, text, and image modalities $\mathcal{H}^t, \mathcal{H}^s, \mathcal{H}^i$, the attention-based fusion module is responsible for fusing them into one single embedding via the attention-based fusion module. The attention weight $\alpha_j^m \in \mathbb{R}$ for j-th sample of modality m is computed as:

$$\alpha_j^m = \frac{\exp(e_j^m)}{\sum_{m' \in \{t,s,j\}} \exp(e_j^{m'})}, \tag{4.2}$$

$$e_j^m = \boldsymbol{w}_{a_2} \cdot \tanh(\boldsymbol{W}_{a_1}^m \boldsymbol{h}_j^m), \tag{4.3}$$

where $e_j^m \in \mathbb{R}$ denotes the unnormalized attention weight, $\boldsymbol{w}_{a_2} \in \mathbb{R}^{d_a^m \times 1}, \boldsymbol{W}_{a_1} \in \mathbb{R}^{d_h^m \times d_a^m}$ denote learnable parameters, and $\boldsymbol{h}_j^m \in \mathbb{R}^{d_h^m}$ denotes the j-th row of $\mathcal{H}^m$ (*i.e.*, embedding of j-th sample of modality m). The fused embedding of j-th sample is then calculated by:

$$\boldsymbol{h}_j = \alpha_j^t \boldsymbol{h}_j^t + \alpha_j^s \boldsymbol{h}_j^s + \alpha_j^i \boldsymbol{h}_i^i. \tag{4.4}$$

The fused embedding $\boldsymbol{h}_j$ incorporates cross-modalities interactions and provides a complete context for the downstream tasks. An additional two-layer MLP is trained to predict the category of j-th sample $\hat{y}_j = \text{softmax}(\boldsymbol{W}_{\text{cls}_2} \cdot \text{LeakyReLU}(\boldsymbol{W}_{\text{cls}_1} \boldsymbol{h}_j))$. We adopt cross-entropy between prediction $\hat{y}$ and target y as MuGNet's loss function.

4.2.3 Experiments

Datasets

We create and release MuG with eight datasets for multimodal classification with tabular, text, and image fields to the community for future studies. Raw data and examples of how to appropriately load the data are provided in https://github.com/lujiaying/MUG-Bench. MuG is under the "CC BY-NC-SA 4.0" license[1], and is designated to use for research purposes. For a comprehensive introduction of creation process and analysis for MuG, we refer interested readers to our EMNLP paper [51].

Dataset	Game	Pred. Target	#Row	#Class	#Feat (tab/txt/img)
pkm_t1	Pokémon	Primary **T**ype	719/45/133	18	23 (17/5/1)
pkm_t2	Pokémon	Secondary **T**ype	719/45/133	19	23 (17/5/1)
hs_ac	HearthStone	**A**ll card's **C**ategory	8569/536/1605	14	18 (12/5/1)
hs_as	HearthStone	**A**ll card's **S**et	8566/533/1607	38	18 (12/5/1)
hs_mr	HearthStone	**M**inion card's **R**ace	5421/338/1017	16	13 (7/5/1)
hs_ss	HearthStone	**S**pell card's **S**chool	2175/170/508	8	11 (5/5/1)
lol_sc	LoL	**S**kin **C**ategory	1000/64/188	7	11 (3/7/1)
csg_sq	CS:GO	**S**kin **Q**uality	766/49/141	6	7 (5/1/1)

Table 4.1: The statistics of the eight datasets in MuG.

