AUTOMATED TAXONOMY DISCOVERY AND EXPLORATION

BY

JIAMING SHEN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Doctoral Committee:

Professor Jiawei Han, Chair
Professor Heng Ji
Professor ChengXiang Zhai
Dr. Michelle T. Vanni, Army Research Lab

ABSTRACT

In an era of information explosion, people are inundated with vast amounts of text data. Every day, there are thousands of scientific papers, tens of thousands of news articles, corporate reports, and millions of social media posts produced and shared worldwide. Turning those massive text data into actionable knowledge is an essential research issue in data science and lays the foundation for realizing machine intelligence.

The goal of my research is to unleash hidden knowledge buried in unstructured text. To bring this vision to reality, I propose to first structure raw text using taxonomies and then analyze structured text in a more fine-grained and semantic way. Due to the diversity of application scenarios, different corpora or different use cases may call for different taxonomies. For example, one analyst aiming to find experts in different scientific areas may want a field-of-study taxonomy, while another analyst who studies the technology readiness may call for a taxonomy capturing technology dependencies. Moreover, even within one taxonomy, we also enable users to organize concepts at their will, such as with different levels containing concepts of different categories. For instance, in a computer science taxonomy, top levels could be about *field of studies*, intermediate levels may discuss *research tasks*, and the bottom levels can cover *evaluation metrics*. Asking human experts to manually curate those taxonomies, one for every possible application, is time-consuming, costly, and unscalable. Therefore, we propose to automatically discover and explore taxonomies based on the datasets and applications, with critical but minimal human guidance.

This thesis outlines a data-driven approach that automatically constructs, enriches, and applies taxonomies for unleashing knowledge from massive unstructured text. Particularly, we investigate four areas of research, including:

1. **Identifying Concept Sets.** To obtain concept nodes in the taxonomy, we first develop a collection of concept set expansion methods [1, 2] to extract concepts from text corpora by expanding a small set of seed concepts into a complete list of concepts that belong to the same semantic class.

2. **Recognizing Taxonomic Relations.** To organize above identified concepts into hierarchical structure, we propose a set of taxonomy construction methods [3, 4] to discover taxonomic relations among concepts by analyzing example relation instances (*i.e.*, concept pairs indicating the target relation semantics) and utilizing distant supervision from existing, open-domain knowledge bases.

3. **Enriching Existing Taxonomies.** As human knowledge is constantly growing, a static taxonomy may fail to capture emerging user needs. Thus, a taxonomy enrichment step would be essential to keep our taxonomies up-to-date in real-world applications. We facilitate this process by expanding the taxonomy to incorporate new concepts [5, 6, 7].

4. **Empowering Knowledge-centric Applications.** After an up-to-date taxonomy is obtained, we develop principled methods to distill knowledge from taxonomies for downstream applications such as text categorization [8, 9] and intelligent literature search [10, 11].

Finally, we explore how to incorporate event knowledge into the taxonomy by automatically detecting event types from a given corpus. Together, these pieces constitute an integrated framework for leveraging taxonomies to convert massive text data into actionable knowledge.

To my family for their love and support.

ACKNOWLEDGMENTS

First and foremost, I want to thank my Ph.D. advisor, Professor Jiawei Han, who is my career and life role model. I still recall the New Year's day of 2016 when Prof. Han shot me an email and invited me to have a discussion over Skype. Since then, we have communicated over 2000 emails and countless meetings through which he teaches me essentials skills to become a mature researcher, from identifying important research problems, to developing rigorous solutions, to collaborating with other researchers, and to communicating ideas clearly. He has also been extremely supportive and reachable, ready for discussions even in the evenings or over weekends. I recall for my first research work SetExpan, he stayed late night and rewrote almost the whole paper to make it clear and accessible. This study is later accepted in ECMLPKDD'17, which significantly boosts my confidence on research and sheds a light on one of the darkest moments in my Ph.D. journey. While Jiawei always praises us for working hard, I deeply believe he is one of the most dedicated and productive people I have ever met and serves as the best example for everyone in our group. My last year at University of Illinois at Urbana-Champaign (UIUC) collides with the COVID-19 breakout and thus I have to work remotely, which further increases the anxiety of job search. During this process, Jiawei provides tremendous helps, by providing timely feedback on my job talk slides, connecting me to relevant folks at different places, and giving me tips for interviews. I have learned from him the courage to take challenges, the positive attitude in front of setbacks, the diligence and self-discipline, and many other characteristics. Such influences will be a forever treasure in my life.

I would also like to thank all my other thesis committee members, Prof. Heng Ji, Prof. Cheng-Xiang Zhai, and Dr. Michelle T. Vanni. Particularly, I would like to thank Cheng for giving me lots of detailed comments on this thesis and asking insightful questions during my thesis proposal and defense. I am very grateful to Heng and Michelle for not only sitting on my dissertation committee, but also advising me through my job application process, answering a lot of questions, and suggesting interesting future directions for my work.

I also want to express my deep gratitude to my advisors and mentors along the way: Prof. Ya Zhang who opens the gate to data mining research for me, Prof. Xinbing Wang who provides a great platform and inspires me to purse a Ph.D. degree abroad, Dr. Maryam Karimzadehgan who hosts my first internship at Google, Dr. Zhihong Shen who helps me explore different styles of doing research at Microsoft Research, and Dr. Jialu Liu who shows me how to adapt research methods to solve industry-scale problems at Google Research.

Moreover, I want to thank Zhen Qin, Donald Metzler, Chi Wang, Chenyan Xiong, Kuan-san Wang, Hongkun Yu, Tianqi Liu and Cong Yu, for their guidance during my internships.

I am fortunate to join the Data Mining Group (DMG) and really enjoy the time working with the members in DMG at UIUC. I want to thank all my friends in the group: Xiang Ren (for teaching me how to position an idea in the paper and how to polish the paper writing), Jingbo Shang (for all the discussions we had on many research ideas and sharing your views on future directions), Chao Zhang (for the advice on how to write papers in a concise and clear way), Liyuan Liu (for showing me different ways of thinking about solution and styles of doing research). I would also like to thank all the master students, Ph.D. students, and Postdoctoral researchers that I am honored to work with. Alphabetically, they are: Xiusi Chen, Junyi Du, Xiaotao Gu, Fang Guo, Xinwei He, Jiaxin Huang, He Jiang, Meng Jiang, Dongming Lei, Keqian Li, Sha Li, Yinghao Li, Yuchen Lin, Ruiliang Lyu, Yuning Mao, Yu Meng, Wenda Qiu, Meng Qu, Dominic Seyler, Yu Shi, Tianhang Sun, Fangbo Tao, Xuan Wang, Ellen Wu, Jinfeng Xiao, Yiqing Xie, Carl Yang, Yue Yu, Hanwen Zha, Yunyi Zhang, Qi Zhu and Wanzheng Zhu. Working with those amazing collaborators makes my Ph.D. study a much joyful journey. Special thanks to my partner, Mei Wang, for being patient and supporting me, and for bring many laughters during my most anxious job search moments.

Last but not least, I own many thanks to my wonderful parents, Yan Shi and Hong Shen, who have always been there during the whole process. They give me unconditional love and supports, provide encouragements and advices, and celebrate with me towards every joy.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 OVERVIEW

Born in an era of information explosion, we are inevitably inundated with vast amounts of text data. Every day, there are thousands of scientific papers, tens of thousands of corporate reports, product reviews, and millions of social media posts produced and shared worldwide. Their volume itself is massive, and more importantly, it keeps growing. When properly analyzed, these text data can be game-changing for science, engineering, business intelligence, policy design, e-commerce, and more. Consequently, turning those massive text data into actionable knowledge is an essential research issue in data science and lays the foundation for realizing machine intelligence.

With massive unstructured text stored or streaming in dynamically, we realize an important methodology for turning them into knowledge is to first 'structure' them. Instead of working on unstructured raw text directly, we can first utilize a taxonomy to structure them and then analyze structured text corpora in a more fine-grained and semantic way. This strategy can significantly accelerate the knowledge discovery process and enable machines to digest the knowledge in an efficient and effective way.

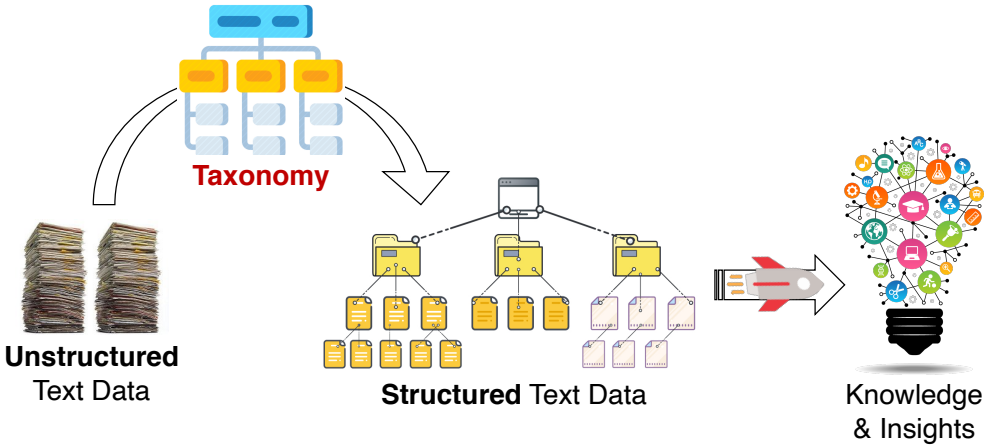


Figure 1.1: Turning unstructured text data to knowledge and insights by utilizing a taxonomy to structure raw text.

There are many existing taxonomies in different domains (*e.g.*, MeSH [12], ACM CCS [13], Pinterest Taxonomy [14], *etc.*). Most of these taxonomies, however, are curated by human exports, which is costly, time-consuming, and non-scalable. More importantly, even with massive human labeling efforts, we could eventually only obtain one taxonomy but a single

taxonomy cannot fit all applications. For example, given the same set of computer science literature, one user may want to identify experts in different fields, and thus calls for a field-of-studies taxonomy, while another person may want to analyze the readiness of different technologies, and thus needs a taxonomy capturing technology dependencies. Naturally, we need to build taxonomies based on the corpora to be analyzed and the applications to be explored. Sometimes, even for the same application, different domain experts may have their own points of view and thus we should ideally enable them to organize concepts at their will, such as with different levels including concepts of different categories. For example, in the computer science domain, top levels could be about *field of studies* (*e.g.*, data mining, natural language processing, *etc.*), immediate levels may discuss *research tasks* (*e.g.*, outlier detection, machine translation, *etc.*), and the bottom levels cover *evaluation metrics* (*e.g.*, F1 score, NDCG, BLUE, *etc.*). As a result, it is unrealistic to ask humans to create a taxonomy for each application.

In this thesis, our goal is to develop principled methods to automatically construct, enrich, and explore taxonomies for knowledge discovery from text data. Our methods alleviate the need for heavy human annotations by utilizing *distant supervision* from existing, open knowledge bases, *weak supervision* from a few user-provided examples, and *self supervision* from signals in massive unlabeled data. Particularly, we investigate four areas of studies,

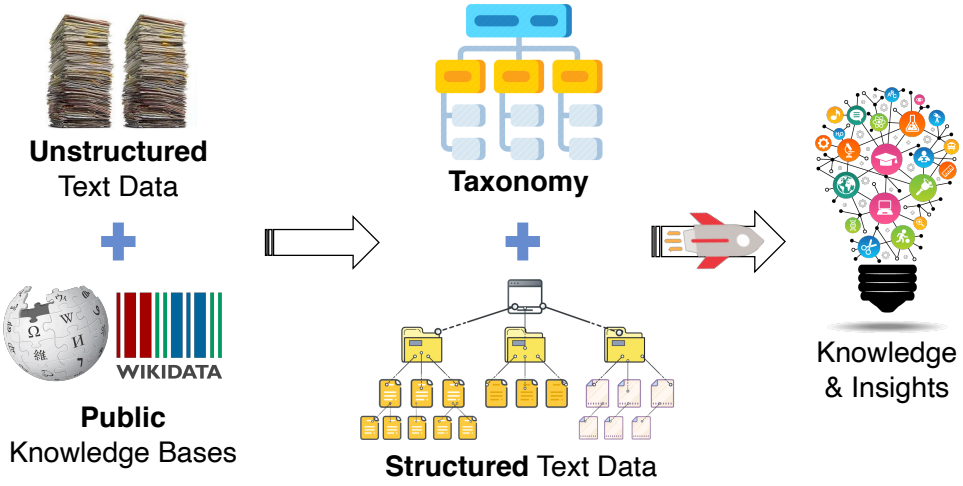


Figure 1.2: Utilizing automatically constructed taxonomy for converting text data into actionable knowledge and insights.

including: (1) **Identifying Concept Sets.** To obtain concept nodes in the taxonomy, we first develop a collection of concept set expansion methods [1, 2, 15] to extract concepts from text corpora by expanding a small set of seed concepts into a complete list of concepts that belong to the same semantic class; (2) **Recognizing Taxonomic Relations.** To

organize above identified concepts into hierarchical structure, we propose a set of taxonomy construction methods [3, 4] to discover taxonomic relations among concepts by analyzing example relation instances (*i.e.*, concept pairs indicating the target relation semantics) and utilizing distant supervision from existing, open-domain knowledge bases; (3) **Enriching Existing Taxonomies.** As human knowledge is constantly growing, a static taxonomy may fail to capture emerging user needs. Thus, a taxonomy enrichment step would be essential to keep our taxonomies up-to-date in real-world applications. We facilitate this process by expanding the taxonomy to incorporate new concepts [5, 6]. (4) **Empowering Knowledge-centric Applications.** When an up-to-date taxonomy is obtained, we develop methods to distill knowledge from taxonomies for downstream applications such as text categorization [8, 9] and intelligent literature search [10, 11]. Besides, we also explore how to incorporate event knowledge into the taxonomy by automatically identifying event types from a given corpus. Together, these pieces constitute an integrated framework for leveraging taxonomies to convert massive text data into actionable knowledge, as shown in Figure 1.3.

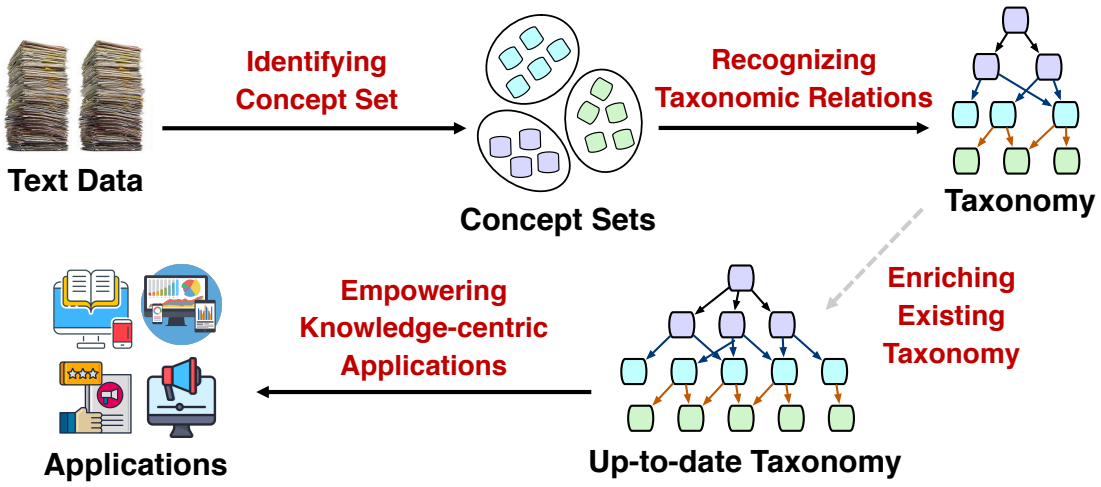


Figure 1.3: An integrated framework for leveraging the taxonomy to unleash hidden knowledge buried in massive unstructured text.

Finally, we want to emphasize that although the taxonomy structure discussed in this thesis shares some commonalities with knowledge graph [16], these two structured data formats have different focuses. Taxonomies capture more abstract concept-level ideas (*e.g.*, one research area, one type of diseases, *etc.*) while knowledge graphs contain more information about concrete entity-level things (*e.g.*, a celebrity, a famous institution, *etc.*). In many cases, taxonomies and knowledge graphs contain complementary world knowledge and they have the potentials to mutually enhance each other.

1.2 TECHNICAL CONTRIBUTIONS

1.2.1 Concept Set Expansion

The first step towards building a corpus- and application-specific concept taxonomy is to identify a set of user-interested concepts from the given corpus. We accomplish this goal by proposing SetExpan [1], a concept set expansion method that automatically expands a small set of seed concepts into a completed list of concepts belong to the same user-interested category. SetExpan is an iterative algorithm and includes two core techniques. The first one is a context feature selection method that chooses clean context features for calculating concept-concept distributional similarity, and The second technique is a ranking-based unsupervised ensemble method for expanding the concept set based on selected context features. Experiments have shown that SetExpan can select high-quality and interpretable context features and outperforms previous best methods by more than 32.6% for concept identification accuracy. Recently, together with my group members, we have further enhanced SetExpan in two ways: (1) leveraging multiple negative sets to guard each other and avoid semantic drifting [17], and (2) incorporating information from pre-trained language models to compensate for the weak supervision signals from the seed set [2].

Contributions.

- We propose SetExpan, a novel iterative algorithm for concept set expansion.
- We develop an unsupervised ranking-based ensemble approach for selecting high quality concepts in a robust way.
- We demonstrate the effectiveness and efficiency of our methods and show improvements over prior methods on multiple real-world datasets in different domains (news articles, Wikipedia articles, and scientific papers).

1.2.2 Taxonomy Construction

After obtaining user-interested concept sets, we continue to organize them into a taxonomy structure. Most of previous taxonomy construction work [18, 19, 20] build taxonomies based on "is-A" relations (*e.g.*, a "*panda*" is a "*mammal*" and a "*mammal*" is an "*animal*") by first leveraging pattern-based or distributional methods to extract hypernym-hyponym term pairs and then organizing them into a tree-structured hierarchy. However, such hierarchies cannot satisfy many real-world needs due to its (1) *inflexible semantics*: many applications may need hierarchies carrying more flexible semantics such as "*city-state-country*" in a location taxonomy; and (2) *limited applicability:* the "universal" taxonomy so constructed is unlikely

to fit diverse and user-specific application tasks.

We propose HiExpan [3], the first *task-guided* concept taxonomy construction method which takes a user-provided "seed" taxonomy tree (as task guidance) along with a domain-specific corpus and generates a desired taxonomy automatically. For example, a user may provide a seed taxonomy containing only two countries and two states along with a large corpus, and HiExpan will output a taxonomy which covers all the countries and states mentioned in the corpus. HiExpan captures weak supervision signals in the seed taxonomy and iteratively expands the seed taxonomy into a fully-fledged taxonomy in a top-down hierarchical way. Specifically, HiExpan views all children under each taxonomy node forming a coherent set and builds the taxonomy by recursively expanding all these sets. Furthermore, HiExpan incorporates a weakly-supervised relation extraction module to extract the initial children of a newly-expanded node and adjusts the taxonomy tree by optimizing its global structure. Experiments have demonstrated the effectiveness of HiExpan for building meaningful taxonomies in various domains, including news, computer science, and life science.

Contributions.

- We study and formulate a new research problem *task-guided taxonomy construction*, which takes a user-provided seed taxonomy along with a domain-specific corpus as input and aims to output a desired taxonomy that satisfies user-specific application tasks.
- We propose HiExpan, a novel expansion-based weakly-supervised framework for task-guided taxonomy construction.
- We conduct extensive experiments to verify the effectiveness of HiExpan on three real-world datasets from different domains.

1.2.3 Taxonomy Enrichment

As human knowledge is constantly growing, it is necessary to expand or enrich an existing concept taxonomy to incorporate new knowledge and be adapted to real-world applications. One naive solution is to re-run the entire taxonomy construction process from scratch. Although being intuitive, this approach has several limitations. First, many taxonomies have a top-level design provided by domain experts and such design shall be preserved. Second, a newly constructed taxonomy may not be consistent with the old one, which can lead to instabilities of its dependent downstream applications. Finally, as targeting the scenario of building taxonomy from scratch, most previous methods are unsupervised and cannot leverage signals from the existing taxonomy to construct a new one.

We propose TaxoExpan [5] to tackles taxonomy expansion problem — the process of automatically incorporating new concepts into an existing taxonomy. A key challenge for

concept taxonomy expansion is the lack of labeled data. TaxoExpan addresses this problem by generating a set of ⟨query concept, anchor concept⟩ pairs from the existing taxonomy as self-supervision data. TaxoExpan first uses a position-enhanced graph neural network to encode each anchor concept's local structure in the existing taxonomy. Then, it learns to predict whether a query concept is the direct child of an anchor concept in the taxonomy using a noise-robust training objective. TaxoExpan can successfully expand a large field-of-study taxonomy with hundreds of thousands of concepts and outperforms the winning solution of SemEval 2016 taxonomy expansion task by 6.8% while running orders of magnitude faster. Together with my collaborators, we further improved TaxoExpan by: (1) deriving concept mini-paths from the existing taxonomy as self-supervision data while learning the model using multi-view co-training [21], and (2) identifying both the parent and children for each new emerging concept in the existing taxonomy [6, 7].

Contributions.

- We propose TaxoExpan, a novel self-supervised framework that automatically expands existing taxonomies without manually labeled data.
- We present an effective method for enhancing graph neural network by incorporating hierarchical positional information and a new training objective that enables the learned model to be robust to label noises in self-supervision data.
- We conduct extensive experiments to verify the effectiveness and efficiency of TaxoExpan framework on three real-world datasets from different domains.

1.2.4 Taxonomy Application

With a concept taxonomy constructed and enriched on a domain-specific document collection, we can explore a lot of downstream applications. For example, we have utilized taxonomies to facilitate semantic literature search [10, 11, 22] or to empower job post recommendation [4]. One important prerequisite of all those applications is that the text unit (either an entire document or an in-context text span) need to be tagged with a set of classes in the corresponding taxonomy. This can be formulated as a hierarchical multi-label text classification (HMTC) problem. Most existing HMTC methods are supervised and require massive human labeled training data that are not available in many real world scenarios.

To fully exploit the power of taxonomy, we propose TaxoClass [23], a weakly-supervised framework using only class surface names for hierarchical multi-label text classification. TaxoClass alleviates heavy human-labeling burdens and thus has a broader application scope. Specifically, TaxoClass leverages the explicit class relations in the given class taxonomy and pinpoints a few most essential classes for each document as its "core" classes. Based on those

core classes, TaxoClass first trains a taxonomy-enhanced classifier and then generalizes this classifier via multi-label self-training. Our experiments have shown TaxoClass can achieve around 0.71 Example-F1 using only class names, outperforming the state-of-the-art weakly-supervised methods by 25%.

Contributions.

- We propose the first weakly-supervised framework TaxoClass that only requires class surface names to perform hierarchical multi-label text classification.
- We develop an unsupervised method to identify document core classes based on which a text classifier can be learned.
- We conduct extensive experiments to verify the effectiveness of TaxoClass on two real-world datasets and show it can significantly reduce human annotation efforts.

1.2.5 Incorporating Event Knowledge in Taxonomy

In this thesis, we also explore how to incorporate event knowledge into the taxonomy so that we may organize text documents based on their internally described world events. A cornerstone step of achieving this goal is to identify a set of important event types and their associated event mentions, ideally without massive human annotations.

We propose ETypeClus, a corpus-based open-domain event type induction method that automatically discovers a set of event types from a given corpus. As events of the same type could be expressed in multiple ways, we propose to represent each event type as a cluster of ⟨predicate sense, object head⟩ pairs. Specifically, ETypeClus (1) selects salient predicates and object heads, (2) disambiguates predicate senses using only a verb sense dictionary, and (3) obtains event types by jointly embedding and clustering ⟨predicate sense, object head⟩ pairs in a latent spherical space. Our experiments, on three datasets from different domains, show our method can discover salient and high-quality event types, according to both automatic and human evaluations.

Contributions

- We present a new event type representation based on a cluster of ⟨predicate sense, object head⟩ tuples.
- We propose ETypeClus, a novel event type induction framework that automatically disambiguates predicate senses and learns a latent space with desired event cluster structures.
- We conduct extensive experiments on three datasets verify the effectiveness of ETypeClus in terms of both automatic and human evaluations.

1.3 AWARDS AND OVERALL IMPACT

This thesis focuses on developing principled methods to automatically construct, enrich, and apply taxonomy for converting unstructured text data into structured knowledge and insights. Our methods have led to over than 25 research papers published in top data mining and natural language processing conferences (*e.g.*, KDD, WWW, SIGIR, ACL, EMNLP, NAACL, AAAI, CIKM, ECMLPKDD) and have a broad impact on numerous downstream applications. Our concept set expansion and taxonomy construction techniques (SetExpan [1] and HiExpan [3]) have been transferred to U.S. Army Research Lab and the MITRE Corporation for further study and deployment. Our taxonomy enrichment method TaxoExpan [5] has been used to improve Microsoft Academia Graph [24]. Those methods are being taught in graduate-level courses (*e.g.*, CS 512: Data Mining Principles at the *University of Illinois at Urbana-Champaign*), and are introduced as major parts of tutorials in the top conferences of data mining (*e.g.*, KDD 2018, 2019, 2021, and ICDM 2021). The software tools developed in this thesis have received over 400 stars (i.e., likes) on GitHub as of Sept. 2021.

Organizations. The remainder of this thesis is organized as follows. The remainder of this thesis is organized as follows. We first discuss how to mine concept sets in Chapter 2 and how to construct a concept taxonomy from raw text corpora in Chapter 3. Then, we present our taxonomy enrichment technique in Chapter 4 and taxonomy application methods in Chapter 5. After that, we describe our open-domain event type induction method in Chapter 6. Finally, in Chapter 7, we conclude this thesis and describe our future work.

CHAPTER 2: CONCEPT SET EXPANSION

2.1 OVERVIEW AND MOTIVATIONS

Concept set expansion refers to the problem of expanding a small set of seed concepts into a complete set of concepts that belong to the same semantic class [25]. For example, if a given seed set is {*Oregon, Texas, Iowa*}, concept set expansion should return a hopefully complete set of concepts in the same semantic class, "*U.S. states*". Concept set expansion can benefit various downstream applications, such as knowledge extraction [26], taxonomy induction [19], and web search [27].

One line of work (*e.g.*, *Google Set* [28] and *SEAL* [25]) solves this task by submitting a query consisting of seed concepts to a search engine and mining top-ranked webpages. While this approach can achieve relatively good quality, the required seed-oriented online data extraction is costly. Therefore, more studies [29, 30, 31, 32, 33] are proposed in a *corpus-based* setting where sets are expanded by offline processing based on a specific corpus.

There are two general approaches for *corpus-based* set expansion—*one-time concept ranking* and *iterative pattern-based bootstrapping*. Based on the assumption that similar concepts appear in similar contexts, the first approach [29, 30, 31] makes a one-time ranking of candidate concepts based on their distributional similarity with seed concepts. A variety of "contexts" are used, including Web table, Wikipedia list, or just free-text patterns, and concept-concept distributional similarity is calculated based on *all* context features. However, blindly using *all* such features can lead to *concept intrusion* errors. Namely, some undesired concepts are wrongly introduced into the expanded set because many context features are not representative for defining the target semantic class although they do have connections with some of the seed concepts.

The second approach, iterative pattern-based bootstrapping [34, 26, 35], starts from seed concepts to extract quality patterns, based on a predefined pattern scoring mechanism, and it then applies extracted patterns to obtain even higher quality concepts using another concept scoring method. This process iterates and the high-quality patterns from all previous iterations are accumulated into a pattern pool which will be used for the next round of concept extraction. This approach works only when patterns/concepts extracted at each iteration are highly accurate, otherwise, it may cause severe *semantic shift* problem. Suppose in the previous example, *"located in _"* is taken as a good pattern from the seed set {*Oregon, Texas, Iowa*}, and this pattern brings in *USA* and *Ontario*. These undesired concepts may bring in even lower quality patterns and iteratively cause the set shifting farther away. Thus,

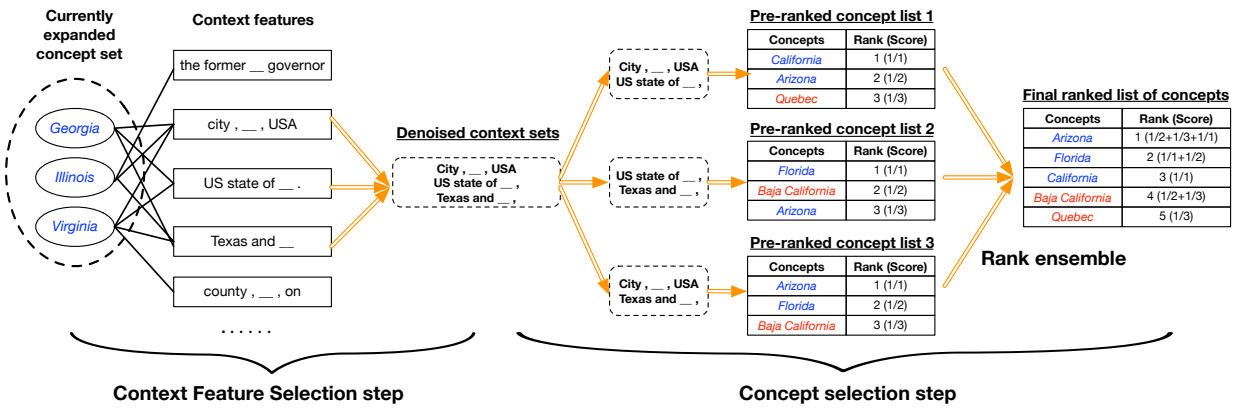


Figure 2.1: An example showing two steps in one iteration of SetExpan.

the pattern and concept scoring methods are crucial but sensitive in iterative bootstrapping methods. If they are not defined perfectly, the semantic shift can cause big problems. However, it is hard to have a perfect scoring mechanism due to the diversity and noisiness of unstructured text data.

To address these challenges, we propose a novel set expansion framework SetExpan in this Chapter. SetExpan carefully and conservatively extracts each candidate concept and iteratively improves the results. First, to overcome the concept intrusion problem, instead of using all context features, context features are carefully selected by calculating distributional similarity. Second, to overcome the semantic drift problem, different from other bootstrapped approaches, our high-quality feature pool will be reset at the beginning of each iteration. Finally, our carefully designed unsupervised ranking-based ensemble method is used at each iteration to further refine concepts and make our system robust to noisy or wrongly extracted pattern features. Figure 2.1 shows the pipeline at each iteration. SetExpan iteratively expands a concept set through a context feature selection step and a concept selection step. At the context feature selection, each context feature is scored based on its strength with currently expanded concepts and top-ranked context features are selected. At the concept selection step, multiple subsets of the selected representative context features are sampled and each subset is used to obtain a ranked concept list. Finally, all the ranked lists are collected to compute the final ranking list of each candidate concept for expansion.

The major contributions of this chapter are highlighted as follows:

- We propose an iterative set expansion framework with a novel context feature selection approach, to handle the issues of concept intrusion and semantic drift.
- We develop an unsupervised ranking-based ensemble algorithm for concept selection to make our system robust and further reduce the impact of semantic drift.

- We demonstrate the effectiveness and efficiency of our methods and show improvements over prior methods on multiple real-world datasets in different domains (news, Wikipedia articles, and scientific papers).

The rest of this chapter is organized as follows. We first discuss some related work in Section 2.2. Then, we propose our SetExpan framework in Section 2.3 and present experiment results in Section 2.4. After that, we briefly discuss how to extend SetExpan in Section 2.5. Finally, we conclude this chapter in Section 2.6.

2.2 RELATED WORK

The problem of completing a concept set given several seed concepts has attracted extensive research efforts due to its practical importance. Google Sets [28] was among the earliest work dealing with this problem. It used proprietary algorithms and is no longer publicly accessible. Later, Wang and Cohen proposed *SEAL* system [25], which first submits a query consisting of all seed concepts into a general search engine and then mines the top-ranked webpages. Recently, Chen *et al.* [27] improved this approach by leveraging a "page-specific" extractor built in a supervised manner and showed good performance on long-tail (*i.e.*, rare) term expansion. All these methods need an external search engine and require seed-oriented data extraction. In comparison, our approach conducts corpus-based set expansion without resorting to online data extraction from specific webpages.

To tackle the corpus-based set expansion problem, Ghahramani and Heller [36] used a Bayesian method to model the probability that a candidate concept belongs to some unknown cluster that contains the input seeds. Pantel *et al.* [29] developed a web-scale set expansion pipeline by exploiting distributional similarity on context words for each candidate concept. He *et al.* [31] proposed the SEISA system that uses query logs along with web lists as external evidence besides free text, and designed an iterative similarity aggregation function for set expansion. Recently, Wang *et al.* [32] leveraged web tables and showed very competitive results when not only seed concepts but also intended class name were given. While these semi-structured lists and tables are helpful, they are not always available for some specific domain corpus such as PubMed articles or DBLP papers. Perhaps the most relevant work to ours is by Rong [33]. In that paper, the authors used the skip-pattern feature combined with additional user-generated ontologies (*i.e.*, Wikipedia list) for set expansion. However, they targeted the multifaceted expansion and exploited all skip-pattern features for calculating the similarity because two concepts. In our work, we keep the core idea of distributional similarity but calculate such similarity using only carefully selected *denoised* context features.

In a broader sense, our work is also related to information extraction and named con-

cept recognition. Without given enough training data, bootstrapped concept extraction system [37, 26] is the most popular and effective choice. At each bootstrap iteration, the system will first create patterns around concepts; score patterns based on their ability to extract more positive concepts and less negative concepts (if provided), and use top-ranked patterns to extract more candidate concepts. Multiple pattern scoring and concept scoring functions are proposed. For example, Riloff *et al.* [38] scored each pattern by calculating the ratio of positive concepts among all concepts extracted by it, and scored each candidate concept by the number and quality of its matched patterns. Gupta *et al.* [37] scored patterns using the ratio of scaled frequencies of positive concepts among all concepts extracted by it. All these methods are heuristic and sensitive to different model parameters.

More generally, our work is also related to class label acquisition [39, 40] which aims to propagate class labels to data instances based on labeled training examples, and concept clustering [41, 42] where the goal is to find clusters of concepts. However, the class label acquisition methods require a much larger number of training examples than the typical size of user input seed set, and the concept clustering algorithms can only find semantically related concepts instead of concepts strictly in the same semantic class.

2.3 THE SETEXPAN FRAMEWORK

We first introduce our context features and data model used by SetExpan in Section 2.3.1 and then present our context-dependent similarity measure in Section 2.3.2. After that, we discuss how to select context features in Section 2.3.3 and present our novel unsupervised ranking-based ensemble method for concept selection in Section 2.3.4.

2.3.1 Data Model and Context Features

We explore two types of context features obtained from the plain text: (1) skip-patterns [33] and (2) coarse-grained types [26]. As shown in Figure 2.2(a), data is modeled as a bipartite graph, with candidate concepts on one side and their context features on the other. Each type of context features are described as follows.

Skip-pattern: Given a target concept e_i in a sentence, one of its skip-pattern is "w_{-1} _ w_1" where w_{-1} and w_1 are two context words and e_i is replaced with a placeholder. For example, one skip-pattern of concept *"Illinois"* in sentence *"We need to pay Illinois sales tax."* is "pay _ sales". As suggested in [33], we extract up to six skip-patterns of different lengths for one target concept e_i in each sentence. One advantage of using skip-patterns is that it imposes strong positional constraints.

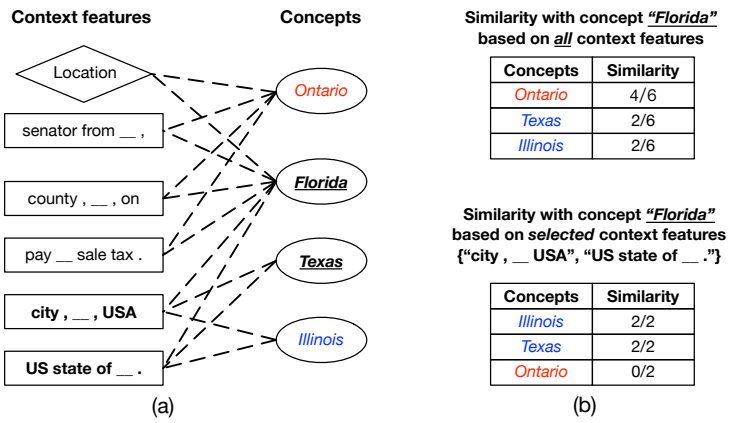


Figure 2.2: (a) A simplified bipartite graph data model. (b) Similarity with seed concept conditioned on two different sets of context features.

Coarse-grained type: Besides the unstructured skip-pattern features, we use coarse-grained types to filter those obviously-wrong concepts. For examples, when we expand the "U.S. states" class, we will not consider any concept that is typed as a person. After this process, we can obtain a cleaner subset of candidate concepts. Such a mechanism is also adopted in [26].

After obtaining the "nodes" in bipartite graph data model, we need to model the edges in the graph. In this work, we assign the weight between each pair of concept e and context feature c using the *TF-IDF transformation* [33], which is calculated as follows:

$$f_{e,c} = \log(1 + X_{e,c}) \left[\log |E| - \log \left(\sum_{e'} X_{e',c} \right) \right], \tag{2.1}$$

where $X_{e,c}$ is the raw co-occurrence count between concept e and context feature c, $|E|$ is the total number of candidate concepts. We refer to such scaling as the *TF-IDF transformation* since it resembles the *tf-idf* scoring in information retrieval if we treat each concept e as a "document" and each of its context feature c as a "term". Empirically, we find such weight scaling outperforms some other alternatives such as point-wise mutual information (PMI) [31], truncated PMI [43], and BM25 scoring [44].

2.3.2 Context-dependent Similarity

With the bipartite graph data model constructed, the task of expanding a concept set at each iteration can be viewed as finding a set of concepts that are most "similar" to the currently expanded set. In this study, we use the weighted Jaccard similarity measure. Specifically, given a set of context features F, we calculate the *context-dependent* similarity

as follows:

$$Sim(e_1, e_2|F) = \frac{\sum_{c \in F} \min(f_{e_1,c}, f_{e_2,c})}{\sum_{c \in F} \max(f_{e_1,c}, f_{e_2,c})}. \tag{2.2}$$

Notice that if we change context feature set F, the similarity between concept pair is likely to change, as demonstrated in Figure 2.2(b). Finally, we want to emphasize that our proposed method is general in the sense that other common similarity metrics (*e.g.*, cosine similarity) can also be used. In practice, we find the performance of a set expansion method depends less on the exact choice of base similarity metrics, but more on which contexts are selected for calculating *context-dependent* similarity. Similar results were also reported in [31].

2.3.3 Context Feature Selection

As shown in Figure 2.2(b), the similarity between two concepts really depends on the selected feature set F. The motivation of context feature selection is to find a feature subset F^* of fixed size Q that best "profiles" the target semantic class. In other words, we want to select a feature set F^* based on which concepts within target class are most "similar" to each other. Given such F^*, the concept-concept similarity conditioned on it can best reflect their distributional similarity with regard to the target class. In some sense, such F^* best profiles the target semantic class. Unfortunately, to find such F^* of fixed size Q, we need to solve the following optimization problem which turns out to be NP-Hard, as shown in [45].

$$F^* = \arg\max_{|F|=Q} \sum_{i=1}^{|X|} \sum_{j>i}^{|X|} Sim(e_i, e_j|F), \tag{2.3}$$

where X is the set of currently expanded concepts. Initially, we treat the user input seed set S as X. As iterations proceed, more concepts will be added into X.

Given the NP-hardness of finding the optimal context feature set, we resort to a heuristic method that first scores each context feature based on its accumulated strength with concepts in X and then selects top Q features with maximum scores. Take Figure 2.2(a) as an example, we assume all edge weights in the bipartite graph are equal to 1 and let the currently expanded concept set X be {"Florida", "Texas"}. Suppose we want to select two "denoised" context features, we will first score each context feature based on its associated concepts in X. The top 4 contexts will obtain a score 1 since they match only one concept in X with strength 1, and the 2 contexts below will get a score 2 because they match both concepts in X. Then, we rank context features based on their scores and select 2 contexts with highest scores: "city , _, USA", "US state of _ ." into F.

Finally, we want to emphasize two major differences of our context feature selection method from other heuristic "pattern selection" methods. First, most pattern selection

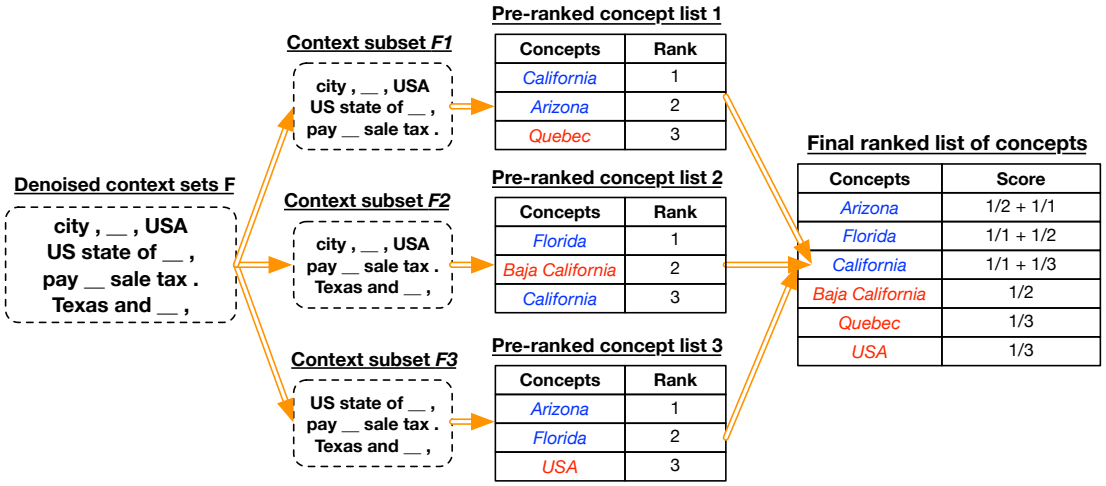


Figure 2.3: A toy example to show concept selection via rank ensemble.

methods require either users to explicitly provide the "negative" examples for the target semantic class [46, 26, 34], or implicitly expand multiple mutually exclusive classes in which instances in one class serve as negative examples for all the other classes [47, 43]. Our method requires only a small number of "positive" examples. In most cases, it is hard for humans to find good discriminative negative examples for one class, or to provide both mutually exclusive and somehow related comparative classes. Second, the bootstrapping method will add its selected "quality patterns" during each iteration into a quality pattern pool, while our method will select high quality context features at each iteration from scratch. If one noisy pattern is selected and added into the pool, it will continue to introduce more irrelevant concepts at all the following iterations. Our method can avoid such noise accumulation.

2.3.4 Concept Selection via Rank Ensemble

Intuitively, the concept selection problem can be viewed as finding those concepts that are most similar to the currently expanded set X conditioned on the selected context feature set F. To achieve this, we can rank each candidate concept based on its score in Eq. (2.4) and then add top-ranked ones into the expanded set:

$$score(e|X, F) = \frac{1}{|X|} \sum_{e' \in X} Sim(e, e'|F). \tag{2.4}$$

However, due to the ambiguity of natural language in free-text corpora, the selected context feature set F may still be noisy in the sense that an irrelevant concept is ranked higher than a relevant one. To further reduce such errors, we propose a novel ranking-based ensemble method for concept selection.

The key insight of our method is that an inferior concept will not appear frequently in multiple pre-ranked concept lists at top positions. Given a selected context set F, we first use sampling without replacement method to generate T subsets of context features $F_t, t = 1, 2, \ldots, T$. Each subset is of size $\alpha|F|$ where α is a model parameter within range $[0, 1]$. For each F_t, we can obtain a pre-ranked list of candidate concepts L_t based on $score(e|X, F_t)$ defined in Eq. (2.4). We use r_t^i to denote the rank of concept e_i in list L_t. If concept e_i does not appear in L_t, we let $r_t^i = \infty$. Finally, we collect T pre-ranked lists and score each concept based on its mean reciprocal rank (mrr). All concepts with average rank above r, namely $mrr(e) \leq T/r$, will be added into concept set X.

$$mrr(e_i) = \sum_{t=1}^{T} \frac{1}{r_t^i}, \qquad r_t^i = \sum_{e_j \in E} I\left(score(e_i|X, F_t) \leq score(e_j|X, F_t)\right), \tag{2.5}$$

where $I(\cdot)$ is the indicator function. Naturally, a relevant concept will rank at top position in multiple pre-ranked lists and thus accumulate a high mrr score, while an irrelevant concept will not consistently appear in multiple lists at high position which leads to low mrr score.

We use the following example to demonstrate the whole process of concept selection. In Figure 2.3, we want to expand the "US states" semantic class given a selected context feature set F with 4 features. We first sample a subset of 3 context features $F_1 = \{$"city , _ , USA", "US state of _ ,", "pay _ sales tax ."$\}$, and then use F_1 to obtain a pre-ranked concept list $L_1 = \langle$"California", "Arizona", "Quebec"$\rangle$. By repeating this process three times, we get three pre-ranked lists and ensemble them into a final ranked list in which concept "Arizona" is scored 1.5 because it is ranked in the 2nd position in L_1 and 1st position in L_3. Finally, we add those concepts with mrr score larger than 1, meaning this concept is ranked at 3rd position on average, into the expanded set X. In this simple example, the model parameters $T = 3, \alpha = \frac{|F_1|}{|F|} = 0.75$, and $r = 3$.

Summary. Algorithm 2.1 summarizes the whole SetExpan framework. The candidate concept set E and bipartite graph data model G are pre-calculated and stored. A user needs only to specify the seed set S as the task guidance and the expected size of output set K. There is a total of 4 model parameters: the number of top quality context features selected in each iteration Q, the number of pre-ranked concept lists T, the relative size of feature subset $0 < \alpha < 1$, and final mrr threshold r. The tuning and sensitivity of these parameters will be discussed in the experiment section.

2.4 EXPERIMENTS

In this section, we will evaluate SetExpan on three massive text corpora across different domains. We first compare the propose method with many other methods to demonstrate

Algorithm 2.1: SetExpan

Input: Candidate concept set E, initial seed set S, concept-context graph G, expected size of output set K, model parameters $\{Q, T, \alpha, r\}$.

Output: The expanded set X.

```
X = S;
while |X| ≤ K do
    Set F = ∅ // Select denoised contexts from scratch;
    Score context features based on X and add top Q denoised contexts into F;
    // concept-selection via rank ensemble;
    for t = 1, 2, ..., T do
        Uniformly sample αQ contexts and construct feature subset F_t;
        Score concepts based on Eq. (2.4) given F_t and obtain the pre-ranked list L_t;
        Update the mrr score of each concept based on Eq. (2.5);
    X = X ∪ {e|mrr(e) ≥ T/r} // Add concepts into expanded set X;
Return X;
```

its high performance. Then, we explore the robustness of our method by varying different hyper-parameters. In this end, we present some interesting case studies.

2.4.1 Datasets

We use three corpora to evaluate the performance of SetExpan. Table 2.1 lists 3 datasets we used in experiments: (1) **APR** is constructed by crawling all 2015 news articles from AP and Reuters; (2) **Wiki** is a subset of English Wikipedia used in [48], and (3) **PubMed-CVD** is a collection of research paper abstracts about cardiovascular disease retrieved from PubMed. For APR and PubMed-CVD datasets, we adopt a data-driven phrase mining tool [49] to obtain concept mentions and type them using ClusType [50]. Each concept mention is mapped heuristically to a concept based on its lemmatized surface name. We then extract variable-length skip-patterns for all concept mentions as features for their corresponding concepts, and construct the bipartite graph data model as introduced in the previous section. For Wiki dataset, the concepts have already been extracted and typed using distant supervision. For the type information in each dataset, there are 16 coarse-grained types in APR and 4 coarse-grained types in PubMed-CVD. For Wiki, since it originally has about 50 fine-grained types, which may reveal too much information, we manually mapped them to 11 more coarse-grained types.

A query is a set of seed concepts of the same semantic class in a dataset, serving as the input for each system to expand the set. The process of query generation is as follows. For each dataset, we first extract 2000 most frequent concepts in it and construct a concept list.

Table 2.1: Datasets statistics and query descriptions.

Dataset	FileSize	#Sentences	#Concepts	#Test queries
APR	775MB	1.01M	122K	40
Wiki	1.02GB	1.50M	710K	20
PubMed-CVD	9.3GB	23M	179K	5

Then, we ask three volunteers to manually scan the concept lists and propose a few semantic classes for each list. The proposed class should be interesting, relatively unambiguous and has a reasonable coverage in its corresponding corpus. These semantic classes cover a wide variety of topics, including locations, companies as well as political parties, and have different degrees of difficulty for set expansion. After finalizing the semantic classes for each dataset, the volunteers randomly select concepts of each semantic class from the frequent concept list to form 5 queries of size 3. To select the queries for PubMed-CVD, we seek help from two additional volunteers with biomedical expertise, following the same previous approach. Due to the large size of PubMed-CVD dataset and runtime limitation, we only select 1 semantic class (hormones) with 5 queries.

With all queries selected, we have humans to label all the classes and instances returned by each of the following compared methods. For APR and Wiki datasets, the inter-rater agreements (kappa-value) over three students are 0.7608 and 0.7746, respectively. For PubMed-CVD dataset, the kappa-value is 0.9236. All concepts with conflicting label results are further resolved after discussions among all human labelers.

2.4.2 Compared Methods

Since the focus on this work is the corpus-based set expansion, we do not compare with other methods that require online data extractions. Also, to further analyze the effectiveness of each module in SetExpan framework. We implement 3 variations of our framework.

- word2vec [51]: We use the SkipGram model in word2vec to learn concept embeddings and return k nearest neighbors around seed concepts as the expanded set.
- PTE [52]: We first construct a heterogeneous information network including concepts, skip-pattern features, and type features. Then, we apply PTE model to learn concept embeddings and determine the k nearest neighbors around seed concepts.
- SEISA [31]: A concept set expansion algorithm based on iterative similarity aggregation. It uses the occurrence of concepts in web list and query log as concept features. In our experiments, we replace the web list and query log with our skip-pattern and coarse-grained context features.