Performance Comparisons

Table 4.2a and 4.2b show the performance of all evaluated models on MuG. As can be seen, multimodal classifiers (except AutoMM) consistently outperform unimodal classifiers in both log-loss and accuracy. It demonstrates that the classification tasks in MuG are multimodal-dependent where each modality only conveys partial information about the required outputs. Among the three multimodal classifiers we used, AutoGluon and MuGNet are the top-2 models with well-matched performances. In Table 4.2a and 4.2b, AutoGluon achieves the best performance eight times, while MuGNet also achieves the best performance eight times. More specifically, Auto-

[1]CC BY-NC-SA 4.0: https://creativecommons.org/licenses/by-nc-sa/4.0/

Method	pkm_t1	pkm_t2	hs_ac	hs_as	hs_mr	hs_ss	lol_sc	csg_sq
	Unimodal Classifiers							
GBM	1.838	2.038	0.911	2.352	0.913	0.603	0.198	1.107
tabMLP	1.442	1.909	1.172	2.155	1.247	0.672	0.533	0.718
RoBERTa	1.834	2.191	1.999	2.393	1.920	1.254	0.847	0.734
Electra	2.907	2.179	2.118	3.155	2.085	1.263	0.611	0.757
ViT	3.680	2.543	1.527	2.786	1.032	2.056	2.049	0.835
Swin	2.657	2.229	2.018	2.795	2.089	1.397	1.470	0.750
	Multimodal Classifiers							
AutoGluon	**0.973**	1.507	**0.654**	1.793	0.403	**0.350**	**0.159**	**0.631**
AutoMM	1.736	2.029	1.987	2.193	1.836	1.320	0.792	0.674
MuGNet	1.000	**1.499**	0.922	**1.499**	**0.321**	0.442	0.248	0.654

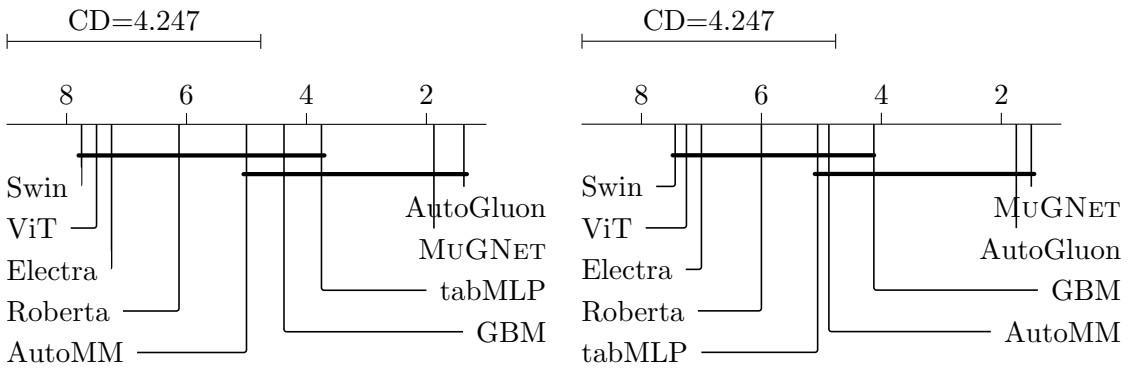
(a) Results in 'log-loss' (the less the better).

Method	pkm_t1	pkm_t2	hs_ac	hs_as	hs_mr	hs_ss	lol_sc	csg_sq
	Unimodal Classifiers							
GBM	0.489	0.489	0.726	0.421	0.737	0.795	0.963	0.610
tabMLP	0.662	0.481	0.627	0.377	0.617	0.776	0.851	0.681
Roberta	0.662	0.466	0.475	0.366	0.535	0.683	0.883	0.688
Electra	0.120	0.466	0.475	0.168	0.535	0.683	0.878	0.702
ViT	0.308	0.406	0.568	0.236	0.787	0.593	0.436	0.674
Swin	0.346	0.451	0.470	0.248	0.536	0.657	0.431	0.702
	Multimodal Classifiers							
AutoGluon	0.744	0.617	**0.787**	0.495	0.879	**0.882**	0.963	**0.766**
AutoMM	0.639	0.511	0.475	0.415	0.549	0.671	0.888	0.738
MuGNet	**0.774**	**0.669**	0.724	**0.572**	**0.908**	0.880	**0.968**	0.745

(b) Result in 'accuracy' (the more the better).