- EgoSet [33]: A multifaceted set expansion system based on skip-patterns, word2vec embeddings and WikiList. The original system expands a seed set to multiple concept sets, considering the ambiguities in seed set. To achieve this, they use a community detection method to separate extracted concepts into several groups. However, in order to better compare with EgoSet, we carefully select queries that have little ambiguity or at least the seed set in the query is dominating in one semantic class. Thus, we discard the community detection part in EgoSet and treat all extracted concepts as in one semantic class.
- SetExpan-cs: Disable the context feature selection module in SetExpan, and use all context features to calculate distributional similarity.
- SetExpan-re: Disable the rank ensemble module in SetExpan. Instead, we use all selected context feature to rank candidate concepts at one time and add top-ranked ones into the expanded set.
- SetExpan-full: The full version of our proposed method, with both context feature selection and rank ensemble components enabled.

For fair comparison, we try different combinations of parameters and report the best performance for each baseline method.

2.4.3 Evaluation Metrics

For each test case, the input is a query, which is a set of 3 seed concepts of the same semantic class. The output will be a ranked list of concepts. For each query, we use the conventional average precision $\text{AP}_k(c, r)$ at k ($k = 10, 20, 50$) for evaluation, given a ranked list of concepts c and an unordered ground-truth set r. For all queries under a semantic class, we calculate the mean average precision (MAP) at k as $\frac{1}{N}\sum_i AP_k(c_i, r)$, where N is the number of queries. To evaluate the performance of each approach on a specific dataset, we calculate the mean-MAP (MMAP) at k over all queried semantic classes as $\text{MMAP}_k = \frac{1}{T}\sum_{t=1}^{T}[(\frac{1}{N_t})\sum_i AP_k(c_{ti}, r_t)]$, where T is the number of semantic classes, N_t is the number of queries of t-th semantic class, c_{ti} is the extracted concept list for i-th query for t-th semantic class, and r_t is the ground truth set for t-th semantic class.

2.4.4 Overall Performance

Table 2.2 shows the MMAP scores of all methods on 3 datasets[1]. We can see that SetExpan outperforms all four baselines in terms of the MMAP score. We further look at their performances on each concept class, as shown in Figure 2.4. We can see that the performance

[1]Results of SEISA on PubMed-CVD are omitted due to the scalability issue.

Table 2.2: Set expansion performance on 3 datasets over all queries.

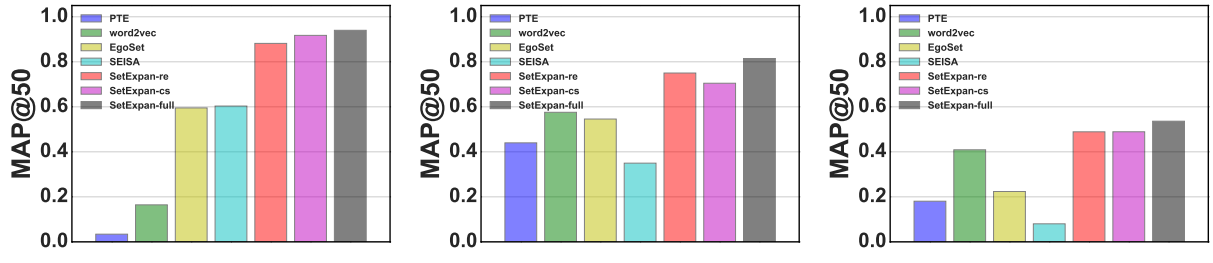
Methods	APR			Wiki			PubMed-CVD		
	MAP@10	MAP@20	MAP@50	MAP@10	MAP@20	MAP@50	MAP@10	MAP@20	MAP@50
EgoSet	0.3949	0.3942	0.3706	0.5899	0.5754	0.5622	0.0511	0.0410	0.0441
SEISA	0.7423	0.6090	0.3892	0.7643	0.6606	0.4998	—	—	—
word2vec	0.6054	0.5385	0.4180	0.7193	0.6289	0.4510	0.8427	0.7701	0.6895
PTE	0.3144	0.2777	0.1996	0.6817	0.5596	0.3839	0.9071	0.7654	0.5641
SetExpan-cs	0.8240	0.7997	0.7674	0.9540	0.8955	0.7439	**1.000**	**1.000**	0.5991
SetExpan-re	0.8509	0.7792	0.7681	0.9392	0.8680	0.7291	**1.000**	0.9605	0.7371
SetExpan-full	**0.8967**	**0.8621**	**0.7885**	**0.9571**	**0.9010**	**0.7457**	**1.000**	**1.000**	**0.7454**

of these baseline methods varies a lot on different semantic classes, while our SetExpan can consistently beat them. One reason is that none of these methods applies context feature selection or rank ensemble, and a single set of unpruned features can lead to various levels of noise in the results. Another reason is the lack of an iterative mechanism in some of those approaches. For example, even if EgoSet includes the results from word2vec to help it boost the performance, it still achieves low MAP scores in some semantic classes. Finding the nearest neighbors in only one iteration can be a key reason. And although SEISA is applying the iterative technique, instead of adding a small number of new concepts in each iteration, it expands a full set in each iteration based on the coherence score of each candidate concept with the previously expanded set. It pre-calculates the size of the expanded set with the assumption that the feature similarities follow a certain distribution, which does not always hold to all datasets or semantic classes. Thus, if the size is far different from the actual size or is too big to extract a confident set at once, each iteration will introduce a lot of noise and cause semantic drift.

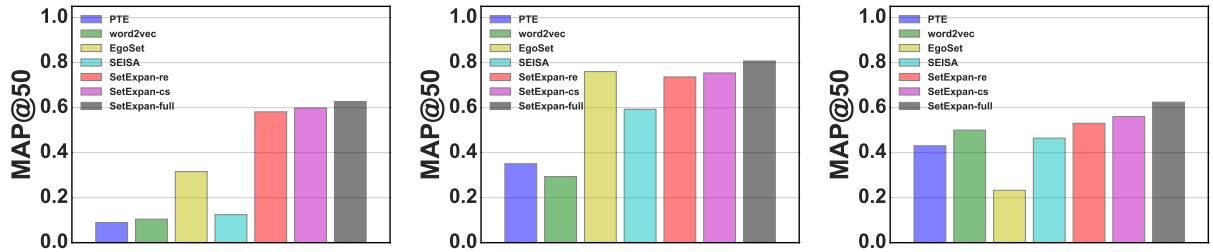
2.4.5 Ablation Studies

Comparison with SetExpan-re and SetExpan-cs. At the dataset level, the MMAP scores of SetExpan-full outperforms its two variation approaches. In the semantic class level, we can see that SetExpan-re and SetExpan-cs sometimes have their MAP much lower than SetExpan-full while sometimes they almost achieve the same performance with SetExpan-full. This means they fail to stably extract concepts with good quality. The main reason is still that a single set of features or ensembles over unpruned features can lead to various levels of noise in the results. Only under the circumstances that the single set of features or the unpruned features happen to be nicely selected without too much noise, which tends to happen when the query is relatively "easy", these variation approaches can achieve good results.

Effects of Context Feature Selection. We already see that adding the context feature

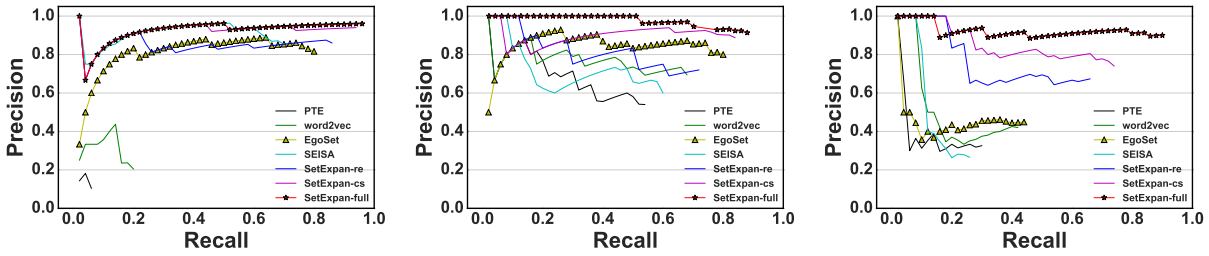


(a) APR Country (b) APR Law (c) APR Party

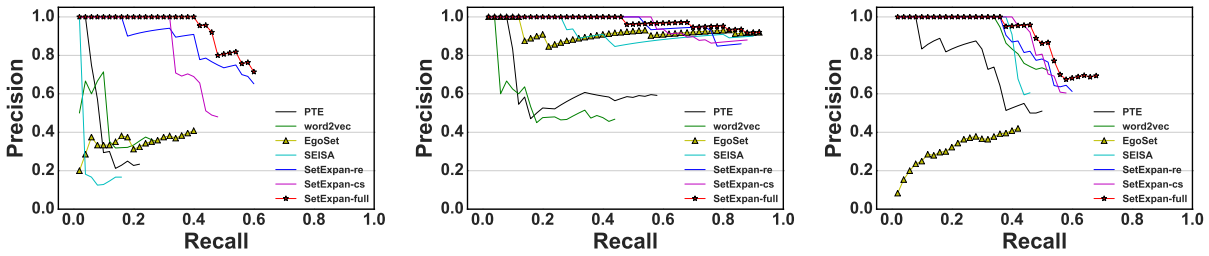


(d) Wiki Sport League (e) Wiki TV Channel (f) Wiki China Province

Figure 2.4: Set expansion performance for each semantic class.



(a) APR Country (b) APR Law (c) APR Party



(d) Wiki Sport League (e) Wiki TV Channel (f) Wiki China Province

Figure 2.5: Set expansion performance for individual queries in different classes.

selection component helps improve the performance. What's also noticeable is that the addition of context selection process becomes more obvious as the size of the corpus increases. The difference between MMAP scores of SetExpan-cs and SetExpan-full is much larger in

PubMed-CVD compared with APR and Wiki datasets. This is because that as the corpus size increases, we will have more noisy features and more candidate concepts while the good features to define the target concept set may be limited. Thus, without context selection, noise can damage the performance much more. The evidence can also be found from the performance of EgoSet across the three datasets. It can achieve reasonably good results in APR and Wiki, however, it performs much worse in PubMed-CVD.

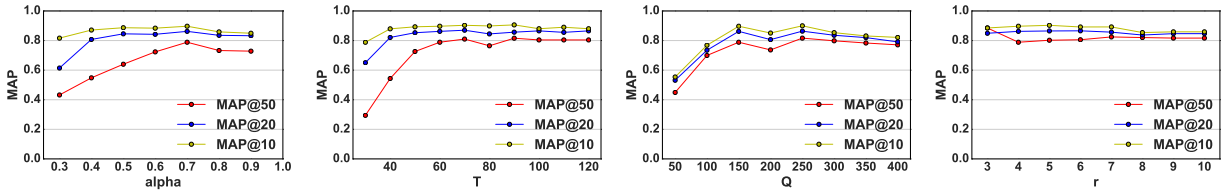
Effect of Rank Ensemble. From the above experiments, we can see that the effect of rank ensemble may vary across different semantic classes. However, the contribution of rank ensemble seems to be more stable across datasets, compared with the effect of context selection. This is because we apply the default set of hyper-parameter values in each test case above. In the below hyper-parameter analysis, we will show that the number of ensemble batches and the percentage of features to be randomly sampled can affect the contribution of rank ensemble to the concept set expansion performance.

Hyper-parameter Analysis. There are totally 4 hyper-parameters in SetExpan: Q (the number of selected context features), α (the percentage of features to be sampled), T (the number of ensemble batches), and r (the threshold of a candidate concept's average rank). We study the influence of each hyper-parameter by fixing all other hyper-parameters to default values, and present one graph showing the MMAP scores of SetExpan on APR dataset versus the changes of that hyper-parameter.

- α: From the graph, the performance increases sharply as α increases until it reaches about 0.6. Then, it starts to stay stable and decreases after 0.7.
- T: The performance first increases as we increase the ensemble batches and then becomes stable after 60 batches.
- Q: In the range of 50 - 150, the performance increases sharply as Q increases, which means the majority of top 150 context features can provide rich information to identify concepts belonging to the target semantic class. The available information gets more and more saturated after Q reaches 150 and start to introduce noises and hamper the performance after around 300.
- r: Our experiments show that the performance is not very sensitive to the threshold of a candidate concept's average rank.

2.4.6 Case Studies.

Figure 2.7 presents three case studies for SetExpan. We show one query for each dataset. In each case, we show top 3 ranked concepts and top/bottom 3 skip-pattern features after



(a) α (b) T (c) Q (d) r

Figure 2.6: Hyper-parameter sensitivity of SetExpan on two datasets.

Dataset	Query	Top ranked concepts in the first 3 iterations	Top/Bottom skip-pattern features selected in the first 3 iterations	Coarse-grained type
APR	**{Patriot Act, Obamacare, Clery Act}**	**Iteration 1**: USA Patriot Act, USA Freedom Act, Voting Rights Act, ... **Iteration 2**: Stock Act, Religious Freedom Restoration Act, Foreign Intelligence Surveillance Act, ... **Iteration 3**: Americans with Disabilities Act, Healthy Families Act, Goonda Act, ...	**Iteration 1:** Top 3: "the __ provisions", "provisions of the __", "defund __ ." Bottom 3: "2010 __ ,", "also known as __ .", "under the __ , and" **Iteration 2:** Top 3: "under the __ to", "provisions of the __ ,", "the __ into law." Bottom 3: "the __ - which has", "the _The House", "the _ , first" **Iteration 3:** Top 3: "under the __ to", "Under the __ ,", "the __ into law" Bottom 3: "of the __ passed", "the __ , the most", "replacing __ ."	**Event**
Wiki	**{ESPN, ESPN2, Spike TV}**	**Iteration 1:** ABC, CBS, NBC, ... **Iteration 2:** BBC, ITV, Channel 4, ... **Iteration 3:** TBS, ITV1, BBC Two, ...	**Iteration 1:** Top 3: "telecast on __ .", "televised on __ .", "televised by __ ." Bottom 3: "on __ , to the", ", and perhaps __ .", "from an __ website" **Iteration 2:** Top 3: "the __ sitcom", "the __ television network", "ABC , __ ," Bottom 3: "on __ on September", "broadcast on __ on", "the __ soap opera The" **Iteration 3:** Top 3: "the __ sitcom", "the __ soap", "the __ soap opera", ... Bottom 3: "aired on __ between", "of the __ show", "on the __ crime"	**Organiz-ation**
PubMed -CVD	**{FSH, TSH, MSH}**	**Iteration 1:** LH, GH, ACTH, ... **Iteration 2:** LHRH, AMH, GHRH, ... **Iteration 3:** Renin, GnRH-I, AVP, ...	**Iteration 1:** Top 3: "stimulating hormone (__)", "hormone (__) ,", "hormone (__) and", ... Bottom 3: "g/L , __ =", ", __ and prolactin", "hormone (__) -" **Iteration 2:** Top 3: "hormone (__) ,", "hormone (__) and", "hormone (__) .", ... Bottom 3: ", __ , estradiol ,", ", __ , and PRL", "hormone (__) -" **Iteration 3:** Top 3: "hormone (__) ,", "hormone (__) and", "hormone (__) .", ... Bottom 3: "(__) and insulin-like", ", TSH , __ ,", "levels of __ , FSH"	**Proteins and Genes (PRGE)**

Figure 2.7: Three case studies of SetExpan on each dataset.

context feature selection for the first 3 iterations as well as the coarse-grained type. In all cases, our algorithm successfully extracts correct concepts in each iteration, and the top-ranked skip-patterns are representative in defining the target semantic class. On the other hand, we notice that most of the bottom 3 skip-patterns selected are very general or not representative at all. These context features could potentially introduce noisy concepts and thus the rank ensemble can play a rival role in improving the results.

2.5 EXTENSIONS OF SETEXPAN

SetExpan demonstrates an effective iterative framework for concept set expansion. Together with collaborators in our group, we further extend SetExpan by exploiting automatically discovered negative sets [17] and incorporating pre-trained language models [2].

2.5.1 SetCoExpan: Guiding SetExpan via Auxiliary Sets Generation and Co-Expansion

We observe that a typical source of SetExpan error comes from the concepts from different semantic classes that share some common relations to the target class. For example, when expanding the *Country* class, we may wrongly introduce those erroneous concepts in the *City* class. If we can capture such subtle relationships between concepts belonging to different semantic classes, we can use them to separate concepts from different classes and conduct set co-expansion. Such co-expansion may incorporate signals from all the related, participating classes, keep warning the target class not to cross over the boundaries of its possible rivals, and guide the expanding direction of each set by avoiding to bump into each other's territory. The co-expansion of such multiple rival sets benefits each other from mutually exclusive signals, and the quality of multiple sets expansion can be improved simultaneously. Some previous studies [53, 54, 46] also found that using mutual exclusive signals from other related "auxiliary" classes could help. However, they often require users to explicitly provide those auxiliary classes, which was not applicable in many real-world scenarios.

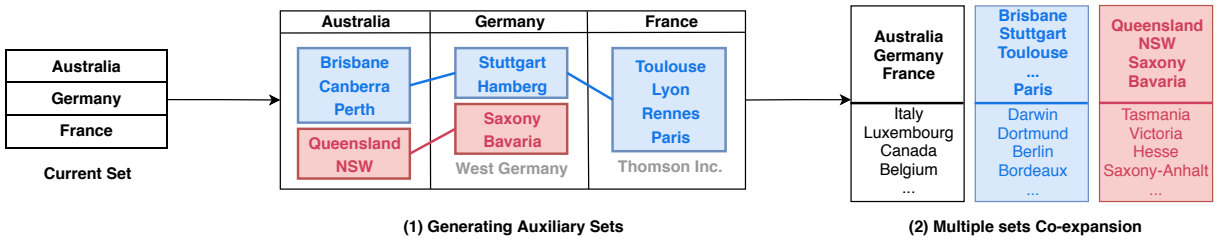


Figure 2.8: Overview of SetCoExpan. For user-input country names, we first retrieve related terms such as provinces and cities and then cluster them into auxiliary sets. Multiple sets are then co-expanded by extracting discriminative context features.

We propose SetCoExpan, a fully automated approach to improve SetExpan without human-provided auxiliary sets. As shown in Figure 2.8, SetCoExpan consists of two modules that are operated iteratively: (1) an auxiliary sets generation module that finds auxiliary sets holding certain relations with the target set in an unsupervised way, and (2) a multiple sets co-expansion module that takes multiple sets as input and extracts the most discriminative

features to tell the target class from auxiliary sets. The auxiliary sets generation module first retrieves semantically related terms to each seed element in an embedding space that captures topical similarity. These related terms are then grouped by their semantic types, captured by intra-seed clustering and inter-seed merging in an unsupervised way. The multiple sets co-expansion modules takes the target seed set as well as auxiliary sets as input. By incorporating knowledge from both seed set and auxiliary sets, we can control the expanding directions of multiple sets. Specifically, context features are scored by how well they can tell different sets apart, and the algorithm drives the expanding direction away from ambiguous areas. We demonstrate the effectiveness of SetCoExpan in below Section 2.5.3.

2.5.2 CGExpan: Empowering SetExpan via Language Model Probing

Besides using auxiliary sets to guide the set expansion process, we also explore how to leverage the target class name to enhance SetExpan. Intuitively, knowing the class name is "country", instead of "state" or "city", can help us identify *unambiguous patterns* and eliminate erroneous concepts like "*Europe*" and "*New York*". Moreover, we can acquire such knowledge (*i.e.*, positive and negative class names) by probing a language model automatically without relying on human annotated data.

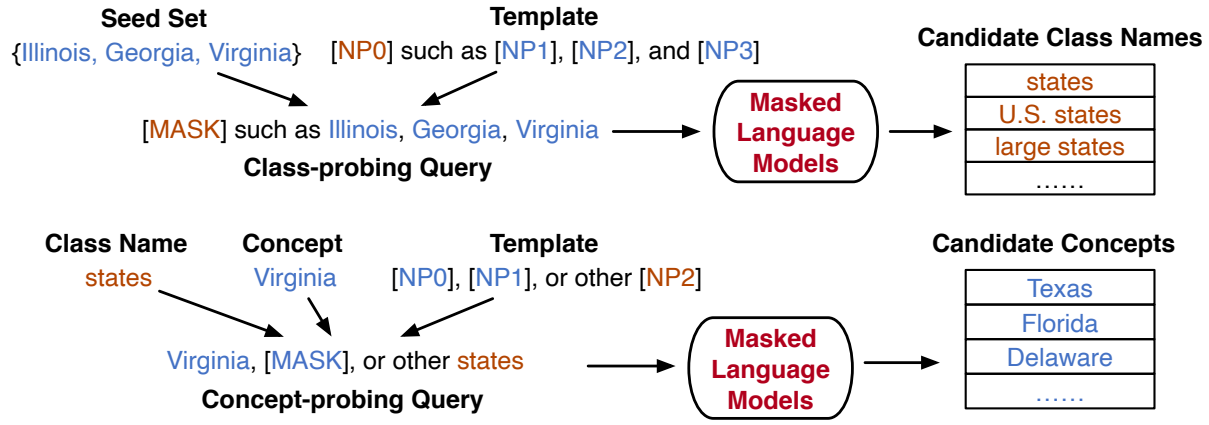


Figure 2.9: Examples of class-probing and concept-probing queries in CGExpan.

Motivated by the above intuition, we propose CGExpan, an iterative framework that empowers concept set expansion with class names automatically generated from pre-trained language models (PLMs) [55, 56]. CGExpan consists of three modules: (1) The first, *class name generation* module, constructs and submits class-probing queries (*e.g.*, "[MASK] such as Illinois, Georgia, and Virginia." as shown in Figure 2.9) to a language model for retrieving a set of candidate class names. (2) The second, *class name ranking* module, builds a

Table 2.3: Mean average precision on Wiki and APR. "$^{\nabla}$" means the number is directly from the original paper.

Methods	Wiki			APR		
	MAP@10	MAP@20	MAP@50	MAP@10	MAP@20	MAP@50
EgoSet [33]	0.904	0.877	0.745	0.758	0.710	0.570
SetExpan [1]	0.944	0.921	0.720	0.789	0.763	0.639
SetExpander [57]	0.499	0.439	0.321	0.287	0.208	0.120
CaSE [58]	0.897	0.806	0.588	0.619	0.494	0.330
MCTS [59]	0.980^{∇}	0.930^{∇}	0.790^{∇}	0.960^{∇}	0.900^{∇}	0.810^{∇}
SetCoExpan	0.976	0.964	**0.905**	0.933	0.915	0.830
CGExpan	**0.995**	**0.978**	0.902	**0.992**	**0.990**	**0.955**

concept-probing query for each candidate class name and retrieves a set of concepts. The similarity between this retrieved set and the current concept set serves as a proxy for the class name quality, based on which we rank all candidate class names. (3) The third, *class-guided concept selection* module, scores each concept conditioned on the above selected class names and add top-ranked concepts into the currently expanded set. As better class names may emerge in later iterations, we score and rank all concepts (including those already in the expanded set) at each iteration, which helps alleviate the semantic drift issue.

2.5.3 Experiments

We evaluate the performance of SetCoExpan and CGExpan on Wiki and APR datasets used in previous SetExpan experiments. Besides the previous EgoSet baseline, we further compare SetCoExpan and CGExpan with the following methods:

- SetExpander [57]: This method trains different embeddings based on different types of context features and leverages additional human-annotated sets to build a classifier on top of learned embeddings to predict whether a concept belongs to the set.
- CaSE [58]: This method combines skip-pattern features and embedding features to score and rank concepts once from the corpus. The original paper has three variants and we use the CaSE-W2V variant since it is the best model claimed in the paper.
- MCTS [59]: This method bootstraps the initial seed set by combing the Monte Carlo Tree Search algorithm with a deep similarity network to estimate delayed feedback for pattern evaluation and to score concepts given selected patterns.

Table 2.3 shows the overall performance of different methods. We can see that SetCoExpan and CGExpan in general outperform all the baselines by a large margin. On the Wiki dataset, both SetCoExpan and CGExpan achieve over 24% improvements over SetExpan in MAP@50,.

On the APR dataset, CGExpan obtains over 49% improvements over SetExpan in terms of MAP@50. These results verify the effectiveness of parallel expansion of auxiliary sets (in SetCoExpan) and PLM-guided expansion model (in CGExpan).

2.6 SUMMARY

In this chapter, we present an iterative concept set expansion framework SetExpan with a ranking-based unsupervised ensemble technique for robust concept selection. Our extensive experiments show SetExpan is domain-independent, outperforms many other set expansion methods, and derives high-quality concept sets with minimal human efforts. Furthermore, we introduce a few ways to extend SetExpan by incorporating pre-trained language models and exploiting automatically discovered auxiliary sets.

For future work, it is interesting to (1) extend SetExpan to multiple languages, (2) develop interactive methods to allow users to directly control the expansion process and/or to provide valuable feedbacks after the iterative expansion process ends, and (3) enable the downstream applications of set expansion to provide explicit or implicit feedbacks to guide the concept set expansion model learning.

CHAPTER 3: TAXONOMY CONSTRUCTION

3.1 OVERVIEW AND MOTIVATIONS

Concept taxonomy is the backbone of many knowledge-rich applications such as question answering [60], query understanding [61], and personalized recommendation [62]. Most existing concept taxonomies (*e.g.*, MeSH [12], ACM CCS [13], Pinterest Taxonomy [14], *etc.*) are constructed by human experts or in a crowd-sourcing manner. However, such manual constructions are labor-intensive, time-consuming, unadaptable to changes, and rarely complete. As a result, automated concept taxonomy construction is in great demand.