Table 4.2: Overall experimental results with explicit modality performance. The bold text represents the best performance and the underlined text represents the runner-up performance.

Gluon is superior in log-loss whereas MuGNet has better accuracy scores. AutoMM performs the worst among multimodal classifiers, and it sometimes underperforms unimodal classifiers. Considering that AutoMM trains powerful deep neural networks on a small scale of datasets and we have observed the gap between the training loss and validation loss, it is highly possible that AutoMM is overfitting. While AutoGluon and MuGNet also adopt deep neural networks as base models, they are



(a) Log-loss. (b) Accuracy.

Figure 4.2: The critical difference diagrams show the mean ranks of each model for the test data of the eight datasets. The lower rank (further to the right) represents the better performance of a model. Groups of models that are not significantly different ($p < 0.05$) are connected by thick lines.

more robust since AutoGluon proposes a repeated bagging strategy and MuGNet utilizes graph sampling techniques to avoid overfitting. Among unimodal classifiers, tabular models seem to outperform textual and visual models in most cases (six out of eight datasets). There is a slight performance gain comparing textual models to visual models because textual models are better on five datasets.

To better understand the overall performance of models across multiple datasets, we propose using critical difference (CD) diagrams [20]. In a CD diagram, the average rank of each model and which ranks are statistically significantly different from each other are shown. Figure 4.2a and 4.2b show the CD diagrams using the Friedman test with Nemenyi post-hoc test at $p < 0.05$. In summary, we observe that *AutoGluon* and MuGNet respectively achieve the best rank among all tested models with respect to log-loss and accuracy, although never by a statistically significant margin. Moreover, tabular models obtain higher ranks than other unimodal classifiers. The similar observations from Table 4.2 and Figure 4.2 support that effectively aggregating information across modalities is critical for the multimodal classification task.

Efficiency Evaluations

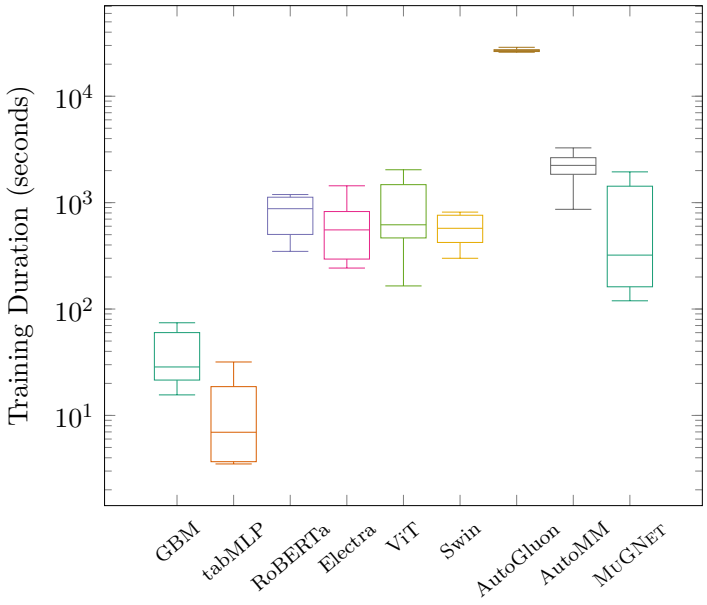


Figure 4.3: Training duration on all datasets.

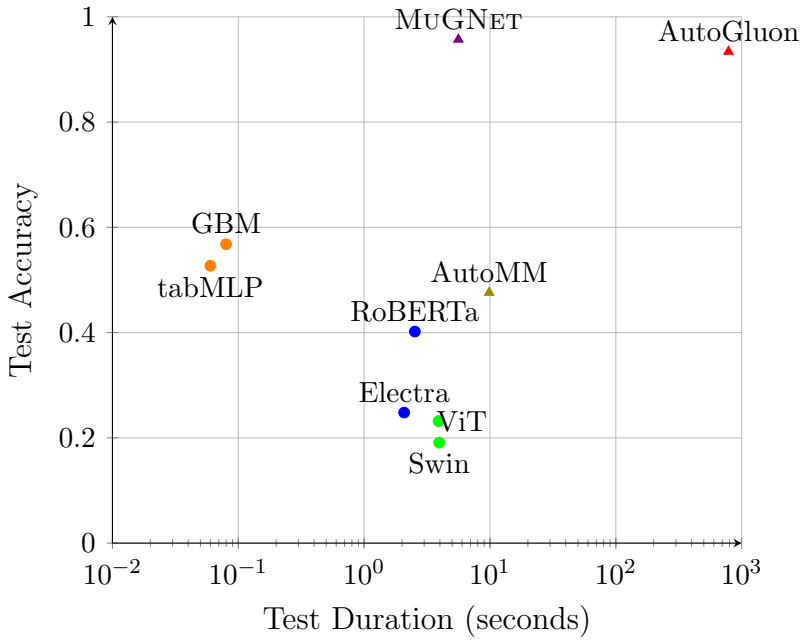


Figure 4.4: Mean testing duration and mean normalized accuracy tradeoffs on all datasets.

Although accuracy (or other metrics such as log-loss in our case) is the central measurement of a machine learning model, efficiency is also a practical requirement

in many applications. Trade-off often exists between *how accurate* the model is and *how long* it takes to train and infer the model. Therefore, we record the training durations and test durations of models to examine their efficiency. In Figure 4.3, we show the aggregated training duration of evaluated models via a box plot. As can be seen, tabular models require an order of magnitude less training duration than the other models, while AutoGluon stands out as requiring significantly longer training duration. Among tabular models, tabMLP is 4x faster than GBM in terms of the median training duration. Except for tabular models and AutoGluon, other models are approximately lightweight to train. It is worth noting that AutoGluon hits the 8-hour training duration constraint on every dataset, thus the variance of its training durations across datasets is very small.

In Figure 4.4, we show the trade-offs between mean inference time and mean accuracy of models. Since the accuracy is not commensurable across datasets, we first normalize all accuracies through a dataset-wise min-max normalization. After the normalization, the best model in each dataset is scaled to 1 while the worst model is scaled to 0. Finally, we take the average on the normalized accuracies and the test durations to draw the scatter plot. When both accuracy and efficiency are objectives models try to improve, there does not exist a model that achieves the best in both objectives simultaneously. As an illustration, MuGNet has the highest test accuracy, but tabMLP has the fastest inference speed. Therefore, we adopt the Pareto-optimal[2] concept to identify which models achieve "optimal" trade-offs. Pareto-optimal is widely used in the decision-making process for multi-objective optimization scenarios. By definition, a solution is Pareto-optimal if any of the objectives cannot be improved without degrading at least one of the other objectives. Following this concept, we observe that tabMLP, GBM, and MuGNet are the models with the best trade-offs between accuracy and efficiency, as these models reside in the Pareto frontier in

[2]Pareto-optimal Definition: https://w.wiki/6sLB

Figure 4.4. Meanwhile, other models are suboptimal with regard to this trade-off, since we can always find a solution that has higher accuracy and better efficiency simultaneously than these models.

4.3 Collaborative Projects on Structured Knowledge for Healthcare Applications

Healthcare data is inherently heterogeneous and multimodal. As a powerful representation and reasoning tool, structured knowledge (or graph in general) can be utilized in empowering healthcare applications. In this direction, we explore the utility of generated concept maps in COVID-19 document retrieval tasks. I am the co-author for this ECIR'22 paper [18], focusing on how Graph Neural Networks can aid in Document Retrieval with a focus on CORD19 and Concept Map Generation. Our experimental results show that semantic-oriented GNNs achieve better performance than structure-oriented GNNs with the help of the unsupervised concept map generation technique I proposed in Chapter. 2.2. For instance, E-Pool and RW-Pool (two proposed semantic-oriented GNNs) improved document retrieval from the initial candidates of BM25 by 11.4% and 12.0% on NDCG@20, respectively. These results highlight the potential of structured knowledge for textual reasoning tasks such as classification and retrieval.