Existing methods mostly build concept taxonomies based on the "is-A" relation (*e.g.*, a "*panda*" is a "*mammal*") [18, 19, 20] or cluster terms into hierarchically organized topics [63, 64, 65]. However, such hierarchies cannot satisfy many real-world needs due to its (1) *inflexible semantics*: many applications may need hierarchies carrying more flexible semantics such as "*city-state-country*" in a location taxonomy; and (2) *limited applicability:* the "universal" taxonomy so constructed is unlikely to fit diverse and user-specific application tasks. This motivates us to work on the *task-guided* taxonomy construction, which takes a user-provided "seed" taxonomy tree (as task guidance) along with a domain-specific corpus and generates a desired taxonomy automatically. For example, as shown in Figure 3.1, a user may provide a seed taxonomy containing only two countries and two states along with a large corpus, and our method will output a taxonomy which covers all the countries and states mentioned in the corpus.

In this chapter, we propose HiExpan, a novel framework for task-guided concept taxonomy construction. HiExpan can automatically generates a key term list[1] from the input corpus and iteratively grows the seed taxonomy. Specifically, HiExpan views all children under each taxonomy node forming a coherent set and builds the taxonomy by recursively expanding all these sets using SetExpan. While such an approach is intuitive, there are two major challenges by utilizing SetExpan to generating high-quality taxonomies: (1) modeling global taxonomy information: a concept that appears in multiple expanded sets may need conflict resolution and hierarchy adjustment accordingly, and (2) cold-start with empty initial seed set: as an example, initial seed set {"*Ontario*", "*Quebec*"} will need to be found once we add "*Canada*" at the country level as shown in Figure 3.1.

HiExpan consists of two novel modules for dealing with the above two challenges. First, whenever we observe a conflict (*i.e.*, the same concept appearing in multiple positions on the

[1]In this chapter, we use the word "*term*" and "*concept*" interchangeably.

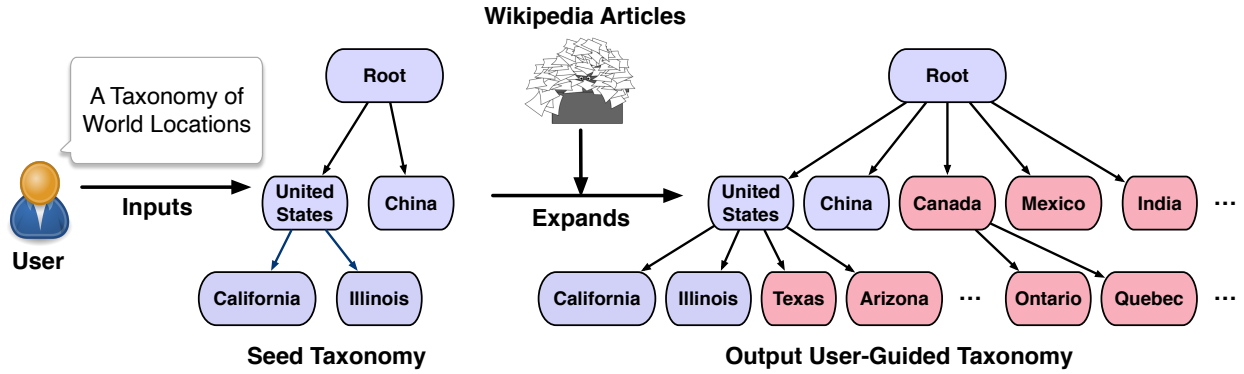


Figure 3.1: An example of task-guided taxonomy construction.

taxonomy) during the tree expansion process, we measure a "confidence score" for putting the term in each position and select the most confident position for it. Furthermore, at the end of our hierarchical tree expansion process, we will do a global optimization of the whole tree structure. Second, we incorporate a weakly-supervised relation extraction method to infer parent-child relation information and to find seed children concepts under a specific parent concept. Equipped with these two modules, HiExpan constructs the task-guided taxonomy by iteratively growing the initial seed taxonomy tree. At each iteration, it views all children under a non-leaf taxonomy node as a coherent set and builds the taxonomy by recursively expanding these sets. Whenever a node with no initial children nodes found, it will first conduct seeds hunting. At the end of each iteration, HiExpan detects all the conflicts and resolves them based on their confidence scores.

We summarize our major contributions as follows:

- We introduce a new research problem *task-guided taxonomy construction*, which takes a user-provided seed taxonomy along with a domain-specific corpus as input and aims to output a desired taxonomy that satisfies user-specific application tasks.
- We propose HiExpan, a novel expansion-based framework for task-guided taxonomy construction. HiExpan requires minimum human annotations and generates the taxonomy by growing the seed taxonomy iteratively. Special mechanisms are also taken by HiExpan to leverage global tree structure information.
- We conduct extensive experiments to verify the effectiveness of HiExpan on three real-world datasets from different domains.

The remaining of this chapter is organized as follows. Section 3.2 discusses the related work. Section 3.3 defines our problem and presents our HiExpan framework. Then, In Section 3.4, we report and analyze the experiment results. Finally, we conclude this chapter and discuss some future directions in Section 3.5.

3.2 RELATED WORK

There are three major lines of previous studies relevant to the current work.

Concept Taxonomy Construction. Most existing approaches to taxonomy construction focus on building hypernym-hyponym taxonomies wherein each parent-child pair expresses the "is-a" relation. Typically, they consist of two key steps: (1) hypernymy relation acquisition (*i.e.*, obtaining hypernym-hyponym pairs), and (2) structured taxonomy induction (*i.e.*, organizing all hypernymy relations into a tree structure).

Methods for hypernymy relation acquisition fall into two classes: pattern-based and distributional. One pioneering pattern-based method is Hearst patterns [66] in which lexical syntactic patterns (*e.g.*, "NP_x such as NP_y") are leveraged to match hypernymy relations. Later studies extend this method by incorporating more linguistic rules [67, 68, 69] or designing generalized patterns such as "*star-pattern*" [70], "*SOL pattern*" [71], and "*meta-pattern*" [72]. These methods could achieve high precision in the result pairs but often suffer low recalls (*i.e.*, many hypernym-hyponym pairs do not match the pre-defined patterns). Along another line, distributional methods predict whether a pair of terms $\langle x, y \rangle$ holds a hypernymy relation based on their distributional representations. Early studies first extract statistical features (*e.g.*, the context words of a term), calculate pairwise term similarity using symmetric metrics (*e.g.*, cosine, Jaccard) [73] or asymmetric metrics (*e.g.*, WeedsPrec [74], SLQS [75]), and predict if $\langle x, y \rangle$ holds a hypernymy relation. More recently, a collections of *supervised* methods [76, 77, 78, 79, 80] are proposed to leverage pre-trained word embeddings and curated training data to directly learn a relation classification/prediction model. However, neither pattern-based nor distributional techniques can be applied to our problem because they are designed exclusively for acquiring hypernym-hyponym pairs, whereas we aim to construct a *task-guided* taxonomy where the parent-child relations are task-specific and subject to user guidance.

For the structured taxonomy induction step, most methods first build a graph where edges represent noisy hypernymy relations, extracted in the former step, and then derive a tree-like taxonomy from this graph. Kozareva and Hovy [81] iteratively retain the longest paths between root and leaf terms and remove other conflicting edges. Velardi *et al.* [19] use the same longest-path idea to weigh edges and then find the largest-weight taxonomy as a Maximum Spanning Tree. Bansal *et al.* [82] build a factor graph to model hypernymy relations and regard taxonomy induction as a structured learning problem, which can be inferred with loop belief propagation. Recently, Gupta *et al.* [83] propose to build the initial graph using hypernym subsequence (instead of single hypernym pair) and model taxonomy induction as a minimum-cost flow problem. Comparing with these methods, our approach

leverages the weak supervision in "seed" taxonomy and builds a task-specific taxonomy in which two terms can hold a non-hypernymy relation. Further, our taxonomy construction framework jointly acquires task-specific relations and induces taxonomy structure, instead of performing the two tasks separately.

Topic Hierarchy Construction. There are a number of methods proposed for automatic topic hierarchy construction from text corpora. In pioneer studies, hierarchical topic modeling [84, 63, 65, 85] and bottom-up agglomerative clustering [86, 87] are two most popular frameworks. More recently, after word embedding technique [51] becomes mature, more top-down hierarchical clustering methods [88, 64] are proposed and achieve the new state-of-the-art. All these methods construct a topic hierarchy where each node is represented by a cluster of terms. However, finding a single concept term to summarize the term cluster is proved to be a non-trivial task [89]. In comparison, our HiExpan framework tries to construct a concept taxonomy where each node is naturally represented by a single term.

Weakly-supervised Relation Extraction. There have been studies on weakly supervised relation extraction, which aims at extracting a set of relation instances containing certain semantic relationships. Our method is related to corpus-level relation extraction that identifies relation instances from the entire text corpora [90, 91, 92, 93]. In the weakly supervised setting, there are generally two approaches for corpus-level relation extraction. The first is pattern-based [72, 71], which usually uses bootstrapping to iteratively extract textual patterns and new relation instances. The second approach [51, 94, 95] tries to learn low-dimensional representations of concepts such that concepts with similar semantic meanings have similar representations. Unfortunately, all these existing methods require a considerable amount of relation instances to train an effective relation classifier, which is infeasible in our setting as we only have a limited number seeds specified by users. Furthermore, these studies do not consider organizing the relation pairs into a taxonomy structure.

3.3 THE HIEXPAN FRAMEWORK

The input for our concept taxonomy construction framework includes two parts: (1) a text corpus $\mathcal{D}$; and (2) a "seed" taxonomy $\mathcal{T}^0$. The "seed" taxonomy $\mathcal{T}^0$, given by a user, is a tree-structured hierarchy and serves as the *task guidance*. Given the corpus $\mathcal{D}$, we aim to expand this seed taxonomy $\mathcal{T}^0$ into a more complete taxonomy $\mathcal{T}$ for the task. Each node $e \in \mathcal{T}$ represents a concept extracted from corpus $\mathcal{D}$ and each edge $\langle e_1, e_2 \rangle$ denotes a pair of concepts that satisfies the task-specific relation. We use $\mathcal{E}$ and $\mathcal{R}$ to denote all the nodes and edges in $\mathcal{T}$ and thus $\mathcal{T} \stackrel{\text{def}}{=} (\mathcal{E}, \mathcal{R})$.

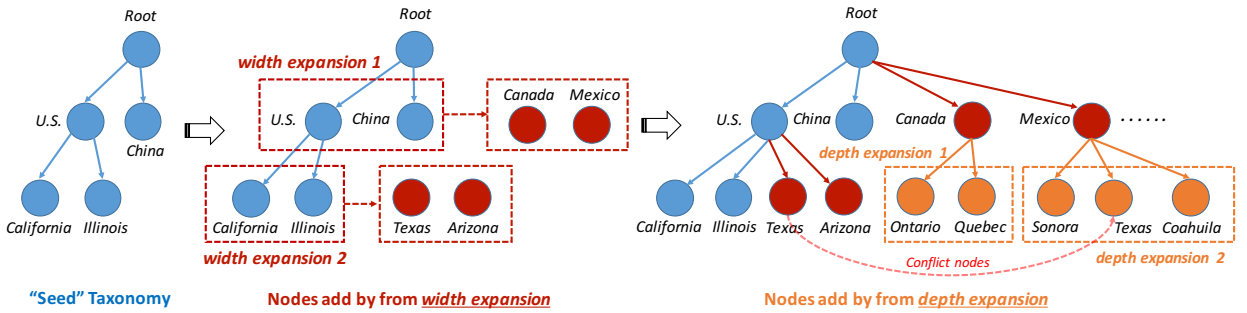


Figure 3.2: An overview of our hierarchical tree expansion algorithm in HiExpan.

Figure 3.1 shows an example of our problem. Given a collection of Wikipedia articles (*i.e.*, $\mathcal{D}$) and a "seed" taxonomy containing two countries and two states in the "*U.S.*" (*i.e.*, $\mathcal{T}^0 = (\mathcal{E}^0, \mathcal{R}^0)$), we aim to output a taxonomy $\mathcal{T}$ which covers all countries and states mentioned in corpus $\mathcal{D}$ and connects them based on the task-specific relation "*located in*", indicated by $\mathcal{R}^0$.

3.3.1 Framework Overview

In short, HiExpan views all children under each taxonomy node forming a coherent *set*, and builds the taxonomy by recursively expanding all these sets. As shown in Figure 3.2, two first-level nodes (*i.e.*, "*U.S.*" and "*China*") form a set representing the semantic class "*Country*" and by expanding it, we can obtain all the other countries. Similarly, we can expand the set {"*California*", "*Illinois*"} to find all the other states in the U.S.

Given a corpus, HiExpan first extracts all key terms using a phrase mining tool followed by part-of-speech filtering (Section 3.3.2). As the generated term list contains many irrelevant terms (*e.g.*, people's names are totally irrelevant to a location taxonomy), we use a set expansion technique to carefully select best terms. We refer this process as *width expansion* as it increases the *width* of taxonomy tree (Section 3.3.3). Furthermore, to address the challenge that some nodes do not have an initial child (*e.g.*, the node "*Mexico*" in Figure 3.2), we find the "seed" children by applying a weakly-supervised relation extraction method, which we refer as *depth expansion* (Section 3.3.4). By iteratively applying these two expansion modules and resolving possible conflicts (Section 3.3.5), our hierarchical tree expansion algorithm will first grow the taxonomy to its full size. Finally, we adjust the taxonomy tree by optimizing its global structure (Section 3.3.6).

3.3.2 Key Term Extraction

We use AutoPhrase, a state-of-the-art phrase mining algorithm [96], to extract all key terms in the given corpus. After that, we apply a Part-of-Speech (POS) tagger to the corpus and obtain the POS tag sequence of each key term occurrence. Then, we retain the key term occurrence whose corresponding POS tag sequence contains a noun POS tag (*e.g.*, "*NN*", "*NNS*", "*NNP*"). Finally, we aggregate the key terms that have at least one remaining occurrence in the corpus into the key term list.

3.3.3 Width Expansion

Width expansion aims to find the sibling nodes of a given set of children nodes which share the same parent. This naturally forms a set expansion problem and thus we adapt the SetExpan algorithm for addressing it.

Features. We use three types of concept features during the width expansion process:

- *skip-pattern*: Given a target concept e_i in a sentence, one of its skip-pattern features is "w_{-1} _ w_1" where w_{-1} and w_1 are two context words and e_i is replaced with a placeholder. One advantage of skip-pattern feature is that it imposes strong positional constraints. Following [1, 33], we extract up to six skip-patterns of different lengths for one target concept e_i in each sentence.
- *concept embedding*: We use the SkipGram model in word2vec [51] to learn the concept embeddings. For a multi-gram term (*e.g.*, "*Baja California*"), we first use "_" to concatenate tokens and then learn the embedding of this concept. The advantage of concept embedding feature is that it captures the semantics of each concept.
- *concept type*: We obtain each concept type information by linking it to Probase [18]. The return types serve as the features of that concept. For concepts that are not linkable, they simply do not have this concept's type feature.

Similarity Measures. With above defined concept features, we can compute the sibling similarity of two concepts e_1 and e_2, denoted as $sim_{sib}(e_1, e_2)$. We first assign the weight between each pair of concept and skip-pattern as: $f_{e,sk} = \log(1+X_{e,sk})\,[\log|V| - \log(\sum_{e'} X_{e',sk})]$, where $X_{e,sk}$ is the raw co-occurrence count between concept e and skip-pattern sk, and $|V|$ is the total number of candidate concepts. Similarly, we can define the association weight between a concept and a type as: $f_{e,ty} = \log(1 + C_{e,ty})\,[\log|V| - \log(\sum_{e'} C_{e',ty})]$, where $C_{e,ty}$ is the confidence score returned by Probase and indicates how confident it believes that concept e has a type ty.

After that, we calculate the similarity of two sibling concepts using skip-pattern features

as follows:

$$sim^{sk}_{sib}(e_1, e_2|SK) = \frac{\sum_{sk \in SK} \min(f_{e_1,sk}, f_{e_2,sk})}{\sum_{sk \in SK} \max(f_{e_1,sk}, f_{e_2,sk})}, \tag{3.1}$$

where SK denotes a selected set of "discriminative" skip-pattern features (see below for details). Similarly, we can calculate $sim^{tp}_{sib}(e_1, e_2)$ using all the type features. Finally, we use the cosine similarity to compute the similarity between two concepts based on their embedding features $sim^{emb}_{sib}(e_1, e_2)$. To combine the above three similarities, we notice that a good pair of sibling concepts should appear in similar contexts, share similar embeddings, and have similar types. Therefore, we use a multiplicative measure to calculate the final sibling similarity as follows:

$$sim_{sib}(e_1, e_2|SK) = \sqrt{(1 + sim^{sk}_{sib}(e_1, e_2|SK)) \cdot sim^{emb}_{sib}(e_1, e_2)} \cdot \sqrt{1 + sim^{tp}_{sib}(e_1, e_2)}. \tag{3.2}$$

The Width Expansion process. Given a seed concept set S and a candidate concept list V, a straightforward idea to compute each candidate concept's average similarity with all concepts in the seed set S using all the features. However, this approach can be problematic because (1) the feature space is huge (*i.e.*, there are millions of possible skip-pattern features) and noisy, and (2) the candidate concept list V is also noisy in the sense that many concepts in V are completely irrelevant to S. Therefore, we take a more conservative approach by first selecting a set of quality skip-pattern features and then scoring a concept only if it is associated with at least one quality skip-pattern feature.

Starting with the seed set S, we first score each skip-pattern feature based on its accumulated strength with concepts in S (*i.e.*, $score(sk) = \sum_{e \in S} f_{e,sk}$), and then select top 200 skip-pattern features with maximum scores. After that, we use sampling without replacement method to generate 10 subsets of skip-pattern features $SK_t, t = 1, 2, \ldots, 10$. Each subset SK_t has 120 skip-pattern features. Given an SK_t, we will consider a candidate concept in V only if it has association will at least one skip-pattern feature in SK_t. The score of a considered concept is calculated as follows:

$$score(e|S, SK_t) = \frac{1}{|S|} \sum_{e' \in S} sim_{sib}(e, e'|SK_t). \tag{3.3}$$

For each SK_t, we can obtain a rank list of candidate concepts L_t based on their scores. We use r^i_t to denote the rank of concept e_i in L_t and if e_i does not appear in L_t, we set $r^i_t = \infty$. Finally, we calculate the mean reciprocal rank (mrr) of each concept e_i and add those concepts with average rank above r into the set S as follows:

$$mrr(e_i) = \frac{1}{10} \sum_{t=1}^{10} \frac{1}{r^i_t}, \qquad S = S \cup \{e_i | mrr(e_i) > \frac{1}{r}\}. \tag{3.4}$$

The key insight of above aggregation mechanism is that an irrelevant concept will not appear frequently in multiple L_t at top positions and thus likely has a low mrr score. The same idea in proved effective in SetExpan and we set $r = 5$ in HiExpan.

3.3.4 Depth Expansion

The width expansion algorithm requires an initial seed concept set to start with. This requirement is satisfied for nodes in the initial seed taxonomy $\mathcal{T}^0$ as their children nodes can naturally form such a set. However, for those newly-added nodes in taxonomy tree (*e.g.*, the node "*Canada*" in Figure 3.2), they do not have any child node and thus we cannot directly apply the width expansion algorithm. To address this problem, we use *depth expansion* algorithm to acquire a target node's initial children by considering the relations between its sibling nodes and its niece/nephew nodes.

Example 3.1 (Depth Expansion) *Consider the node "*Canada*" in Figure 3.2 as an example. This node is generated by the previous width expansion algorithm and thus does not have any child node. We aim to find its initial children (i.e., "*Ontario*" and "*Quebec*") by modeling the relation between the siblings of node "*Canada*" (e.g., "*U.S.*") and its niece/nephew node (e.g., "*California*", "*Illinois*"). Similarly, given the target node "*Mexico*", we want to find its initial children such as node "*Sonora*".*

Our depth expansion algorithm relies on concept embeddings, which encode the concept semantics in a fix-length dense vector. We use $\mathbf{v}(t)$ to denote the embedding vector of concept t. As shown in [51, 77, 79], the offset of two concepts' embeddings can represent the relationship between them, which leads to the following observation that $\mathbf{v}(\text{“}U.S\text{”}) - \mathbf{v}(\text{“}California\text{”}) \approx \mathbf{v}(\text{“}Canada\text{”}) - \mathbf{v}(\text{“}Ontario\text{”})$. Therefore, given a target parent node e_t, a set of reference edges $E = \{\langle e_p, e_c\rangle\}$ where e_p is the parent node of e_c, we calculate the "goodness" of putting node e_x under parent node e_t as follows:

$$sim_{par}(\langle e_t, e_x\rangle) = cos\left(\mathbf{v}(e_t) - \mathbf{v}(e_x), \frac{1}{|E|}\sum_{\langle e_p, e_c\rangle} \mathbf{v}(e_p) - \mathbf{v}(e_c)\right), \tag{3.5}$$

where $cos(\mathbf{v}(x), \mathbf{v}(y))$ denotes the cosine similarity between vector $\mathbf{v}(x)$ and $\mathbf{v}(y)$. Finally, we score each candidate concept e_i based on $sim_{par}(\langle e_t, e_i\rangle)$ and select top-3 concepts with maximum score as the initial children nodes under node e_t.

The concept embedding is learned from REPEL [90], a model for weakly-supervised relation extraction using pattern-enhanced embedding learning. It takes a few seed relation mentions (e.g. "US-Illinois" and "US-California") and outputs concept embeddings as well

as reliable relational phrases for target relation type(s). REPEL consists of a pattern module which learns a set of reliable textual patterns, and a distributional module, which learns a relation classifier on concept representations for prediction. As both modules provide extra supervision for each other, the distributional module learns concept embeddings supervised by more reliable patterns from the pattern module. By doing so, the learned concept embeddings carry more useful information than those obtained from other embedding models like word2vec [51], specifically for finding relation tuples of the target relation type(s).

3.3.5 Conflict Resolution

We can *iteratively* apply width expansion and depth expansion to grow the taxonomy tree to its full size. As the supervision signal from the user-specified seed taxonomy $\mathcal{T}^0$ is very weak, we need to make sure those nodes introduced in the first several iterations are of high quality and will not mislead the expansion process in later iterations to a wrong direction. In HiExpan, for each task-related concept, we aim to find its single best position on our output task-guided taxonomy $\mathcal{T}$. Therefore, when finding a concept appears in multiple positions during our tree expansion process, we say a "conflict" happens and aim to resolve such conflict by finding the best position that concept should reside in. Given a set of conflicting nodes $\mathcal{C}$ which corresponds to different positions of a same concept, we apply the following three rules to select the best node out of this set. First, if any node is in the seed taxonomy $\mathcal{T}^0$, we directly select this node and skip the following two steps. Otherwise, for each pair of nodes in $\mathcal{C}$, we check whether one of them is the ancestor of the other and retain only the ancestor node. After that, we calculate the "confidence score" of each remaining node $e \in C$ as follows:

$$conf(e) = \frac{1}{|sib(e)|} \sum_{e' \in sib(e)} sim_{sib}(e, e'|SK) \cdot sim_{par}(\langle par(e), e \rangle), \tag{3.6}$$

where $sib(e)$ denotes the set of all sibling nodes of e and $par(e)$ represents its parent node. The skip-pattern feature in SK is selected based on its accumulated strength with concepts in $sib(e)$. The node with highest confidence score will be selected. Finally, for each node in $\mathcal{C}$ that is not selected, we will delete the whole subtree rooted by it, cut all the sibling nodes added after it, and put it in its parent node's "children backlist".

Example 3.2 (Conflict Resolution) *In Figure 3.2, we can see there are two "*Texas*" nodes, one under "*U.S.*" and the other under "*Mexico*". As none of them is from initial "seed" taxonomy and they do not hold an ancestor-descendant relationship, we need to*

*calculate each node's confidence score based on Eq. (3.6). Since "*Texas*" has a stronger relation with other states in U.S., comparing with those in Mexico, we will select the "*Texas*" node under "*U.S.*". Then, for the other node under "*Mexico*", we will delete it and cut "*Coahuila*", a sibling node added after "*Texas*". Finally, we let the node "*Mexico*" to remember that "*Texas*" is not one of its children, which prevents the "*Texas*" node being added back later. Notice that although the "*Coahuila*" node is cut here, it may be added back in a later iteration by our tree expansion algorithm.*

3.3.6 Taxonomy Global Optimization

In the above hierarchical tree expansion algorithm, a node will be selected and attached onto the taxonomy based on its "local" similarities with other sibling nodes and its parent node. While modeling only the "local" similarity can simplify the tree expansion process, we find the resulting taxonomy may not be the best from a "global" point of view. For example, when expanding the France regions, we find that "Molise", an Italy region, will be mistakenly added under the "France" node, likely because it shares many similar contexts with some other regions of France. However, when we take a global view of the taxonomy and ask the following question—*which country is Molise located in?*, we can easily put "Molise" under "Italy" as it shares more similarities with those in Italy than in France.