On the other hand, we conduct a comprehensive review of healthcare knowledge graphs. This project has been published as a survey paper [19], titled "*A Review on Knowledge Graphs for Healthcare: Resources, Applications, and Promises*". We cover the definition, construction, and application of healthcare KGs. As can be seen, most works from my dissertation focus on the construction perspective. In this paper, we categorize existing healthcare KG construction into (1) constructing from scratch and (2) constructing by integration. Regarding the application areas, healthcare

KGs are promising in the following areas, including (1) basic science research such as medical chemistry, and bioinformatics; (2) pharmaceutical development such as drug development, and clinical trial; (3) clinical decision support; (4) public health such as epidemiology, environmental health, policy & management, and social & behavioral.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this dissertation, I present a series of my first authored works on constructing structured knowledge and how learned structured knowledge can benefit downstream tasks. I first present novel methods on concept map constructions (Ch.2) and taxonomic knowledge graph constructions (Ch.3). I then present the utility of structured knowledge on the multimodal classification task (Ch.4). All these works share a common spirit of developing supervision-efficient algorithms for structured knowledge construction, to accommodate real-world scenario challenges. The proposed methods are mainly unsupervised (Parsing-based concept map generation in Ch. 2.2), weakly-supervised (GT-D2G in Ch. 2.3), or few-shot-supervised (HiPrompt in Ch. 3.2). I hope this dissertation can be the starting point for inspiring future research in learning structured knowledge that includes but is not limited to concept maps, knowledge bases, social networks, and biological networks.

5.2 Future Work

Combining the research areas I have explored and recent progress in the artificial intelligence communities, the following directions can be investigated in the future:

- **Mutual Enhancement between Structured Knowledge and Large Language Models**: this can be further divided into two sub-directions.
  - Knowledge-enhanced LLMs: Although LLMs have achieved remarkable success and generalizability in various applications, they still suffer from context windows, static knowledge, hallucination, etc. Structured knowledge is a curated data source that explicitly stores high-quality knowledge, and can be periodically updated to catch ever-evolving world knowledge.
  - LLM-enhanced Structured Knowledge: On the other hand, structured knowledge is hard to construct and requires ad-hoc methods to handle incompleteness. LLMs offer a variety of supervision-efficient utilities in many fundamental knowledge-related tasks such named entity recognition, information extraction, knowledge fusion, knowledge completion, etc.
- **Structured Knowledge For Healthcare**: The integration of structured knowledge databases and advanced artificial intelligence models will be pivotal in creating more efficient, accurate, and personalized healthcare solutions. By harnessing structured knowledge, we can empower healthcare professionals with instant access to a vast repository of medical information, facilitating quicker and more accurate diagnoses and treatment recommendations. Additionally, the continued development of knowledge graphs and ontologies will enhance our ability to extract meaningful insights from vast and complex healthcare datasets, enabling advancements in epidemiology, drug discovery, and patient management. Furthermore, structured knowledge will play a crucial role in

the ongoing evolution of telemedicine, as it enables AI-driven decision support systems that can bridge geographical gaps and provide expert guidance in real-time.

Bibliography

[1] Shih-Ming Bai and Shyi-Ming Chen. Automatically constructing concept maps based on fuzzy rules for adapting learning systems. *Expert Syst. Appl.*, 2008.

[2] Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *COLING'1998*, 1998.

[3] Krisztian Balog and Robert Neumayer. Hierarchical target type identification for entity-oriented queries. In *CIKM*, 2012.

[4] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR*, 2013.

[5] Bodo Billerbeck and Justin Zobel. Document expansion versus query expansion for ad-hoc retrieval. In *Proceedings of the 10th Australasian Document Computing Symposium*, 2005.

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *JMLR*, 2003.

[7] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 2004.

[8] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Tay-

lor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[9] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NeurIPS*, 2013.

[10] Samuel Broscheit, Kiril Gashteovski, Yanjie Wang, and Rainer Gemulla. Can we predict new facts with open knowledge graph embeddings? a benchmark for open link prediction. In *ACL*, 2020.

[11] Adam S Brown and Chirag J Patel. A standard database for drug repositioning. *Scientific data*, 2017.

[12] 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.

[13] Jose Camacho-Collados, Claudio Delli Bovi, Luis Espinosa Anke, Sergio Oramas, Tommaso Pasini, Enrico Santus, Vered Shwartz, Roberto Navigli, and Horacio Saggion. Semeval-2018 task 9: Hypernym discovery. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, 2018.

[14] Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *IJCAI*, 2017.

[15] Sheng Chen, Colin FN Cowan, and Peter M Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on neural networks*, 1991.

[16] Shiah Lian Chen, Tienli Liang, Mei Li Lee, and I Chen Liao. Effects of concept map teaching on students' critical thinking and approach to learning and studying. *Journal of Nursing Education*, 2011.

[17] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*, 2014.

[18] Hejie Cui, Jiaying Lu, Yao Ge, and Carl Yang. How can graph neural networks help document retrieval: A case study on cord19 with concept map generation. In *44th European Conference on Information Retrieval - Short Paper*, ECIR'22, Apr. 2022.

[19] Hejie Cui, Jiaying Lu, Shiyu Wang, Ran Xu, Wenjing Ma, Shaojun Yu, Yue Yu, Xuan Kan, Chen Ling, Joyce Ho, et al. A survey on knowledge graphs for healthcare: Resources, applications, and promises. In *ICML 2023 Workshop on Interpretable Machine Learning in Healthcare*, IMLH@ICML'23, 2023.

[20] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 2006.

[21] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, 2019.

[22] Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. Cognitive Graph for Multi-Hop Reading Comprehension at Scale. In *ACL*, 2019.

[23] Xin Luna Dong, Xiang He, Andrey Kan, Xian Li, Yan Liang, Jun Ma, Yifan Ethan Xu, Chenwei Zhang, Tong Zhao, Gabriel Blanco Saldana, et al. Au-

toknow: Self-driving knowledge collection for products of thousands of types. In *SIGKDD*, 2020.

[24] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *EMNLP*, 2011.

[25] Tobias Falke and Iryna Gurevych. GraphDocExplore: A framework for the experimental comparison of graph-based document exploration techniques. In *EMNLP*, 2017.

[26] Tobias Falke, Christian M. Meyer, and Iryna Gurevych. Concept-Map-Based Multi-Document Summarization using Concept Coreference Resolution and Global Importance Optimization. In *IJNLP*, 2017.

[27] Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel F Cruz, and Francisco M Couto. The agreementmakerlight ontology matching system. In *ODBASE*, 2013.

[28] Kiril Gashteovski, Sebastian Wanner, Sven Hertling, Samuel Broscheit, and Rainer Gemulla. Opiec: An open information extraction corpus. In *AKBC*, 2018.

[29] Costa Georgantas and Jonas Richiardi. Multi-view omics translation with multiplex graph neural networks. In *WWW*, 2022.

[30] Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. Re2g: Retrieve, rerank, generate. In *NAACL*, 2022.

[31] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 2005.

[32] Xiawei Guo, Yuhan Quan, Huan Zhao, Quanming Yao, Yong Li, and Weiwei Tu. Tabgnn: Multiplex graph neural network for tabular data prediction. In *dlp-kdd*, 2021.

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

[34] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *NeurIPS*, 2017.

[35] Han He and Jinho D. Choi. Establishing Strong Baselines for the New Decade: Sequence Tagging, Syntactic and Semantic Parsing with BERT. In *FLAIRS*, 2020.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[37] Xiaopeng Huang, Kyeong Yang, and Victor B Lawrence. An efficient data mining approach to concept map generation for adaptive learning. In *ICDM*, 2015.