Motivated by the above example, we propose a *taxonomy global optimization module* in HiExpan. The key idea is to adjust each two contiguous levels of the taxonomy tree and to find the best "parent" node at the upper level for each "child" node at the lower level. Our taxonomy global optimization makes the following two hypotheses: (1) concepts that have the same parent are similar to each other and form a coherent set, and (2) each concept is more similar to its correct parent compared with other siblings of its correct parent.

Formally, suppose there are m "parent" nodes at the upper level and n "child" nodes at the lower level, we use $\mathbf{W} \in \mathbb{R}^{n\times n}$ to model the concept-concept sibling similarity and use $\mathbf{Y^c} \in \mathbb{R}^{n\times p}$ to capture the two concepts's parenthood similarity. We let $\mathbf{W_{ij}} = sim_{sib}(e_i, e_j)$ if $i \neq j$, otherwise we set $\mathbf{W_{ii}} = 0$. We set $\mathbf{Y^c_{ij}} = sim_{par}(\langle e_j, e_i\rangle)$. Furthermore, we define another $n \times p$ matrix $\mathbf{Y}^s$ with $\mathbf{Y^s_{ij}} = 1$ if a child node e_i is under parent node e_j and $\mathbf{Y^s_{ij}} = 0$ otherwise. This matrix captures the current parent assignment of each child node. We use $\mathbf{F} \in \mathbb{R}^{n\times p}$ to represent the child nodes' parent assignment we intend to learn. Given a $\mathbf{F}$, we can assign each "child" node e_i to a "parent" node $e_j = \arg\max_j \mathbf{F_{ij}}$. Finally, we propose the following optimization problem to reflect the previous two hypotheses:

$$\min_{\mathbf{F}} \sum_{i,j}^{n} \mathbf{W_{ij}} \left\| \frac{\mathbf{F_i}}{\sqrt{\mathbf{D_{ii}}}} - \frac{\mathbf{F_j}}{\sqrt{\mathbf{D_{jj}}}} \right\|_2^2 + \mu_1 \sum_{i=1}^{n} \left\| \mathbf{F_i} - \frac{\mathbf{Y^c_i}}{\|\mathbf{Y^c_i}\|_1} \right\|_2^2 + \mu_2 \sum_{i=1}^{n} \|\mathbf{F_i} - \mathbf{Y^s_i}\|_2^2 , \tag{3.7}$$

where $\mathbf{D_{ii}}$ is the sum of i-th row of $\mathbf{W}$, and μ_1, μ_2 are two nonnegative model hyper-parameters. The first term in Eq. (3.7) corresponds to our first hypothesis and models two concepts' sibling similarity. The second term in Eq. (3.7) follows our second hypothesis to model the parenthood similarity. The last term in Eq. (3.7) serves as the smoothness constraints and captures the taxonomy structure information before the global adjustment.

To solve the above optimization problem, we take the derivative of its objective function with respect to $\mathbf{F}$ and can obtain the following closed form solution:

$$\mathbf{F}^* = (\mathbf{I} - \alpha S)^{-1} \cdot (\beta_1 \mathbf{Y^c} + \beta_2 \mathbf{Y^s}), \quad \mathbf{S} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}, \tag{3.8}$$

where $\alpha_1 = \frac{1}{1+\mu_1+\mu_2}$, $\beta_1 = \frac{\mu_1}{1+\mu_1+\mu_2}$ and $\beta_2 = \frac{\mu_2}{1+\mu_1+\mu_2}$. The calculation procedure is similar to the one in [97].

3.4 EXPERIMENTS

3.4.1 Datasets

We use two corpora from different domains to evaluate the performance of HiExpan. The first one is **DBLP** which contains about 156 thousand paper abstracts (1.1 million sentences) and a vocabulary of over 17 thousand concepts in the computer science field. The second one is **Wiki** which includes a subset of English Wikipedia pages (1.5 million sentences) used in [48, 1] and a vocabulary of more than 41 thousand concepts.

3.4.2 Compared Methods

To the best of our knowledge, we are the first to study the problem of task-guided taxonomy construction problem, and thus there is no suitable baseline to compare with directly. Therefore, here we evaluate the effectiveness of HiExpan by comparing it with a heuristic set-expansion based method and its own variations as follows:

- HSetExpan is a baseline method which iteratively applies SetExpan algorithm [1] at each level of taxonomy. For each lower level node, this method finds its best parent node to attach according to the children-parent similarity measure defined in Eq. (3.5).
- NoREPEL is a variation of HiExpan without the REPEL [90] module which jointly leverages pattern-based and distributional methods for concept embedding learning. Instead, we use the SkipGram model [51] for learning concept embeddings.
- NoGTO is a variation of HiExpan without the taxonomy global optimization module. It directly outputs the taxonomy generated by hierarchical tree expansion algorithm.

- HiExpan is the full version of our proposed framework, with both REPEL embedding learning module and taxonomy global optimization module enabled.

We use the above methods to generate two taxonomies, one for each corpus. When extracting the key term list using AutoPhrase [96], we treat phrases that occur over 15 times in the corpus to be frequent. The embedding dimension is set to 100 in both REPEL [90] and SkipGram model [51]. The maximum expansion iteration number *max_iter* is set to 5 for all above methods. Finally, we set the two hyper-parameters used in taxonomy global optimization module as $\mu_1 = 0.1$ and $\mu_2 = 0.01$.

3.4.3 Evaluation Metrics

Evaluating the quality of an entire taxonomy is challenging due to the existence of multiple aspects that should be considered and the difficulty of obtaining gold standard [20]. Following [98, 99], we use **Ancestor-F1** and **Edge-F1** for taxonomy evaluation in this study. **Ancestor-F1** measures correctly predicted ancestral relations. It enumerates all the pairs on the predicted taxonomy and compares these pairs with those in the gold standard taxonomy as follows:

$$P_a = \frac{|\text{is-ancestor}_{\text{pred}} \cap \text{is-ancestor}_{\text{gold}}|}{|\text{is-ancestor}_{\text{pred}}|}, \tag{3.9}$$

$$R_a = \frac{|\text{is-ancestor}_{\text{pred}} \cap \text{is-ancestor}_{\text{gold}}|}{|\text{is-ancestor}_{\text{gold}}|}. \tag{3.10}$$

$$F1_a = \frac{2P_a * R_a}{P_a + R_a} \tag{3.11}$$

We denote P_a, R_a, $F1_a$ as the ancestor precision, ancestor recall, and ancestor F1-score, respectively. **Edge-F1** compares edges predicted by different taxonomy construction methods with edges in the gold standard taxonomy. Similarly, we denote edge-based metrics as P_e, R_e, and $F1_e$, respectively.

To construct the gold standard, we extract all the parent-child edges in taxonomies generated by all compared methods. Then we pool all the edges together and ask five people, to judge these pairs independently. We show them seed parent-child pairs as well as the generated parent-child pairs, and ask them to evaluate whether the generated parent-child pairs have the same relation as the given seed parent-child pairs. After collecting these answers from the annotators, we simply use majority voting to label the pairs. We then use these annotated data as the gold standard.

Table 3.1: Qualifications of the taxonomies constructed by HSetExpan, NoREPEL, NoGTO, and HiExpan.

Methods	Wiki						DBLP					
	P_a	R_a	$F1_a$	P_e	R_e	$F1_e$	P_a	R_a	$F1_a$	P_e	R_e	$F1_e$
HSetExpan	0.740	0.444	0.555	0.759	0.471	0.581	0.743	**0.448**	**0.559**	0.739	0.448	0.558
NoREPEL	0.696	0.596	0.642	0.697	0.576	0.631	0.722	0.384	0.502	0.705	**0.464**	0.560
NoGTO	0.827	0.708	0.763	0.810	0.671	0.734	0.821	0.366	0.506	0.779	0.433	0.556
HiExpan	**0.847**	**0.725**	**0.781**	**0.848**	**0.702**	**0.768**	**0.843**	0.376	0.520	**0.829**	0.460	**0.592**

3.4.4 Quantitative Results

Table 3.1 shows both the ancestor-based and edge-based precision/recalls as well as F1-scores of different methods. We can see that HiExpan achieves the best overall performance, and outperforms other methods, especially in terms of the precision. By comparing the performance of HiExpan, NoREPEL, and NoGTO, we can see that both the REPEL module and the taxonomy global optimization algorithm in HiExpan play important roles in improving the quality of the generated taxonomy. Specifically, REPEL learns more discriminative representations by iteratively letting the distributional module and pattern module mutually enhance each other, and the taxonomy global optimization module leverages the global information from the entire taxonomy tree structure. In addition, HiExpan resolves the "conflicts" at the end of each tree expansion iteration by cutting many nodes on a currently expanded taxonomy. This leads HiExpan to generate a smaller tree comparing with the one generated by HSetExpan, given that both methods running the same number of iterations. However, we can see that HiExpan still beats HSetExpan on Wiki dataset, in terms of the recall, which further demonstrates the effectiveness of HiExpan.

3.4.5 Case Studies

In Figure 3.3, we show the taxonomy trees generated by HiExpan in two domains. First, given a "seed" taxonomy containing two countries and six states/provinces. we can expand it to a full location taxonomy which covers all countries and state/provinces mentioned in the corpus and connects them based the "country-state/province" relation. Similarly, we can expand a seed computer science area taxonomy to automatically discover many other CS subareas. We can also zoom in to look at the taxonomy at a more granular level. Taking the node "natural language processing" as an example, HiExpan successfully finds major subtopics in natural language processing such as "question answering", "text summarization", and "word sense disambiguation" even without any initial seed concepts.

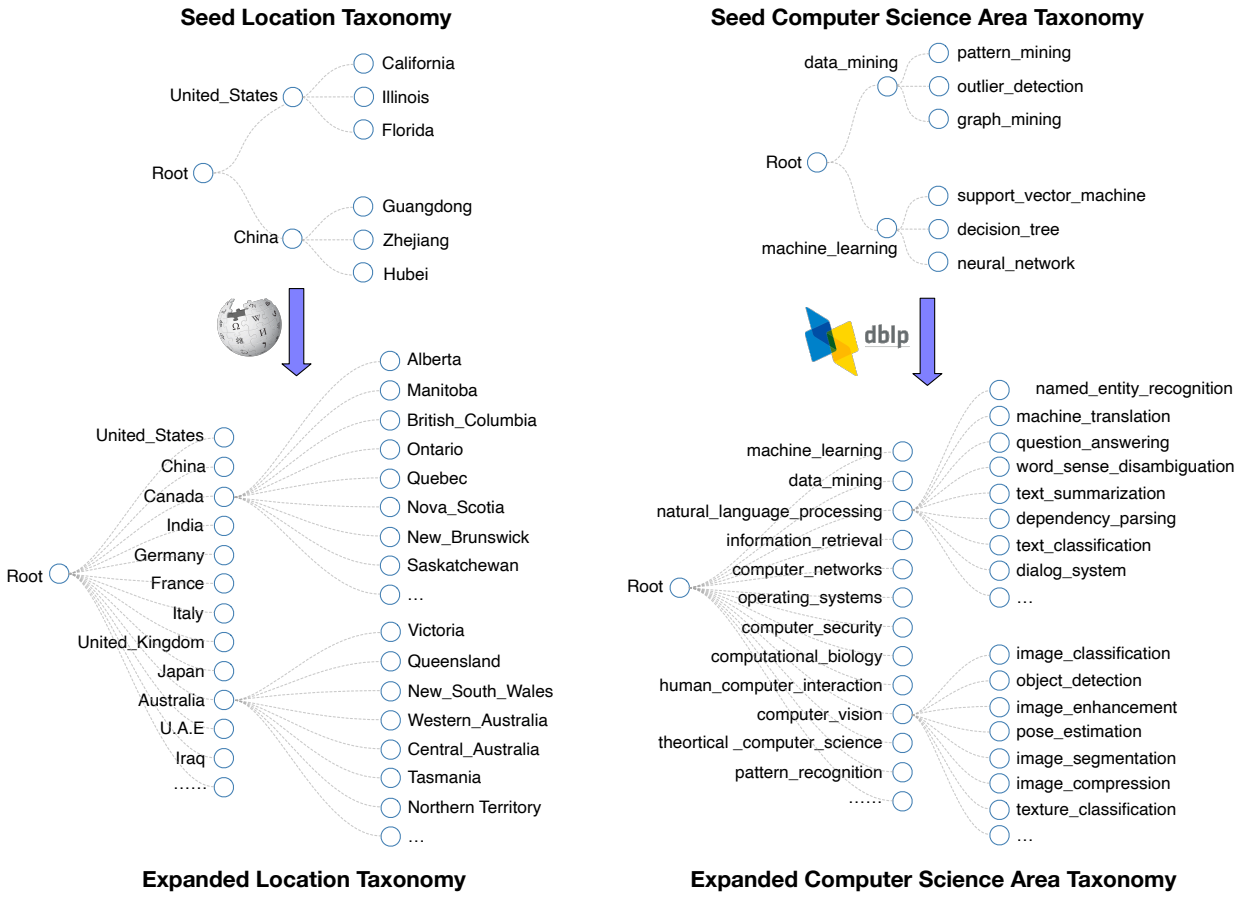


Figure 3.3: The taxonomy trees generated by HiExpan in two different domains.

Table 3.2 shows the effect of taxonomy global optimization module in HiExpan. From the experiment on the Wiki dataset, we observe that the node "London" was originally attached to "Australia", but after applying the taxonomy global optimization module, this node is correctly moved under "England". Similarly, in the DBLP dataset, the term "unsupervised learning" was initially located under "data mining" but later being moved under the parent node "machine learning". This demonstrates the effectiveness of our taxonomy global optimization module.

3.5 SUMMARY

In this chapter, we explore how to construct a task-guided concept taxonomy based on the initial user-provided seed taxonomy. We propose HiExpan, an effective expansion-based concept taxonomy construction framework that grows the seed taxonomy by recursive expansions. In addition, we discuss how to incorporate a weakly-supervised relation extraction

Table 3.2: NoGTO shows the parent of a concept before applying taxonomy structure optimization. HiExpan shows the parent node of this concept after optimizing the taxonomy structure.

Dataset	Concept	NoGTO	HiExpan
Wiki	London	Australia	England
	Chiba	China	Japan
	Molise	Frances	Italy
	New_South_Wales	England	Australia
	Shropshire	Scotland	England
DBLP	unsupervised_learning	data_mining	machine_learning
	social_network_analysis	natural_language_processing	data_mining
	multi-label_classification	information_retrieval	machine_learning
	pseudo-relevance_feedback	computational_biology	information_retrieval
	function_approximate	data_analysis	machine_learning

module to infer parent-child concept relations and adjust the taxonomy tree by optimizing its global structure.

In the future, we plan to extend HiExpan by incorporating new concept set expansion methods (including our own studies SetCoExpan and CGExpan), embedding learning methods [100, 101], and more supervision signals from either existing knowledge bases [4] or pretrained language models [2]. Furthermore, the current taxonomy construction process as well as the evaluation metrics are task-agnostic. Therefore, separate modules need to be designed to apply this taxonomy for different applications. We plan to study an application-guided taxonomy construction method that leverages the performance of a downstream application to guide the upstream taxonomy construction process. Moreover, the current HiExpan takes in user guidances (from the seed taxonomy) only at the initial stage. We plan to extend this framework to allow users provide feedbacks within the whole iterative expansion process. Finally, another interesting direction is to study how the topic taxonomy (where each node contains a set of terms) could be integrated with HiExpan's output single term based taxonomy, which allows for a more flexible and interpretable taxonomy structure.

CHAPTER 4: TAXONOMY ENRICHMENT

4.1 OVERVIEW AND MOTIVATIONS

Taxonomies have been fundamental to organizing knowledge for centuries. In today's Web, taxonomies provide valuable knowledge to support many applications such as query understanding [61], content browsing [102], personalized recommendation [62, 103], and web search [18, 104]. For example, many online retailers (*e.g.*, `eBay` and `Amazon`) organize products into categories of different granularities, so that customers can easily search and navigate this category taxonomy to find the items they want to purchase. In addition, web search engines (*e.g.*, `Google` and `Bing`) leverage a taxonomy to better understand user queries and improve the search quality.

As the web contents and human knowledge are constantly growing, people need to expand an existing taxonomy to include new emerging concepts. Most of previous methods, however, construct a taxonomy entirely *from scratch* and thus when we add new concepts, we have to re-run the entire taxonomy construction process. Although being intuitive, this approach has several limitations. First, many taxonomies have a top-level design provided by domain experts and such design shall be preserved. Second, a newly constructed taxonomy may not be consistent with the old one, which can lead to instabilities of its dependent downstream applications. Finally, as targeting the scenario of building taxonomy from scratch, most previous methods are unsupervised and cannot leverage signals from the existing taxonomy to construct a new one.

In this chapter, we study the *taxonomy expansion* task: given an existing taxonomy and a set of new emerging concepts, we aim to automatically expand the taxonomy to incorporate these new concepts (without changing the existing relations in the given taxonomy). Figure 4.1 shows an example where a taxonomy in computer science domain is expanded to include new subfields (*e.g.*, "*Quantum Computing*") and new techniques (*e.g.*, "*Meta Learning*" and "*UDA*"). Some previous studies [105, 106, 107, 108] attempt this task by using an additional set of labeled concepts with their true insertion positions in the existing taxonomy. However, such labeled data are usually small and thus forbid us from learning a more powerful model that captures the subsumption semantics in the existing taxonomy.

We propose a novel framework named TaxoExpan to tackle the lack-of-supervision challenge. TaxoExpan formulates a taxonomy as a directed acyclic graph (DAG), automatically generates pseudo-training data from the existing taxonomy, and uses them to learn a matching model for expanding a given taxonomy. Specifically, we view each concept in the existing

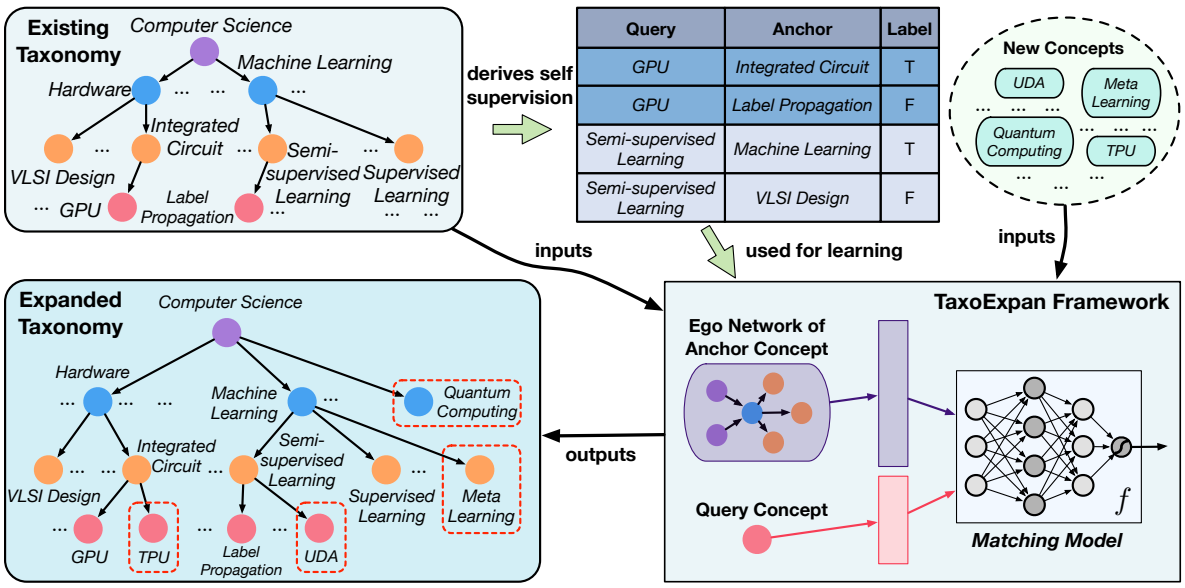


Figure 4.1: An example of expanding one computer science field-of-studies taxonomy to include new concepts.

taxonomy as a *query* and one of its parent concepts as an *anchor*. This gives us a set of positive ⟨query concept, anchor concept⟩ pairs. Then, we generate negative pairs by sampling those concepts that are neither the descendants nor the direct parents of the query concept in the existing taxonomy. In Figure 4.1, for example, the ⟨"*GPU*", "*Integrated Circuit*"⟩ is a positive pair and ⟨"*GPU*", "*Label Propagation*"⟩ is a negative pair. We refer to these training pairs as *self-supervision* data, because they are procedurally generated from the existing taxonomy and no human curation is involved.

To make the best use of above self-supervision data, we develop two novel techniques in TaxoExpan. The first one is a position-enhanced graph neural network (GNN) which encodes the local structure of an anchor concept using its ego network (egonet) in the existing taxonomy. If we view this anchor concept as the "parent" of the query concept, this ego network includes the potential "siblings" and "grand parents" of the query concept. We apply graph neural networks (GNNs) to model this ego network. However, regular GNNs fail to distinguish nodes with different relative positions to the query (*i.e.*, some nodes are grand parents of the query while the others are siblings of the query). To address this limitation, we present a simple but effective enhancement to inject such position information into GNNs using position embedding. We show that such embedding can be easily integrated with existing GNN architectures (*e.g.*, GCN [109] and GAT [110]) and significantly boosts the prediction performance. The second technique is a new noise-robust training scheme based on the InfoNCE loss [111]. Instead of predicting whether each individual ⟨query concept, anchor concept⟩

pair is positive or not, we first group all pairs sharing the same query concept into a single training instance and learn a model to select the positive pair among other negative ones from the group. We show that such training scheme is robust to the label noise and leads to performance gains.

We test the effectiveness of TaxoExpan framework on three real-world taxonomies from different domains. Our results show that TaxoExpan can generate high-quality concept taxonomies in scientific domains and achieves state-of-the-art performance on the WordNet taxonomy expansion challenge [106].

To summarize, our major contributions include:

- We propose a self-supervised framework TaxoExpan that automatically expands existing taxonomies without manually labeled data.
- We develop an effective method for enhancing graph neural network by incorporating hierarchical positional information.
- We design a new training objective that enables the learned model to be robust to label noises in self-supervision data.
- We conduct extensive experiments that verify both the effectiveness and the efficiency of TaxoExpan framework on three real-world taxonomies from different domains.

The rest of this chapter is organized as follows. Section 4.2 discusses the related work. Section 4.3 formalizes our problem. Then, we present our TaxoExpan framework in Section 4.4 and conduct experiments in Section 4.5. Finally, we conclude this chapter in Section 4.6.

4.2 RELATED WORK

In many real-world applications, some existing taxonomies may have already been laboriously curated by experts or via crowdsourcing, and are deployed in online systems. Instead of constructing the entire taxonomy from scratch, these applications demand the feature of expanding an existing taxonomy dynamically. There exists some studies on expanding WordNet with named entities from Wikipedia [112] or domain-specific concepts from different corpora [113, 114, 105, 115]. Task 14 of SemEval 2016 challenge [106] is specifically setup to enrich WordNet with concepts from domains like health, sport, and finance. One limitation of these approaches is that they depend on the synset structure unique to WordNet and thus cannot be easily generalized to other taxonomies.

To address the above limitation, more recent works try to develop methodologies for expanding a generic taxonomy. Wang *et al.* [116] design a hierarchical Dirichlet model to extend the category taxonomy in search engines using query logs. Plachouras *et al.* [117] learn paraphrase models on external PPDB datasets and apply learned models to directly

find paraphrases of concepts in the existing taxonomy. Vedula *et al.* [118] combine multiple features, some of which are retrieved from an external `Bing` Search API, into a ranking model to score candidate positions in terms of their matching scores with the query concept. Aly *et al.* [119] first learn term embeddings in a hyperbolic space and then attach each new concept to its most similar node in the existing taxonomy based on the hyperbolic embeddings. Comparing with these methods, our proposed framework (details in the next section) has two advantages. First, it requires no additional resource and makes full use of the existing taxonomy as the self supervision, which leads to a border application scope. Second, it explicitly models the local structure around each candidate position, which boosts the quality of expanded taxonomy.

Our work is also related to Graph Neural Network (GNN) which is a generic method of learning on graph-structure data. Many GNN architectures have been proposed to either learn individual node embeddings [109, 120, 121, 110] for the node classification and the link prediction tasks or learn an entire graph representation [122, 123, 124] for the graph classification task. In this work, we tackle the taxonomy expansion task with a fundamentally different formulation from previous tasks. We leverage some existing GNN architectures and enrich them with additional relative position information. Recently, You *et al.* [125] propose a method to add position information into GNN. Our methods are different from You *et al.*. They model the *absolute* position of a node in a full graph without any particular reference points; while our technique captures the *relative* position of a node with respect to the query node. Finally, some work on graph generation [126, 127, 128] involves a module to add a new node into a partially generated graph, which shares the similar goal as our model. However, such graph generation model typically requires fully labeled training data to learn from. To the best of our knowledge, this is the first study on how to expand an existing directed acyclic graph (as we model a taxonomy as a DAG) using self-supervised learning.

4.3 PROBLEM FORMULATION

In this section, we first define a taxonomy, then formulate our problem, and finally discuss the scope of our study.

Taxonomy. A taxonomy $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ is a directed acyclic graph where each node $n \in \mathcal{N}$ represents a concept (*i.e.*, a word or a phrase) and each directed edge $\langle n_p, n_c \rangle \in \mathcal{E}$ indicates a relation expressing that concept n_p is the most specific concept that is more general than concept n_c. In other words, we refer to n_p as the "*parent*" of n_c and n_c as the "*child*" of n_p.

Problem Definition. The input of the *taxonomy expansion task* includes two parts: (1)

an existing taxonomy $\mathcal{T}^0 = (\mathcal{N}^0, \mathcal{E}^0)$, and (2) a set of new concepts $\mathcal{C}$. This new concept set can be either manually specified by users or automatically extracted from text corpora. Our goal is to expand the existing taxonomy $\mathcal{T}^0$ into a larger taxonomy $\mathcal{T} = (\mathcal{N}^0 \cup \mathcal{C}, \mathcal{E}^0 \cup \mathcal{R})$, where $\mathcal{R}$ is a set of newly discovered relations each including one new concept $c \in \mathcal{C}$.

Example 4.1 (Taxonomy Expansion) *Figure 4.1 shows an example of our problem. Given a field-of-study taxonomy $\mathcal{T}^0$ in the computer science domain and a set of new concepts $\mathcal{C} = \{$ "UDA", "Meta Learning", $\dots\}$, we find each new concept's best position in $\mathcal{T}^0$ (e.g., "UDA" under "Semi-supervised Learning" as well as "GPU" under "Integrated Circuit") and expand $\mathcal{T}^0$ to include those new concepts.*

Simplified Problem. A simplified version of the above problem is that we assume the input set of new concepts contains only one element (*i.e.*, $|\mathcal{C}| = 1$), and we aim to find one single parent node of this new concept (*i.e.*, $|\mathcal{R}| = 1$). We discuss the connection between these two problem settings at the end of Section 4.4.1.

Discussion. In this work, we follow previous studies [106, 118, 119] and assume each concept in $\mathcal{N}^0 \cup \mathcal{C}$ has an initial embedding vector learned from this concept's surface name, or if available, its definition sentences [107] and associated web pages [116]. We also note that our problem formulation assumes those relations in the existing taxonomy are not modified. We acknowledge that such modification is necessary in some cases, but it is much less frequent and requires high cautiousness from human curators. Therefore, we leave it out of the scope of automation in this study.

4.4 THE TAXOEXPAN FRAMEWORK

In this section, we first introduce our taxonomy model and expansion goal. Then, we elaborate how to represent a query concept and an insertion position (*i.e.*, an anchor concept), based on which we present our query-concept matching model. Finally, we discuss how to generate self-supervision data from the existing taxonomy and use them to train the TaxoExpan framework.

4.4.1 Taxonomy Model and Expansion Goal

A taxonomy $\mathcal{T}$ describes a hierarchical organization of concepts. These concepts form the node set $\mathcal{N}$ in $\mathcal{T}$. Mathematically, we model each node $n \in \mathcal{N}$ as a categorical random variable and the entire taxonomy $\mathcal{T}$ as a Bayesian network. We define the probability of a

taxonomy $\mathcal{T}$ as the joint probability of node set $\mathcal{N}$ which can be further factorized into a set of conditional probabilities as follows:

$$\mathbf{P}(\mathcal{T}|\Theta) = \mathbf{P}(\mathcal{N}|\mathcal{T}, \Theta) = \prod_{i=1}^{|\mathcal{N}|} \mathbf{P}(n_i|parent_{\mathcal{T}}(n_i), \Theta), \tag{4.1}$$

where Θ is the set of model parameters and $parent_{\mathcal{T}}(n_i)$ is the set of n_i's parent node(s) in taxonomy $\mathcal{T}$. Given learned model parameters Θ, an existing taxonomy $\mathcal{T}^0 = (\mathcal{N}^0, \mathcal{E}^0)$, and a set of new concepts $\mathcal{C}$, we can ideally find the best taxonomy $\mathcal{T}^*$ by solving the following optimization problem:

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} \mathbf{P}(\mathcal{T}|\Theta) = \arg\max_{\mathcal{T}} \sum_{i=1}^{|\mathcal{N}^0 \cup \mathcal{C}|} \log \mathbf{P}(n_i|parent_{\mathcal{T}}(n_i), \Theta). \tag{4.2}$$

This naïve approach has two limitations. First, the search space of all possible taxonomies over the concept set $|\mathcal{N}^0 \cup \mathcal{C}|$ is prohibitively large. Second, we cannot guarantee the structure of existing taxonomy $\mathcal{T}^0$ remains unchanged, which can be undesirable from the application point of view. We address the above limitations by restricting the search space of our output taxonomy to be the exact expansion of the existing taxonomy $\mathcal{T}^0$. Specifically, we keep the parents of each existing taxonomy node $n \in \mathcal{N}^0$ unchanged and only try to find a *single* parent node of each new concept in $\mathcal{C}$. As a result, we divide the above computationally intractable problem into the following set of $|\mathcal{C}|$ tractable optimization problems:

$$a_i^* = \arg\max_{a_i \in \mathcal{N}^0} \log \mathbf{P}(n_i|a_i, \Theta), \quad \forall i \in \{1, 2, \ldots, |\mathcal{C}|\}, \tag{4.3}$$

where a_i is the parent of a new concept $n_i \in \mathcal{C}$ and we refer to it as the "*anchor concept*".

Discussion. The above equation defines $|\mathcal{C}|$ *independent* optimization problems and each problem aims to find one single parent of a new concept n_i. Therefore, we essentially reduce the more generic taxonomy expansion problem into $|C|$ independent simplified problems (c.f. Section 4.3) and tackle it by inserting new concepts *one-by-one* into the existing taxonomy. As a result of the above reduction, possible interactions among new concepts are ignored and we leave it to the future work. In the following sections, we continue to answer two keys questions: (1) how to model the conditional probability $\mathbf{P}(n_i|a_i, \Theta)$, and (2) how to learn model parameters Θ.

4.4.2 Query-Anchor Matching: Overview

We model the matching score between a query concept n_i and an anchor concept a_i by projecting them into a vector space and calculating matching scores using their vectorized representations. We show the entire model architecture of TaxoExpan in Figure 4.2.

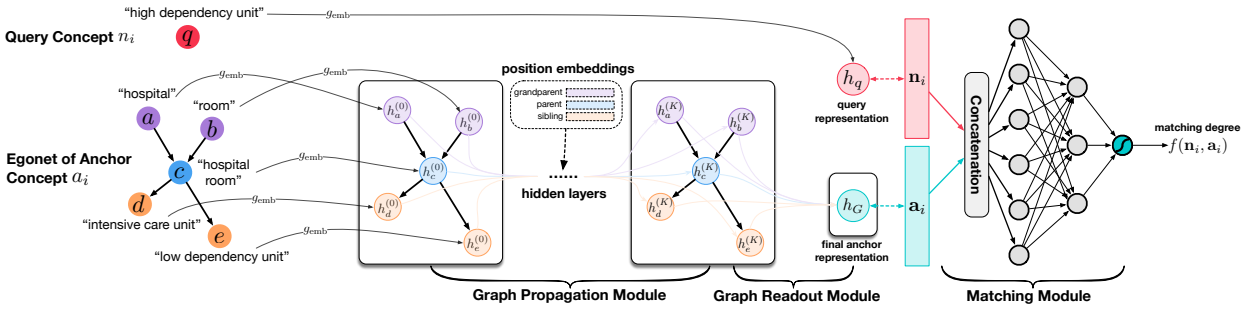


Figure 4.2: Overview of TaxoExpan framework. g_{emb} is an embedding model that provides query concept's initial feature vector h_q and the initial feature vector of each node in the egonet. The graph propagation module transforms initial feature vectors into better node representations based on which the graph readout module outputs the egonet embedding as the final anchor representation. Finally, a matching module inputs both query and anchor representations and outputs their matching score.

4.4.3 Query-Anchor Matching: Query Concept Representation

In this study, we assume each query concept has an *initial feature vector* learned based on some text associated with this concept. Such text can be as simple as the concept surface name, or in some prior studies [106, 116], the definition sentences and clicked web pages about the concept. We represent each query concept n_i using its initial feature vector denoted as $\mathbf{n}_i$. We will discuss how to obtain such initial feature vectors using embedding learning methods in the experiment section.

4.4.4 Query-Anchor Matching: Anchor Concept Representation

Each anchor concept corresponds to one node in the existing taxonomy $\mathcal{T}^0$ that could be the "*parent*" of a query concept. One naïve way to represent an anchor concept is to directly use its initial feature vector. A key limitation of this approach is that it captures only the "*parent*" node information and loses other surrounding nodes' signals, which could be crucial for determining whether the query concept should be put in this position. We illustrate this limitation below:

Example 4.2 *Suppose we are given a query concept* "high dependency unit" *to predict whether it should be under the* "hospital room" *node (i.e., an anchor concept) in an existing taxonomy. As these two concepts have dissimilar embeddings based on their surface names, we may believe this query concept shouldn't be placed underneath this anchor concept. However, if we know that this anchor concept has two children nodes, i.e.,* "intensive care unit"

and "low dependency unit"*, that are closely related to the query concept, we are more likely to put the query concept under this anchor concept, correctly.*

The above example demonstrates the importance of capturing local structure information in the anchor concept representation. Thus, we model the anchor concept using its ego network. Specifically, we consider the anchor concept to be the "*parent*" node of a query concept. The ego network of the anchor concept consists of the "*sibling*" nodes and "*grand parent*" nodes of the query concept, as shown in Figure 4.3. We represent the anchor concept based on its ego network using a graph neural network.

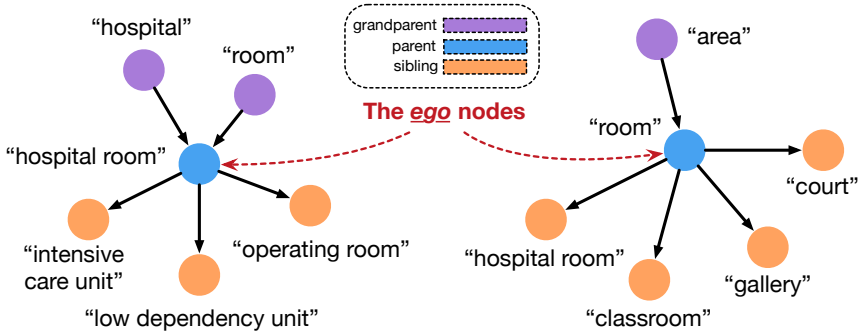


Figure 4.3: Two egonets correspond to two anchor concepts.

Graph Neural Network Architectures. Given an anchor concept a_i with its corresponding ego network G_{a_i} and its initial representation a_i, we use a graph neural network (GNN) to generate its final representation $\mathbf{a}_i$. This GNN contains two components: (1) a *graph propagation* module that transforms and propagates node features over the graph structure to compute individual node embeddings in G_{a_i}, and (2) a *graph readout* module that combines node embeddings into a vector representing the full ego network G_{a_i}. The final graph embedding encodes all local structure information centered around the anchor concept and we use it as the final anchor representation $\mathbf{a}_i$.

A graph propagation module uses a neighborhood aggregation strategy to iteratively update the representation of a node u by aggregating representations of its neighbors $N(u)$ and itself. We denote $N(u) \cup \{u\}$ as $\widetilde{N(u)}$. After K iterations, a node's representation captures the structural information within its K-hop neighborhood. Formally, we define a GNN with K-layers as follows:

$$h_u^{(k)} = \text{AGG}^{(k)}\left(\{h_v^{(k-1)} | v \in \widetilde{N(u)}\}\right), \quad k \in \{1, \ldots, K\}, \tag{4.4}$$

where $h_u^{(k)}$ is node u's feature in the k-th layer; $h_u^{(0)}$ is node u's initial feature vector, and $\text{AGG}^{(k)}$ is an aggregation function in the k-th layer. We instantiate $\text{AGG}^{(k)}$ using two

popular architectures: Graph Convolutional Network (GCN) [109] and Graph Attention Network (GAT) [110]. GCN defines the AGG function as follows:

$$\text{AGG}^{(k)}\left(\{h_v^{(k-1)}|v\in\widetilde{N(u)}\}\right)=\rho\left(\sum_{v\in\widetilde{N(u)}}\alpha_{uv}^{(k-1)}\mathbf{W}^{(k-1)}h_v^{(k-1)}\right),\tag{4.5}$$

where $\alpha_{uv}^{(k-1)}=1/\sqrt{|\widetilde{N(u)}||\widetilde{N(v)}|}$ is a normalization constant (same for all layers); ρ is a non-linear function (*e.g.*, ReLU), and $\mathbf{W}^{(k-1)}$ is the learnable weight matrix. If we interpret $\alpha_{uv}^{(k-1)}$ as the *importance* of node v's feature to node u, GCN calculates it using only the graph structure without leveraging the node features. GAT addresses this limitation by defining $\alpha_{uv}^{(k-1)}$ as follows:

$$\alpha_{uv}^{(k-1)}=\frac{\exp\left(\gamma\left(\mathbf{z}^{(k-1)}[\mathbf{W}^{(k-1)}h_u^{(k-1)}\|\mathbf{W}^{(k-1)}h_v^{(k-1)}]\right)\right)}{\sum_{v'\in\widetilde{N(u)}}\exp\left(\gamma\left(\mathbf{z}^{(k-1)}[\mathbf{W}^{(k-1)}h_u^{(k-1)}\|\mathbf{W}^{(k-1)}h_{v'}^{(k-1)}]\right)\right)},\tag{4.6}$$

where both $\mathbf{z}^{(\mathbf{k-1})}$ and $\mathbf{W}^{(k-1)}$ are learnable parameters; $\gamma(\cdot)$ is another non-linear function (*e.g.*, LeakyReLU), and "$\|$" represents the concatenation operation. Plugging the above $\alpha_{uv}^{(k-1)}$ into Eq. (4.5) we obtain the aggregation function in a *single-head* GAT. Finally, We execute M independent transformations of Eq. (4.5) and concatenate their output features to compose the final output embedding of node u. This defines the aggregation function in a *multi-head* GAT (with M heads) as follows:

$$\text{AGG}^{(k)}\left(\{h_v^{(k-1)}|v\in\widetilde{N(u)}\}\right)=\Big\|_{m=1}^{M}\rho\left(\sum_{v\in\widetilde{N(u)}}\alpha_{uv}^{(k-1)}\mathbf{W}_m^{(k-1)}h_v^{(k-1)}\right),\tag{4.7}$$

where $\mathbf{W}_m^{(k-1)}$ is the m-th weight matrix in the m-th attention head.

After obtaining each node's final representation $h_u^{(K)}$, we generate the ego network's representation h_G using a graph readout module as follows:

$$h_G=\text{READOUT}(\{h_u^{(K)}|u\in G\}),\tag{4.8}$$

where READOUT is a permutation invariant function [129] such as element-wise average, maximum, or summation.

Position-enhanced Graph Neural Networks. One key limitation of the above GNN model is that they fail to capture each node's position information relative to the query concept. For example, in Figure 4.3, the "*hospital room*" node in the left ego network is the anchor node itself while in the right ego network it is the child of the anchor node. Such

position information will influence how node feature propagates within the ego network and how the final graph embedding is aggregated.

An important innovation in TaxoExpan is the design of position-enhanced graph neural networks. The key idea is to learn a set of "position embeddings" and enrich each node feature with its corresponding position embedding. We denote node u's position as p_u and its position embedding at k-th layer as $\mathbf{p}_u^{(k)}$. We replace each node feature $h_u^{(k-1)}$ with its position-enhanced version $h_u^{(k-1)} \| \mathbf{p}_u^{(k-1)}$ in Eqs. (4.5-4.7) and adjust the dimensionality of $\mathbf{W}^{(k-1)}$ accordingly. Such position embeddings help us to learn better node representations from two aspects. First, we can capture more neighborhood information. Take $\mathbf{W}^{(k-1)} h_v^{(k-1)}$ in the right hand side of Eq. (4.5) as an example, we enhance it to the following:

$$\left[\mathbf{W}^{(k-1)} \| \mathbf{O}^{(k-1)}\right] \left[h_v^{(k-1)} \| \mathbf{p}_v^{(k-1)}\right] = \mathbf{W}^{(k-1)} h_v^{(k-1)} + \mathbf{O}^{(k-1)} \mathbf{p}_v^{(k-1)}, \tag{4.9}$$

where $\mathbf{O}^{(k-1)}$ is another weight matrix used to transform position embeddings. The above equation shows that a node's new representation is jointly determined by its neighborhoods' contents (*i.e.*, $h_v^{(k-1)}$) and relative positions in the ego network (*i.e.*, $\mathbf{p}_v^{(k-1)}$). Second, for GAT architecture, we can better model neighbor importance as the term $\alpha_{uv}^{(k-1)}$ in Eq. (4.5) currently depends on both $\mathbf{p}_u^{(k-1)}$ and $\mathbf{p}_v^{(k-1)}$.

Furthermore, we propose two schemes to inject position information in the graph readout module. The first one, called weighted mean readout (WMR), is defined as follows:

$$\text{READOUT}(\{h_u^{(K)} | u \in G\}) = \sum_{u \in G} \frac{\log(1 + \exp(\alpha_{p_u}))}{\sum_{u' \in G} \log(1 + \exp(\alpha_{p'_u}))} h_u^{(K)}, \tag{4.10}$$

where α_{p_u} is the parameter indicating the importance of position p_u. The second scheme is called concatenation readout (CR) which combines the average embeddings of nodes with the same position as follows:

$$\text{READOUT}(\{h_u^{(K)} | u \in G\}) = \Big\|_{p \in \mathcal{P}} \frac{\mathcal{I}(p_u = p) h_u^{(K)}}{\sum_{u' \in G} \mathcal{I}(p_{u'} = p)}, \tag{4.11}$$

where $\mathcal{P}$ is the set of all positions we are modeling and $\mathcal{I}(\cdot)$ is an indicator function which returns 1 if its internal statement is true and returns 0 otherwise.

4.4.5 Query-Anchor Matching: Matching Model

Based on the learned query concept representation $\mathbf{n}_i \in \mathbb{R}^{D_1}$ and anchor concept representation $\mathbf{a}_i \in \mathbb{R}^{D_2}$, we calculate their match score using a matching module $f(\cdot) : \mathbb{R}^{D_2} \times \mathbb{R}^{D_1} \to \mathbb{R}$. We study two architectures. The first one is a multi-layer perceptron with one hidden

layer, defined as follows:

$$f^{\text{MLP}}(\mathbf{a}_i, \mathbf{n}_i) = \sigma\left(\mathbf{W}_2\gamma(\mathbf{W}_1(\mathbf{a}_i \| \mathbf{n}_i) + \mathbf{B}_1) + \mathbf{B}_2\right), \tag{4.12}$$

where $\{\mathbf{W}_1, \mathbf{B}_1, \mathbf{W}_2, \mathbf{B}_2\}$ are parameters; $\sigma(\cdot)$ is the sigmoid function, and $\gamma(\cdot)$ is the LeakyReLU activation function. The second architecture is a log-bilinear model defined as follows:

$$f^{\text{LBM}}(\mathbf{a}_i, \mathbf{n}_i) = \exp\left(\mathbf{a}_i^T \mathbf{W} \mathbf{n}_i\right), \tag{4.13}$$

where $\mathbf{W}$ is a learnable interaction matrix. We choose these MLP and LBM as they are representative architectures in linear and bilinear interaction models, respectively.

4.4.6 Model Learning and Inference

The above sections discuss how to model query-anchor matching using a parameterized function $f(\cdot|\Theta)$. Here, we introduce how to learn those parameters using self-supervision from the existing taxonomy and establish the connection between the matching score with the conditional probability $\mathbf{P}(n_i|a_i)$. Finally, we discuss how to conduct model inference.

Self-supervision Generation. Figure 4.4 shows the generation process of self supervision data. Given one edge $\langle n_p, n_c\rangle$ in the existing taxonomy $\mathcal{T}^0 = (\mathcal{N}^0, \mathcal{E}^0)$, we first construct a positive $\langle$anchor, query$\rangle$ pair by using child node n_c as the "*query*" and parent node n_p as the "*anchor*". Then, we construct N negative pairs by fixing the query node n_c and randomly selecting N nodes $\{n_r^l|_{l=1}^N\} \subset \mathcal{N}^0$ that are neither parents nor descendants of n_c. These $N+1$ pairs (one positive and N negatives) collectively consist of one training instance $\mathbf{X} = \{\langle n_p, n_c\rangle, \langle n_r^1, n_c\rangle, \ldots, \langle n_r^N, n_c\rangle\}$. By repeating the above process for each edge in $\mathcal{T}^0$, we obtain the full self-supervision dataset $\mathbb{X} = \{\mathbf{X_1}, \ldots, \mathbf{X}_{|\mathcal{E}^0|}\}$. Notice that a node with C parents in $\mathcal{T}^0$ will derive C training instances in $\mathbb{X}$.

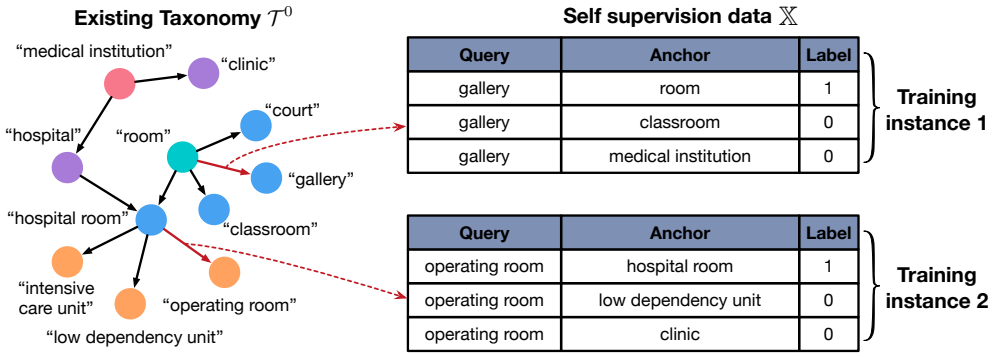


Figure 4.4: Self-supervision generation in TaxoExpan framework.

Model Training. We learn our model on $\mathbb{X}$ using the InfoNCE loss [111] as follows:

$$\mathcal{L}(\Theta) = -\frac{1}{|\mathbb{X}|}\sum_{\mathbf{X}_i \in \mathbb{X}}\left[\log \frac{f(n_p, n_c)}{\sum_{\langle n_j, n_c\rangle \in \mathbf{X}_i} f(n_j, n_c)}\right], \tag{4.14}$$

where the subscript $j \in [1, 2, \ldots, N+1]$. If $j = 1$, $\langle n_j, n_c\rangle$ is a positive pair, otherwise, $\langle n_j, n_c\rangle$ is a negative pair. In other words, $\mathbf{X}$ contains query n_c's one positive anchor (*i.e.*, its true parent n_p) sampled from the true distribution $\mathbf{P}(a_i|n_c)$ and N negative anchors $\{n_r^l|_{l=1}^N\}$ sampled from a uniform distribution $\mathbf{P}(a_i)$. If we merge these $N+1$ anchors into a small set and consider the task of selecting true anchor n_p's position j^* in $[1, 2, \ldots, N+1]$, we can view Eq. (4.14) as the cross entropy of position distribution $\hat{\mathbf{P}}$ from model prediction relative to the true distribution $\mathbf{P}^*$. Specifically, the model predicted position distribution $\hat{\mathbf{P}}_j = \frac{f(a_j, n_c)}{\sum_{k=1}^{N+1} f(a_k, n_c)}$ where one of $\{a_k|_{k=1}^{N+1}\}$ is the true anchor and all the others are negative anchors. Meanwhile, in the true position distribution:

$$\mathbf{P}_j^* = \frac{\mathbf{P}(a_j|n_c)\prod_{l\neq j}\mathbf{P}(a_l)}{\sum_{k=1}^{N+1}\left(\mathbf{P}(a_k|n_c)\prod_{l\neq k}\mathbf{P}(a_l)\right)} = \frac{\frac{\mathbf{P}(a_j|n_c)}{\mathbf{P}(a_j)}}{\sum_{k=1}^{N+1}\frac{\mathbf{P}(a_k|n_c)}{\mathbf{P}(a_k)}}. \tag{4.15}$$

From above, we can see that the optimal value for $f(a_j, n_c)$ is proportional to $\frac{\mathbf{P}(a_j|n_c)}{\mathbf{P}(a_j)}$. Therefore, optimizing the loss in Eq. (4.14) results in $f(a_i, n_i)$ estimating the following probability density (up to a multiplicative constant):

$$f(a_i, n_i) \propto \frac{\mathbf{P}(a_i|n_i)}{\mathbf{P}(a_i)}. \tag{4.16}$$

We establish the connection between matching score $f(a_i, n_i)$ with the probability $\mathbf{P}(n_i|a_i)$ in Eq. (4.3) as follows:

$$\mathbf{P}(n_i|a_i) = \frac{\mathbf{P}(a_i|n_i)}{\mathbf{P}(a_i)} \cdot \mathbf{P}(n_i) \propto f(a_i, n_i) \cdot \mathbf{P}(n_i). \tag{4.17}$$

We elaborate the implication of the above equation below and summarize our self-learning procedure in Algorithm 4.1.

Model Inference. At the inference stage, we are given a new query concept n_i and apply the learned model $f(\cdot|\Theta)$ to predict its parent node in the existing taxonomy $\mathcal{T}^0$. Mathematically, we aim to find the anchor position a_i that maximizes $\mathbf{P}(n_i|a_i)$, which is equivalent to maximizing $f(a_i, n_i)$ because of Eq. (4.17) and the fact that $P(n_i)$ is the same across all positions. Therefore, we rank all candidate positions a_i based on their matching scores with n_i and select the top ranked one as the predicted parent node of this query concept. Although we currently select only the top one as query's single parent, we can also choose top-k ones as query's parents, if needed.