[38] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*, 2017.

[39] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In *ISWC*, 2011.

[40] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017.

[41] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *ICML 2022 Workshop on Knowledge Retrieval and Language Models*, 2022.

[42] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[43] Fangyu Liu, Ehsan Shareghi, Zaiqiao Meng, Marco Basaldella, and Nigel Collier. Self-alignment pretraining for biomedical entity representations. In *NAACL*, 2021.

[44] Shih-Hwa Liu and Gwo-Guang Lee. Using a concept map knowledge management system to enhance the learning of biology. *Computers & Education*, 2013.

[45] Xiao Liu, Li Mian, Yuxiao Dong, Fanjin Zhang, Jing Zhang, Jie Tang, Peng Zhang, Jibing Gong, and Kuansan Wang. Oag_know: Self-supervised learning for linking knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[46] Xiao Liu, Haoyun Hong, Xinghao Wang, Zeyi Chen, Evgeny Kharlamov, Yuxiao Dong, and Jie Tang. Selfkg: Self-supervised entity alignment in knowledge graphs. In *The Web Conference*, 2022.

[47] Zhengzhong Liu, Chenyan Xiong, Teruko Mitamura, and Eduard Hovy. Automatic event salience identification. In *EMNLP'18*, 2018.

[48] Jiaying Lu and Jinho D. Choi. Evaluation of Unsupervised Entity and Event Salience Estimation. In *Proceedings of the 34rd International Florida Artificial Intelligence Research Society Conference*, FLAIRS'21, May. 2021.

[49] Jiaying Lu and Carl Yang. Open-world taxonomy and knowledge graph co-learning. In *4th Conference on Automated Knowledge Base Construction*, AKBC'22, Nov. 2022.

[50] Jiaying Lu, Xiangjue Dong, and Carl Yang. Weakly supervised concept map generation through task-guided graph translation. *IEEE Transactions on Knowledge and Data Engineering*, 35(10):10871–10883, March 2023.

[51] Jiaying Lu, Yongchen Qian, Shifan Zhao, Yuanzhe Xi, and Carl Yang. Mug: A multimodal classification benchmark on game data with tabular, textual, and visual fields. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Findings-EMNLP'23, Dec. 2023.

[52] Jiaying Lu, Jiaming Shen, Bo Xiong, Wengjing Ma, Staab Steffen, and Carl Yang. Hiprompt: Few-shot biomedical knowledge fusion via hierarchy-oriented prompting. In *46th International ACM SIGIR Conference on Research and Development in Information Retrieval - Short Paper*, SIGIR'23, Apr. 2023.

[53] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *ICLR*, 2017.

[54] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL*, 2014.

[55] Yuning Mao, Tong Zhao, Andrey Kan, Chenwei Zhang, Xin Luna Dong, Christos Faloutsos, and Jiawei Han. Octet: Online catalog taxonomy enrichment with self-supervision. In *SIGKDD*, 2020.

[56] Yoshitomo Matsubara, Thuy Vu, and Alessandro Moschitti. Reranking for efficient transformer-based answer selection. In *SIGIR*, 2020.

[57] Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. The NomBank project: An interim report. In *HLT-NAACL 2004*, 2004.

[58] Rada Mihalcea and Paul Tarau. TextRank: Bringing order into text. In *EMNLP'04*, 2004.

[59] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *EMNLP*, 2004.

[60] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *AAAI*, 2016.

[61] Joseph D Novak. Concept mapping: A useful tool for science education. *Journal of research in science teaching*, 1990.

[62] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, 2019.

[63] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[64] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[65] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[66] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *ACL*, 2013.

[67] Eric Sven Ristad and Peter N Yianilos. Learning string-edit distance. *TPAMI*, 1998.

[68] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 2009.

[69] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 1988.

[70] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*, 2021.

[71] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, 2018.

[72] Lynn M Schriml, James B Munro, Mike Schor, Dustin Olley, Carrie McCracken, Victor Felix, J Allen Baron, Rebecca Jackson, Susan M Bello, Cynthia Bearer, et al. The human disease ontology 2022 update. *Nucleic acids research*, 2022.

[73] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R. Voss, and Jiawei

Han. Automated Phrase Mining from Massive Text Corpora. *IEEE Trans. Knowl. Data Eng.*, 2018.

[74] Jiaming Shen, Wenda Qiu, Yu Meng, Jingbo Shang, Xiang Ren, and Jiawei Han. Taxoclass: Hierarchical multi-label text classification using only class names. In *NAACL*, 2021.

[75] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *EMNLP*, 2020.

[76] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *WWW*, 2007.

[77] Fabian M Suchanek, Serge Abiteboul, and Pierre Senellart. Paris: Probabilistic alignment of relations, instances, and schema. *VLDB*, 2011.

[78] Zequn Sun, Qingheng Zhang, Wei Hu, Chengming Wang, Muhao Chen, Farahnaz Akrami, and Chengkai Li. A benchmarking study of embedding-based entity alignment for knowledge graphs. *VLDB*, 2020.

[79] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2018.

[80] Together. GPT-JT-6B. https://huggingface.co/togethercomputer/GPT-JT-6B-v1, 2023. Accessed on February 14, 2023.

[81] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling Coverage for Neural Machine Translation. In *ACL*, 2016.

[82] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *ICLR*, 2020.

[83] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[84] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *NeurIPS*, 2015.

[85] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 2014.

[86] Lidan Wang, Jimmy Lin, and Donald Metzler. A cascade ranking model for efficient ranked retrieval. In *SIGIR*, 2011.

[87] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

[88] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *TKDE*, 2017.

[89] Tongxin Wang, Wei Shao, Zhi Huang, Haixu Tang, Jie Zhang, Zhengming Ding, and Kun Huang. Mogonet integrates multi-omics data using graph convolutional networks allowing patient classification and biomarker identification. *Nature Communications*, 2021.

[90] Wenbo Wang, Yang Gao, He-Yan Huang, and Yuxiang Zhou. Concept pointer network for abstractive summarization. In *EMNLP-IJCNLP*, 2019.

[91] David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al.

Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, 2018.

[92] Tien-Hsuan Wu, Zhiyong Wu, Ben Kao, and Pengcheng Yin. Towards practical open knowledge base canonicalization. In *CIKM*, 2018.

[93] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. Probase: A probabilistic taxonomy for text understanding. In *SIGMOD*, 2012.

[94] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *ACL*, 1994.

[95] Chengjin Xu, Fenglong Su, Bo Xiong, and Jens Lehmann. Time-aware entity alignment using temporal relational attention. In *WWW*, 2022.

[96] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *ICLR*, 2019.

[97] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*, 2015.

[98] Carl Yang, Jieyu Zhang, Haonan Wang, Bangzheng Li, and Jiawei Han. Neural Concept Map Generation for Effective Document Classification with Interpretable Structured Summarization. In *SIGIR*, 2020.

[99] Carl Yang, Jieyu Zhang, Haonan Wang, Sha Li, Myungwan Kim, Matt Walker, Yiou Xiao, and Jiawei Han. Relation learning on social networks with multi-modal graph edge variational autoencoders. In *WSDM*, 2020.

[100] Rui Ye, Xin Li, Yujie Fang, Hongyu Zang, and Mingzhong Wang. A vectorized relational graph convolutional network for multi-relational network alignment. In *IJCAI*, 2019.

[101] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. In *ICML*, 2018.

[102] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *ICLR*, 2019.

[103] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[104] Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *AAAI*, 2020.

[105] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[106] Chaoyu Zhu, Zhihao Yang, Xiaoqiong Xia, Nan Li, Fan Zhong, and Lei Liu. Multimodal reasoning based on knowledge graph embedding for specific diseases. *Bioinformatics*, 2022.