Algorithm 4.1: Self-supervised learning of TaxoExpan

Input: A taxonomy $\mathcal{T}^0$; negative size N, batch size B; model $f(\cdot|\Theta)$.
Output: Learned model parameters Θ.

Randomly initialize Θ;
while $\mathcal{L}(\Theta)$ *in Eq. (4.14) not converge* **do**
    Enumerate edges in $\mathcal{T}^0$ and sample B edges without replacement;
    $\mathbb{X} = \{\}$ # current batch of training instances;
    for *each sampled edge* $\langle n_p, n_c \rangle$ **do**
        Generate N negative pairs $\{\langle n_r^l, n_c \rangle|_{l=1}^N\}$;
        $\mathbb{X} \leftarrow \mathbb{X} \cup \{\langle n_p, n_c \rangle, \langle n_r^1, n_c \rangle, \ldots, \langle n_r^N, n_c \rangle\}$;
    Update Θ based on $\mathbb{X}$.
Return Θ;

Summary. Given an existing taxonomy and a set of new concepts, our TaxoExpan framework first generates a set of self-supervision data and learns its internal model parameters using Algorithm 4.1. For each new concept, we run the inference procedure and find its best parent node in the existing taxonomy. Finally, we place these new concepts underneath their predicted parents one at a time, and output the expanded taxonomy.

Computational Complexity Analysis. At the training stage, our model uses $|\mathcal{E}^{(0)}|$ training instances every epoch and thus scales linearly to the number of edges in the existing taxonomy. At the inference stage, for each query concept, we calculate $|\mathcal{N}^{(0)}|$ matching scores, one for every existing node in $\mathcal{T}^0$. Although such $O(|\mathcal{N}^{(0)}|)$ cost per query is expensive, we can significantly reduce it using two strategies. First, most computation efforts of TaxoExpan are matrix multiplications and thus we use GPU for acceleration. Second, as the graph propagation and graph readout modules are query-independent (c.f. Figure 4.4), we pre-compute and cache all anchor representations. When a set of queries are given, we only run the matching module. In practice, it takes less than 30 seconds to calculate all matching scores between 2,450 queries with over 24,000 anchor positions on a single K80 GPU.

4.5 EXPERIMENTS

We conduct two sets of experiments to verify the effectiveness of TaxoExpan framework. Sections (4.5.1—4.5.2) present our results on the Field-of-Study (FoS) Taxonomy[1] in Microsoft Academia Graph (MAG) [24]. Sections (4.5.3—4.5.4) present the results on the SemEval 2016 Task 14 benchmark dataset[2] [106]. Table 4.1 lists the dataset statistics.

[1] `https://docs.microsoft.com/en-us/academic-services/graph/reference-data-schema`
[2] `http://alt.qcri.org/semeval2016/task14/`.

Table 4.1: Dataset statistics. $|\mathcal{N}|$ and $|\mathcal{E}|$ are the number of nodes and edges in the existing taxonomy. $|\mathcal{D}|$ indicates the taxonomy depth and $|\mathcal{C}|$ is the number of new concepts.

Dataset	$\|\mathcal{N}\|$	$\|\mathcal{E}\|$	$\|\mathcal{D}\|$	$\|\mathcal{C}\|$
MAG-CS	24,754	42,329	6	2,450
MAG-Full	355,808	638,674	6	37,804
SemEval	95,882	89,089	20	600

4.5.1 Experiment Settings on MAG Datasets

Datasets. We evaluate TaxoExpan on the public Field-of-Study (FoS) Taxonomy in Microsoft Academic Graph (MAG). This FoS taxonomy contains over 660 thousand scientific concepts and more than 700 thousand taxonomic relations. Although being constructed semi-automatically, this taxonomy is of high quality, as shown in the previous study [130]. Thus we treat each concept's original parent nodes as its correct anchor positions. We remove all concepts that have no relation in the original FoS taxonomy and then randomly mask 20% of leaf concepts (along with their relations) for validation and testing[3]. The remaining FoS taxonomy is then treated as the input existing taxonomy. We refer to this dataset as **MAG-Full**. Based on MAG-Full, we construct another dataset focusing on the computer science domain. Specifically, we first select a subgraph consisting of all descendants of "computer science" node and then mask 10% of leaf concepts in this subgraph for validation and another 10% of leaf nodes for testing. We name this dataset as **MAG-CS**.

Compared Methods. We compare the following methods:

- **Closest-Parent**: A rule-based method which first scores each candidate position in the existing taxonomy based on its cosine distance to the query concept between their initial embedding, and then ranks all positions using this score. The position with the smallest distance is chosen to be query concept's parent.
- **Closest-Neighbor**: Another rule-based method that scores each position based on its distance to the query concept plus the average distance between its children and the query.
- **dist-XGBoost**: A self-supervised boosting method that works directly on 39 manually-designed features generated using initial node embeddings without any embedding transformation. We input these features into XGBoost [131], a tree-based boosting model, to predict the matching score between a query concept and a candidate position.
- **ParentMLP**: A self-supervised method that first concatenates the query concept em-

[3]Here we mask only leaves because if we remove intermediate nodes, we have to remove their descendants from the candidate parent pool, which causes different masked nodes (as testing query concepts) having different candidate pools.

bedding with the candidate position embedding and then feeds them into a Multi-Layer Perceptron (MLP) for prediction.

- **DeepSetMLP**: Another self-supervised method that extends ParentMLP by adding information of candidate position's children nodes. Specifically, we first use DeepSet architecture [129] to generate the representation of the children node set and then concatenate it with query & candidate position representations before the final MLP module.
- **TaxoExpan**: Our proposed framework using position-enhanced GAT (PGAT) as graph propagation module and weighted mean readout (WMR) for graph readout. We learn this model using our proposed InfoNCE loss.

Evaluation Metrics. As our model returns a rank list of all candidate parents for each input query concept, we evaluate its performance using three ranking-based metrics.

- **Mean Rank (MR)** measures the average rank position of a query concept's true parent among all candidates. For queries with multiple parents, we first calculate the rank position of each individual parent and then take the average of all rank positions. Smaller MR value indicates better model performance.
- **Hit@k** is the number of query concepts whose parent is ranked in the top k positions, divided by the total number of queries.
- **Mean Reciprocal Rank (MRR)** calculates the reciprocal rank of a query concept's true parent. We follow [132] and use a scaled version of MRR in the below equation:

$$\text{MRR} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|parent(c)|} \sum_{i \in parent(c)} \frac{1}{\lceil R_{i,c}/10 \rceil}, \tag{4.18}$$

  where $parent(c)$ represents the parent node set of the query concept c, and $R_{i,c}$ is the rank position of query concept c's true parent i. We scale the original MRR by a factor 10 in order to amplify the performance gap between different methods.

Implementation Details. For a fair comparison, we use the same 250-dimension embeddings across all compared methods. We use Google's original word2vec implementation[4] for learning embeddings and employ gensim[5] to load trained embeddings for calculating term distances in Closest-Parent, Closest-Neighbor, and dist-XGBoost methods. For the other three methods, we implement them using PyTorch and DGL framework[6]. We tune hyper-parameters in all self-supervised methods on the masked validation set. For TaxoExpan, we use a two-layer position-enhanced GAT where the first layer has four attention heads (of size 250) and the second layer has one attention head (of size 500). For both layers, we

[4] `https://github.com/tmikolov/word2vec`
[5] `https://github.com/RaRe-Technologies/gensim`
[6] `https://github.com/dmlc/dgl`

Table 4.2: Overall results on MAG-CS and MAG-Full datasets. We run all methods three times and report the averaged result with the the best two models highlighted under each metric.

Method	MAG-CS				MAG-Full			
	MR	Hit@1	Hit@3	MRR	MR	Hit@1	Hit@3	MRR
Closest-Parent	1327.16	0.0531	0.0986	0.2691	14355.5	0.0360	0.0728	0.1897
Closest-Neighbor	382.07	0.1085	0.2000	0.3987	4160.8	0.0221	0.0419	0.1405
dist-XGBoost	136.86	0.1903	0.3483	0.6618	**426.70**	**0.1498**	**0.3046**	0.5621
ParentMLP	**114.79**	0.0729	0.2656	0.6454	457.14	0.098	0.1928	0.4950
DeepSetMLP	115.26	**0.1988**	**0.3581**	**0.6653**	444.83	0.1461	0.2971	**0.6392**
TaxoExpan	**80.33**	**0.2121**	**0.3823**	**0.6929**	**341.31**	**0.1523**	**0.3087**	**0.6453**

use 50-dimension position embeddings and apply dropout with rate 0.1 on the input feature vectors. We use Adam optimizer with initial learning rate 0.001 and ReduceLROnPlateau scheduler[7] with three patience epochs. We discuss the influence of these hyper-parameters in the next subsection.

4.5.2 Experiment Results on MAG Datasets

We present the experiment results in the following aspects.

Overall Performance. Table 4.2 presents the results of all compared methods. First, we find that Closest-Neighbor method clearly outperforms Closest-Parent method. Also, the DeepSetMLP method is much better than ParentMLP. This demonstrates the effectiveness of modeling local structure information. Second, we compare dist-XGBoost method with Closest-Neighbor and show that self-supervision indeed helps us to learn an effective way to combine various neighbor distance information. All four self-supervised methods outperform rule-based methods. Finally, our proposed TaxoExpan has the overall best performance across all the metrics and defeats the second best method by a large margin.

Ablation Analysis of Model Architectures. TaxoExpan contains three key components: a graph propagation module, a graph readout module, and a matching model. Here, we study how different choices of these components affect the performance of TaxoExpan. Table 4.3 lists the results and the first column contains the index of each model invariant.

First, we analyze graph propagation module by using simple average scheme for graph readout and MLP for matching. By comparing model 1 to model 3 and model 2 to model

[7] `https://pytorch.org/docs/stable/optim.html\#torch.optim.lr\_scheduler.ReduceLROnPlateau`

Table 4.3: Ablation analysis of model architectures on MAG-CS dataset. We assign an index to each model variant (shown in the first column). All models are run three times with their averaged scores reported.

Ind	Graph Propagate	Graph Readout	Matching	MR	Hit@1	Hit@3	MRR
1	GCN	Mean	MLP	167.82	0.1581	0.2964	0.6002
2	GAT	Mean	MLP	131.46	0.1584	0.3192	0.6409
3	PGCN	Mean	MLP	148.54	0.1809	0.3015	0.6255
4	**PGAT**	**Mean**	**MLP**	**100.80**	**0.1896**	**0.3304**	**0.6525**
5	PGCN	WMR	MLP	144.81	0.1798	0.3014	0.6309
6	PGCN	CR	MLP	135.89	0.1902	0.3118	0.6348
7	**PGAT**	**WMR**	**MLP**	**92.62**	**0.1945**	**0.3584**	**0.6619**
8	PGAT	CR	MLP	95.84	0.1897	0.3512	0.6596
9	PGCN	WMR	LBM	139.41	0.1829	0.3370	0.6642
10	PGCN	CR	LBM	130.12	0.1934	0.3462	0.6776
11	**PGAT**	**WMR**	**LBM**	**80.33**	**0.2121**	**0.3823**	**0.6929**
12	PGAT	CR	LBM	84.40	0.2089	0.3813	0.6894

4, we can see that graph attention architecture (GAT) is better than graph convolution architecture (GCN). Furthermore, the position-enhanced variants clearly outperform their non-position counterparts (model 3 versus model 1 and model 4 versus model 2). This illustrates the efficacy of the position embeddings in the graph propagation module.

Second, we study graph readout module by fixing the graph propagation module to be the best two variants among models 1-4. We can see both model 5 & 6 outperform model 3 and model 7 & 8 outperform model 4. This signifies that the position information also helps in the graph readout module. However, the best strategy of incorporating position information depends on the graph propagation module. The concatenation readout scheme works better for PGCN while the weighted mean readout is better for PGAT. One possible explanation is that the concatenation readout leads to more parameters in matching model and as PGAT itself has more parameters than PGCN, further introducing more parameters in PGAT may cause the model to be overfitted.

Finally, we examine the effectiveness of different matching models. We replace the MLP in models 5-8 with LBM to create model variants 9-12. We can clearly see that LBM works better than MLP. It could be that LBM better captures the interaction between the query representation and the final anchor representation.

Ablation Analysis of Training Schemes. In this subsection, we evaluate the effectiveness of our proposed training scheme. In this study, we first group a set of positive and negative $\langle query, anchor \rangle$ pairs into *one single* training instance and learn the model using InfoNCE loss (c.f. Eq. (4.14)). An alternative is to treat these pairs as different instances and train

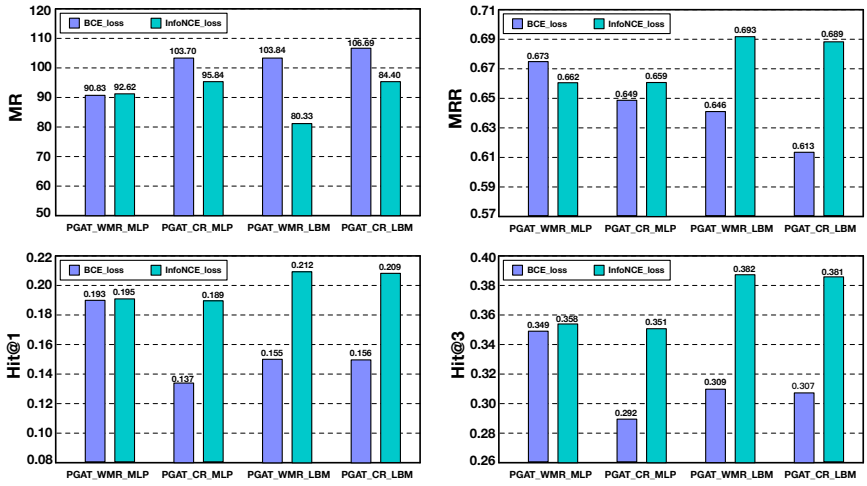


Figure 4.5: Ablation analysis of training schemes on MAG-CS dataset. We compare models trained using Binary Cross Entropy (BCE) loss with those trained using InfoNCE loss.

the model using standard binary cross entropy (BCE) loss. Under this training scheme, we formulate our problem as a binary classification task. We compare these two training schemes for the top 4 best models in Table 4.3 (*i.e.*, model 7, 8, 11, and 12). Results are shown in Figure 4.5. Our proposed training scheme with InfoNCE loss is overall much better, it beats the BCE loss scheme on 14 out of total 16 cases. One reason is that BCE loss is very sensitive to the noises in the generated self-supervision data while InfoNCE loss is more robust to such label noise. Furthermore, we find that LBM matching can benefit more from our training scheme with InfoNCE loss – with larger margin on all 8 cases, compared with the simple MLP matching.

Hyper-parameter Sensitivity Analysis. We analyze how some hyper-parameters in TaxoExpan affect the performance in Figure 4.6. First, we find that choosing an approximate position embedding dimension is important. The model performance increases as this dimensionality increases until it reaches about 50. When we further increase position embedding dimension, the model will overfit and the performance decreases. Second, we study the effect of negative sampling ratio N. As shown in Figure 4.6, the model performance first increases as N increases until it reaches about 30 and then becomes stable. Finally, we examine two hyper-parameters controlling the model complexity: the number of heads in PGAT and the final graph embedding dimension. We observe that the best model performance is reached when the number of attention heads falls in range 3 to 5 and the graph embedding dimension is set to 500. Too many attention heads or too large graph embedding dimension will lead to overfit and performance degradation.

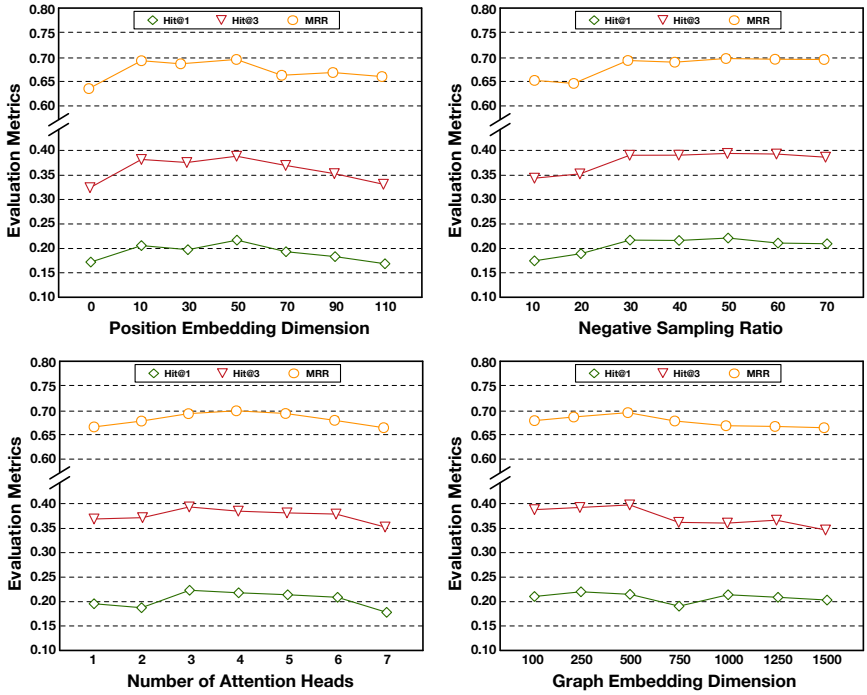


Figure 4.6: Hyper-parameter sensitivity analysis on MAG-CS dataset. We use PGAT for graph propagation, WMR for graph readout, and LBM for query-graph matching. Model is trained using InfoNCE loss.

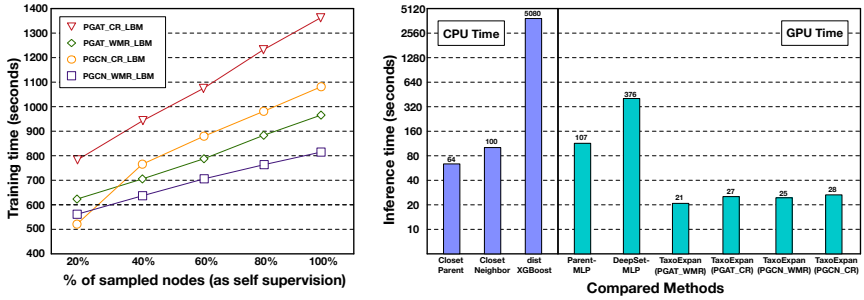


Figure 4.7: (Left) Training time of 20 epochs on GPU with respect to % of sampled nodes in the existing taxonomy. (Right) Inference time of all 2450 queries in MAG-CS dataset. Note here y-axis is in logarithm scale.

Efficiency and Scalability. We further analyze the scalability of TaxoExpan and its efficiency during model inference stage. Figure 4.7 (left) tests the model scalability by running on MAG-CS dataset sampled using different ratios. The training time (of 20 epochs) are measured on one single K80 GPU. TaxoExpan demonstrates a linear runtime trend, which validates our complexity analysis in Section 4.4.6. Second, Figure 4.7 (right) shows that TaxoExpan is very efficient during model inference stage. Using GPU, TaxoExpan takes less than 30 seconds to predict the anchor positions for all 2450 new query concepts.

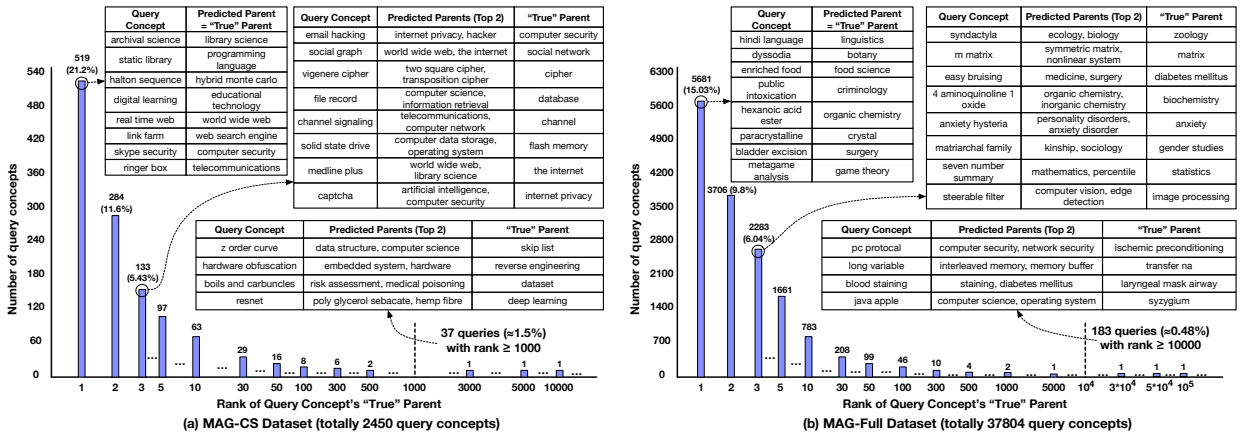


Figure 4.8: Example output of TaxoExpan on MAG-CS and MAG-Full datasets.

Case Studies. Figure 4.8 shows some outputs of TaxoExpan on both MAG-CS and MAG-Full datasets. On MAG-CS dataset, we can see that over 20% of queries have their true parents correctly ranked at the first position and less than 1.5% queries have their "true" parents ranked outside of top 1000 positions. Among these 1.5% significantly wrong queries, we find some of them actually have incorrect existing parents. For example, the concept "*boils and carbuncles*", which is a disease entity, is mistakenly put under parent node "*dataset*". We also observe two common mistake patterns. The first type of mistakes is caused by term ambiguity. For instance, the term "*java*" in concept "*java apple*" refers to an island in Indonesia where fruit apple is produced, rather than a programming language used in Apple company. The second type of mistakes results from term granularity. For example, TaxoExpan outputs the two most likely parent nodes of concept "*captcha*" are "*artificial intelligence*" and "*computer security*". Although these two concepts are certainly relevant to "*captcha*", they are too general compared to its true parent node "*internet privacy*". Finally, we observe that TaxoExpan can return very sensible anchor positions of query concepts, even though they are not exactly the current "true" parents. For example, the concept "*medline plus*" refers to a large online medical library and thus is related to both "*world wide web*" and "*library science*". Also, the concept "*email hacking*" is clearly relevant to both "*internet privacy*" and "*hacker*".

TaxoExpan for Taxonomy Self-Cleaning. From the above case studies, we find another interesting application of TaxoExpan is to use it for cleaning the existing taxonomy. Specifically, we partition all leaf nodes of the existing taxonomy into 5 groups and randomly mask one group of nodes. Then, we train a TaxoExpan model on the remaining nodes and predict on the masked leaf nodes. Next, we select those entities whose true parents appear at the bottom of the rank lists returned by TaxoExpan (*i.e.*, the long-tail part of two histograms

in Figure 4.8). The parents of those selected entities are highly questionable and calls for further manual inspections. In fact, we invite three human annotators to inspect those selected entities on the MAG-CS taxonomy and find that about 30% of these entities have existing parent nodes which are less appropriate than the parents inferred by TaxoExpan. In the future, we plan to study how to equip human taxonomists with our TaxoExpan-based models to enable them identify potentially flawed relations in the existing taxonomies faster.

4.5.3 Experiment Settings on SemEval Dataset

Datasets. Besides using the MAG datasets, we also evaluate TaxoExpan using SemEval Task 14 Benchmark dataset[8] [106] which includes WordNet 3.0 as the existing taxonomy and additional 1,000 domain-specific concepts with manual labels, split into 400 training concepts and 600 testing concepts. Each concept is either a verb or a noun and has a textual definition of a few sentences. The original task goal is to enrich the taxonomy by performing two actions for each new concept: (1) *attach*, where a new concept is treated as a new synset and is attached as a hyponym of one existing synset in WordNet, and (2) *merge*, where a new concept is merged into an existing synset. However, previous state-of-the-art methods [106, 107, 118], including the winning solution, are only performing the *attach* operation. In this work, we also follow this convention and attach each new concept to the top-ranked synset in the WordNet. Finally, we obtain the initial feature vectors (for both new concepts and existing words in the WordNet) using pre-trained subword-aware fasttext embeddings[9]. For each concept, we generate its definition embedding and name embedding by averaging the embedding of each token in its textual definition and name string, correspondingly. Then, we sum the definition and name embeddings of a concept and use them as the initial embeddings for the TaxoExpan model.

Compared Methods. We compare TaxoExpan with the following methods:

- **FWFS** [106]: The original baseline in Task 14. Given a concept c with its definition d_c, this method picks the first word w in d_c that has the same part of speech as c and treats this word as the parent node of c.
- **MSejrKU** [107]: The winning solution of Task 14. This method leverages distributional and syntactic features to train a SVM classifier which is then used to predict the goodness of fit for a new concept with an existing synset in WordNet.
- **ETF** [118]: The current state-of-the-art method that learns a LambdaMART model with 15 manually designed features, including topological features from the taxonomy's graph

[8] `http://alt.qcri.org/semeval2016/task14/`.

[9] We use the wiki-news-300d-1M-subword.vec.zip version on fastText official website.

structure and semantic features from corpus and Bing search results.

- **ETF-FWFS** [118]: The ensemble model of FWFS and ETF, which adds the FWFS property as a binary feature into the LambdaMART model in ETF.
- **dist-XGBoost**: The same tree boosting model described in the previous subsection.
- **TaxoExpan**: Our proposed taxonomy expansion framework.
- **TaxoExpan-FWFS**: Similar to ETF-FWFS, this is the ensemble model of FWFS and TaxoExpan. We treat the FWFS heuristic as a binary feature and add it into the final matching module.

For all previous methods, we directly report their best performance in the literature. For the remaining methods, we tune them following the same procedure as described before.

Evaluation Metrics. We use the three official metrics defined in original SemEval Task 14 for evaluation:

- **Accuracy (Wu&P)** is the semantic similarity between a predicted parent node x_p and the true parent x_t, calculated as $\text{Wu\&P}(x_p, x_t) = \frac{2 \cdot depth_{LCA(x_p,x_t)}}{depth_{x_p} + depth_{x_t}}$, where $depth_x$ is the depth of node x is the WordNet taxonomy and $LCA(x_p, x_t)$ represents the Least Common Ancestor of x_p and x_t.
- **Recall** is the percentage of concepts for which an attached parent is predicted[10].
- **F1** is the harmonic mean of Wu&P accuracy and recall.

4.5.4 Experiment Results on SemEval Datasets

Table 4.4: Model performance on SemEval dataset. TaxoExpan versus all previous state-of-the-art methods. We report the best performance of all existing methods in the literature.

Method	Wu&P	Recall	F1
MSejrKU [107]	0.523	0.973	0.680
FWFS [106]	0.514	1.000	0.679
ETF [118]	0.473	1.000	0.642
ETF-FWFS [118]	0.562	1.000	0.720
dist-XGBoost	0.528	1.000	0.691
TaxoExpan	0.543	1.000	0.704
TaxoExpan-FWFS	0.566	1.000	0.723

Table 4.4 shows the experiment results on SemEval dataset. We can see that both dist-

[10]This metric is used because the original task allows a model to decline to place new concepts in order to avoid making placements with low confidence.

XGBoost and TaxoExpan methods can outperform the previous winning system of this task (*i.e.*, MSejrKU) and the baseline ETF. In addition, we can see the FWFS heuristic is indeed very powerful for this dataset and incorporating it as a strong feature can significantly boost the performance. However, this feature requires human-labeled definition sentences and thus can not be easily generalized to taxonomies other than WordNet. Finally, we show that TaxoExpan-FWFS can achieve the new state-of-the-art performance on this dataset.

4.6 SUMMARY

This chapter studies the problem of concept taxonomy expansion when no human labeled supervision data are given. We propose a novel TaxoExpan framework which generates self-supervision data from the existing taxonomy and learns a position-enhanced GNN model for expansion. To make the best use of self-supervision data, we design a noise-robust objective for effective model training. Extensive experiments demonstrate the effectiveness and efficiency of TaxoExpan on three taxonomies from different domains.

Interesting future work includes modeling inter-dependency among new concepts, leveraging current method to cleaning the input existing taxonomy, and incorporating feedbacks from downstream applications (*e.g.*, search & recommendation) to generate more diverse supervision signals for expanding the taxonomy.

CHAPTER 5: TAXONOMY APPLICATION

5.1 OVERVIEW AND MOTIVATIONS

With taxonomy constructed and enriched on a domain-specific document collection, we can leverage it to enhance lots of downstream knowledge-centric applications. For example, we can use taxonomies to organize scientific literatures and support semantic literature retrieval. Similarly, we can categorize products based on a product category taxonomy and build a better search and recommender system. Within all those applications, we assume the text unit (either an entire document or an in-context text span) has already been associated with a set of classes in the corresponding taxonomy. Such assumption, however, may not hold in many real-world applications for which the taxonomy powered methods can not be applied. Therefore, to fully exploit the power of taxonomy, we need to study the hierarchical multi-label text classification (HMTC) problem which aims to assign each text document to a set of relevant classes from a class taxonomy.

Most existing methods address HMTC in a supervised fashion — they first ask humans to provide many labeled documents and then train a text classifier for prediction. Many classifiers have been developed with different deep learning architectures such as CNN [133], RNN [134], Attention Network [135], and achieved decent performance when trained on massive human-labeled documents. Despite such a success, people find that applying these methods to many real-world scenarios remains challenging as the human labeling process is often too time-consuming and expensive.

Recently, more studies have been developed to address text classification using smaller amount of labeled data. First, several semi-supervised methods [136, 137] propose to use abundant unlabeled documents to assist model training on labeled dataset. Although mitigating the human annotation burden, these methods still require a labeled dataset that covers all classes, which could be too expensive to obtain when we have a large number of classes in HMTC. Second, some weakly-supervised models exploit class indicative keywords [8, 138, 139] or class surface names [140, 141] to derive pseudo-labeled data for model training. Nevertheless, these models all assume each document has only one class and all class surface names (or class indicative keywords) must appear in the corpus, which are too restrictive for HMTC.

In this chapter, we study the problem of *weakly-supervised* hierarchical multi-label text classification where only class surface names, a class taxonomy, and an unlabeled corpus are available for model training. This setting is closer to how humans resolve the HMTC problem

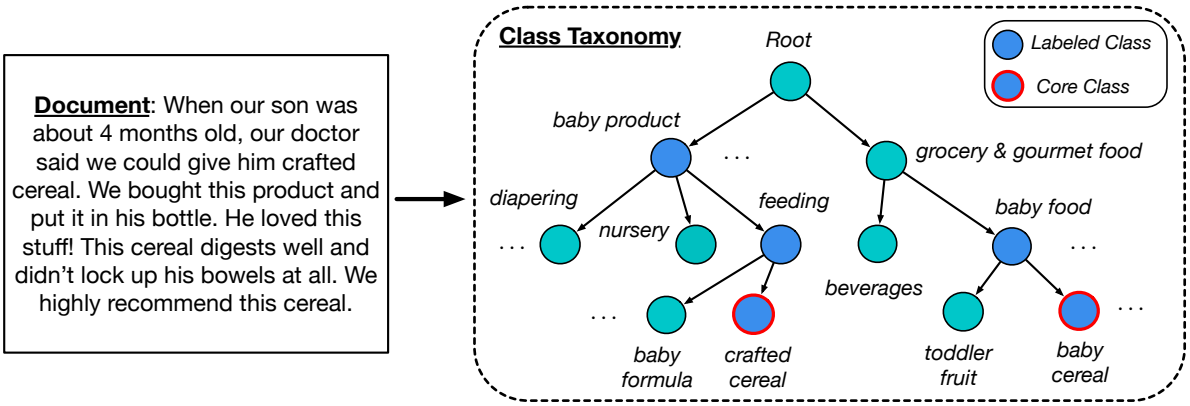


Figure 5.1: An exemplar document tagged with five classes. Here, if we are able to pinpoint this document's most essential classes, *crafted cereal* and *baby cereal*, as core classes, we can check their ancestor classes in the taxonomy and recover all the true classes.

— we perform classification by understanding each class from its surface name rather than learning from labeled documents. We observe that when asked to assign multiple classes to a document, humans will first pinpoint most essential "core classes" and then check whether their ancestor classes in the taxonomy should also be tagged. Taking the document in Figure 5.1 as an example, humans can quickly identify this review text is clearly about "*baby cereal*" and "*crafted cereal*", which are the core classes. After assigning these two most essential classes to the document, people continue to check the core classes' ancestor classes and find "*feeding*" as well as "*baby food*" should be tagged.

Motivated by the above human labeling process, we propose TaxoClass, a *weakly-supervised* HMTC framework including four major steps. First, we calculate the document-class similarity using a pre-trained textual entailment model [142]. Second, we identify each document's core classes by (1) selecting candidate core classes that are most similar to the document at each level in a top-down fashion, and (2) choosing ⟨document, candidate core class⟩ pairs that are salient across the whole unlabeled corpus. Third, we derive training data from document core classes and use them to train a text classifier. This classifier includes a document encoder based on pre-trained BERT [55], a class encoder capturing class taxonomy structure, and a text matching network computing the probability of a document being tagged with each class. Finally, we generalize this text classifier using multi-label self-training on all unlabeled documents.

To summarize, our major contributions are as follows:

- We propose a weakly-supervised framework TaxoClass that only requires class surface names to perform hierarchical multi-label text classification. To the best of our knowledge,

TaxoClass is the first weakly-supervised HMTC method.

- We develop an unsupervised method to identify document core classes based on which a text classifier can be learned.
- We conduct extensive experiments on two real-world datasets to verify the effectiveness of TaxoClass.

We organize the rest of this chapter as follows. Section 5.2 discusses the related work. Section 5.3 formalizes our problem. Section 5.4 presents our TaxoClass framework in and we conduct experiments in Section 5.5. Finally, we conclude this chapter in Section 5.6.

5.2 RELATED WORK

There are three major lines of work related to our study.

Weakly-supervised Text Classification. There exist some previous studies that leverage a few labeled documents or class-indicative keywords as weak supervision signals for text classification. A pioneering method is dataless classification [143, 144] which embeds documents and classes into the same semantic space of Wikipedia concepts and performs classification using the embedding similarity. After that, researchers extend this idea by mining concepts directly from the corpus rather than using the external Wikipedia [145, 146] . Along another line, Chen *et al.* and Li *et al.* propose to apply a seed-guided topic model to infer class-specific topics from class-indicative keywords and to predict document classes from posterior class-topic assignments. Compared with these methods, our TaxoClass framework neither restricts document and class embeddings to live in the same semantic space nor imposes strong statistical assumptions.

Recently, neural models are applied to weakly-supervised text classification. Researchers propose a pretrain-and-refine paradigm which first generates pseudo documents to pretrain a neural classifier and then refine this classifier via self-training [8, 9] . More studies [139, 140, 141] improve the above methods by introducing contextualized weak supervision and using a pre-trained language model to obtain better text representations. While achieving inspiring performance, these methods all assume each document has only one class and all class names (or class-indicative keywords) must appear in the corpus for pseudo training data generation. In this work, we relax these assumptions and develop a new method for weakly-supervised hierarchical multi-label text classification task.

Zero-shot Text Classification. Zero-shot text classification learns a text classifier based on training documents belonging to *seen* classes and applies the learned classifier to predict testing documents belonging to *unseen* classes. Nam *et al.* [147] jointly embed documents and

classes into a shared semantic space where knowledge from seen classes can be transferred to unseen classes. Such an idea is further developed in [148, 149, 142] where external resources (*e.g.*, knowledge graphs, natural language explanations of unseen classes, and open domain data) are introduced to help learn a better shared semantic space. Comparing with these methods, our TaxoClass framework does not require labeled data for a set of seen classes.

Hierarchical Text Classification. Hierarchical text classification leverages a class hierarchy to improve the standard text classification performance. Typical methods can be divided into two categories: (1) *local approaches* which learn a text classifier per class [150], per parent class [151], or per level [152], and (2) *global approaches* which incorporate taxonomy structure information into one single classifier through recursive regularization [153] or graph neural network (GNN) based encoder [154, 135, 155]. Our TaxoClass framework adopts the global approach and uses a GNN-based encoder to obtain each class's representation.

5.3 PROBLEM FORMULATION

In this section, we introduce the notations and present our task definition.

Notations. A *corpus* $\mathcal{D} = \{D_1, \ldots, D_N\}$ is a text collection where each document $D_i \in \mathcal{D}$ is a sequence of words. A *class taxonomy* $\mathcal{T} = (\mathcal{C}, \mathcal{R})$ is a directed acyclic graph where each node represents a class c_j and each directed edge $\langle c_m, c_n \rangle \in \mathcal{R}$ indicates that parent class c_m is more general than the child class c_n. In this work, we assume each class c_j has a surface name s_j (either a word or a phrase) that serves as the weak supervision signal.

Task Definition. Given an unlabeled corpus $\mathcal{D}$, a class hierarchy $\mathcal{T} = (\mathcal{C}, \mathcal{R})$, and class surface names $\mathcal{S} = \{s_j\}_{j=1}^{|\mathcal{C}|}$, our task is to learn a text classifier $f(\cdot)$ that maps a new document D_{new} to its target $\mathbf{y} = [y_1, \ldots, y_{|\mathcal{C}|}] \in \mathcal{Y} = \{0, 1\}^{|\mathcal{C}|}$ where y_j equals to 1 if this document is categorized with class c_j and 0 otherwise.

Discussion. When the number of classes $|\mathcal{C}|$ is large (as it is in many HMTC applications), we can no longer assume all class surface names in $\mathcal{S}$ will explicitly appear in the given corpus $\mathcal{D}$ as done in most previous studies [9, 146, 141]. This is because many class names are actually summarizing phrases provided by humans (*e.g.*, "*grocery & gourmet food*" in Figure 5.1). As a result, we need to design a method that works under such a scenario.

5.4 THE TAXOCLASS FRAMEWORK

Our TaxoClass framework consists of four major steps: (1) document-class similarity calculation, (2) document core class mining, (3) core class guided classifier training, and (4)

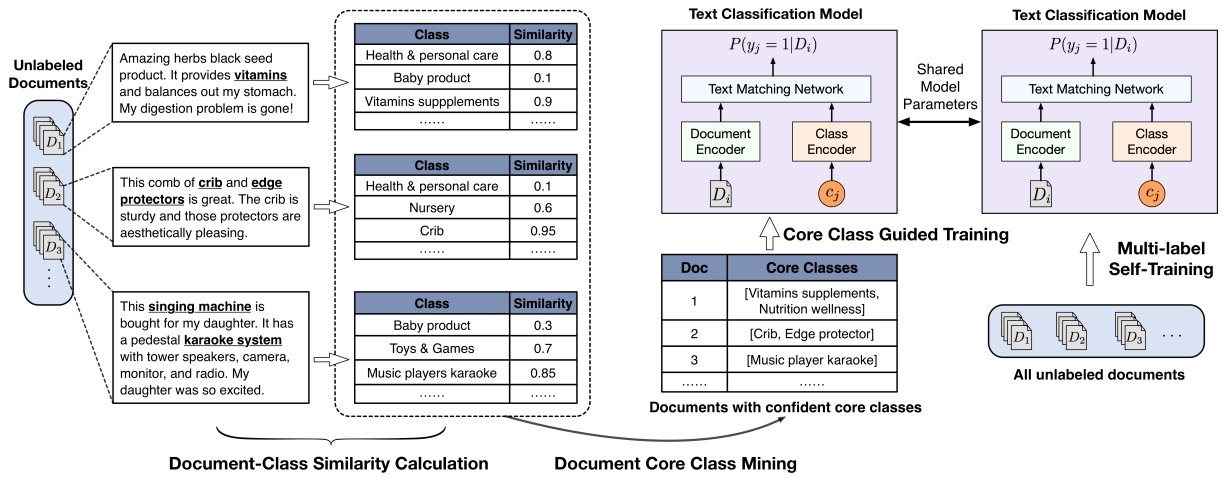


Figure 5.2: Our TaxoClass framework overview. We first calculate document-class similarities using a textual entailment model (Section 5.4.1). Then, we identify document core classes (Section 5.4.2) and train a taxonomy-enhanced text classifier (Section 5.4.3). Finally, we generalize the classifier via multi-label self-training (Section 5.4.4). The "shared model parameters" indicates that we do self-training on the same model learned using our identified core classes.

multi-label self-training. Figure 5.2 shows our framework overview and below sections discuss each step in more details.

5.4.1 Document-Class Similarity Calculation

We take a textual entailment approach [142] to calculate the semantic similarity between each ⟨document, class⟩ pair. This approach imitates how humans determine whether a document is similar to a class or not — we read this document, create a hypothesis by filling the class name into a template (*e.g.*, "*this document is about* __"), and ask ourselves to what extent this hypothesis is correct, given the context document.

In this work, we adopt a pre-trained textual entailment model that inputs a document D_i as the "premise", a template filled with a class name s_j as the "hypothesis", and outputs a probability of how likely this premise can entail the hypothesis. We treat this probability $\mathbf{P}(D_i \rightarrow c_j)$ as the document-class similarity $sim(D_i, c_j)$. More specifically, we use `Roberta-Large-MNLI`[1] as our textual entailment model which utilizes the pre-trained `Roberta-Large` as its backbone and is fine-tuned on the MNLI dataset.

[1] `https://huggingface.co/roberta-large-mnli`

5.4.2 Document Core Class Mining

When asked to tag a document with a set of classes from a class taxonomy, humans will first pinpoint a few classes that are most essential to this document. We refer to those most essential classes as the "core classes" and identify them in below two steps.

Core Class Candidate Selection. We observe that on average each document is tagged with a small set of classes from the entire class taxonomy. Therefore, we first reduce the search space of core classes using a top-down approach (c.f. Figure 5.3). Given a document D, we start with the "*Root*" class at level $l = 0$, find its two children classes that have the highest similarity with D, and add them into a queue. Then, for each class at level l in the queue, we select $l + 2$ classes from its children classes that are most similar to D. After all level l classes are processed, we aggregate all selected children classes and choose $(l + 1)^2$ classes (at level $l + 1$) with the highest path score (ps) defined below:

$$ps(Root) = 1, \quad ps(c_j) = \max_{c_k \in Par(c_j)} \{ps(c_k) \cdot sim(c_j, D)\}, \tag{5.1}$$

where $Par(c_j)$ is class c_j's parent class set. All chosen classes (at level $l + 1$) will be pushed into the queue and we stop this process when no class in the queue has further children. Finally, all classes that have entered the queue, except for the "*Root*" class, consist of the core class candidate set, denoted as $\mathbb{C}_i^{cand}$ for document D_i.

Confident Core Class Identification. For each document, we identify its core classes from the above selected candidate set based on two observations. First, a document usually has higher similarity with its core class c than with the parent and sibling classes of c. Take the document D_2 in Figure 5.2 as an example, the similarity between D_2 and its core class "*crib*" is 0.95, much higher than the similarity between D_2 and core class's parent class "*nursery*" (0.6) as well as core class's sibling classes. Based on this observation, we define the "confidence score" of a candidate core class c for a document D as below:

$$conf(D, c) = sim(D, c) - \max_{c' \in Par(c) \cup Sib(c)} \{sim(D, c')\}, \tag{5.2}$$

where $Sib(c)$ represents the sibling class set of c.

Our second observation is that the similarity between a document D and its core class c is salient from a *corpus-wise* perspective. Namely, if a class c is a document D's core class, the confidence score $conf(D, c)$ is higher than the median confidence score[2] between class c and all documents tagged with c (denoted as $\mathcal{D}(c)$). Formally, we have:

$$conf(D, c) \geq median\{conf(D', c) | D' \in \mathcal{D}(c)\}. \tag{5.3}$$

[2]We have also tried using "average" but empirically found that using "median" is better and more robust to outliers.

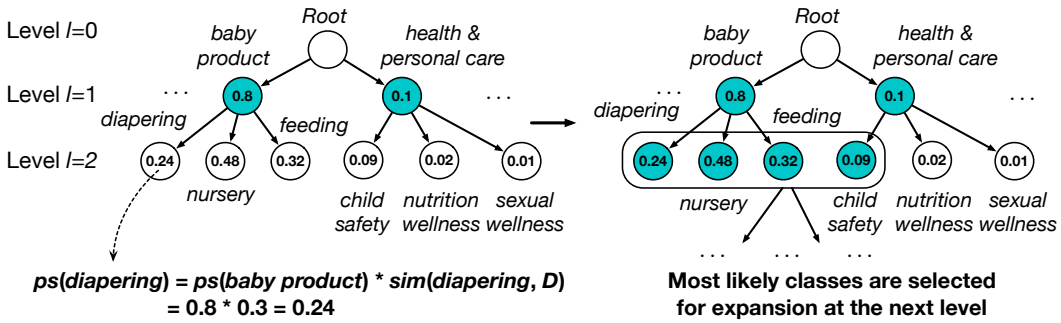


Figure 5.3: Top-down core class candidate selection.

According to this observation, we check each class in document D_i's candidate core set $\mathbb{C}_i^{cand}$ and add classes that satisfy the above criteria into the final core class set $\mathbb{C}_i$. Note here this core class set $\mathbb{C}_i$ could be empty when document D_i does not have any confident core class.

5.4.3 Core Class Guided Classifier Training

Based on identified document core classes, we train one classifier for hierarchical multi-label text classification. Below we first introduce our classifier architecture and then present our training method.

Text Classifier Architecture. We design our classifier to have a dual-encoder architecture: one document encoder maps document D_i to its representation $\mathbf{D}_i$, one class encoder learns class c_j's representation $\mathbf{c}_j$, and one matching network returns the probability of document D_i being tagged with class c_j.

We instantiate our document encoder $g_{\text{doc}}(\cdot)$ to be a pre-trained `BERT-base-uncased` model [55] and follow previous work [140] to use the `[CLS]` token representation as the document representation. For class encoder $g_{\text{class}}(\cdot)$, we follow [5] and use a graph neural network (GNN) [109] to model the class taxonomy structure. This taxonomy-enhanced class encoder can capture both the textual information from class surface names and structural information from the class taxonomy.

Given a class c_j, we first obtain its ego network that includes its parent and children classes in the class taxonomy, as shown in Figure 5.4. Then, we input this ego network to a GNN that propagates node features over the network structure. The node features are initialized with the pre-trained word embeddings of class surface names[3]. The propagation mechanism updates the feature of a node u by iteratively aggregating representations of its

[3]For multi-gram class names, we use their averaged word embeddings.

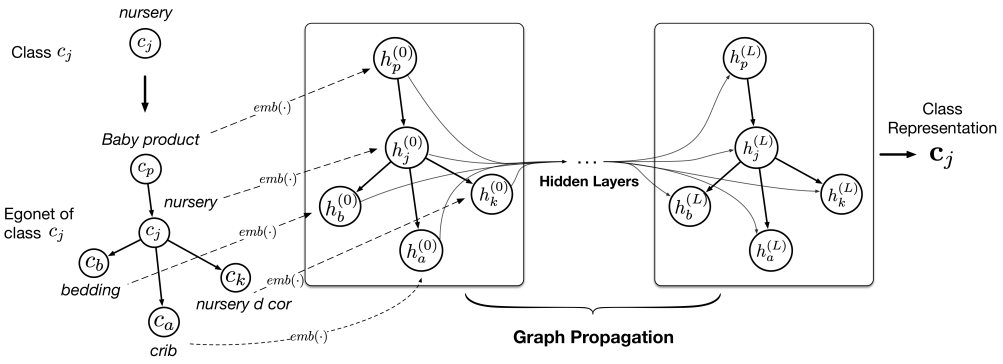


Figure 5.4: Taxonomy-enhanced class encoder in TaxoClass.

neighbors and itself. Formally, we define a GNN with L-layers as follows:

$$h_u^{(l)} = \text{ReLU}\left(\sum_{v \in N(u)} \alpha_{uv}^{(l-1)} \mathbf{W}^{(l-1)} h_v^{(l-1)}\right), \tag{5.4}$$

where $l \in \{1, \ldots, L\}$, $N(u)$ includes node u's neighbors and itself, $\alpha_{uv}^{(l-1)} = \frac{1}{\sqrt{|N(u)||N(v)|}}$ is a normalization constant (same for all layers), and $\mathbf{W}^{(l-1)}$ are learnable parameters.

After obtaining individual node features, we combine them into a vector representing the whole ego network G as follows:

$$h_G = \frac{1}{|G|} \sum_{u \in G} h_u^{(L)}. \tag{5.5}$$

As this ego network is centered on class c_j and encodes its both textual and structural information, we treat this final graph representation as the class representation $\mathbf{c}_j$.

Based on the document representation $\mathbf{D}_i$ and the class representation $\mathbf{c}_j$, we use a log-bilinear text matching model to compute the probability of document D_i being tagged with class c_j as follows:

$$p_{ij} = \mathbf{P}(y_j = 1 | D_i) = \sigma(\exp(\mathbf{c}_j^T \mathbf{B} \mathbf{D}_i)), \tag{5.6}$$

where $\sigma(\cdot)$ is the sigmoid function and $\mathbf{B}$ is a learnable interaction matrix.

Text Classifier Training. We use our discovered document core classes to train a text classifier. One strategy is to treat each document's core classes as positive classes and all the remaining classes as negative classes. However, this strategy has a high false negative rate because some non-core classes could still be relevant to the document (c.f. Figure 5.1).

We observe a document's multiple labeled classes usually have some ancestor-descendent relations in the class hierarchy $\mathcal{T} = (\mathcal{C}, \mathcal{R})$. This implies that given a document's core class, its parent class and some of its children classes are also likely to be tagged with this document. Therefore, we introduce all core classes' parent classes into the positive class set

and exclude their children classes from the negative class set. Formally, given a document D_i with its core class set $\mathbb{C}_i$, we define its positive and negative class set as follows:

$$\mathbb{C}_i^{pos} = \left(\bigcup_{c_j \in \mathbb{C}_i} Par(c_j) \right) \cup \mathbb{C}_i, \quad \mathbb{C}_i^{neg} = \mathcal{C} - \mathbb{C}_i^{pos} - \bigcup_{c_j \in \mathbb{C}_i} Chd(c_j), \tag{5.7}$$

where $Chd(c_j)$ is class c_j's children class set. Finally, we train our classification model using the below binary cross entropy (BCE) loss:

$$\mathcal{L} = -\sum_{\substack{i=1 \\ \mathbb{C}_i \neq \emptyset}}^{|\mathcal{D}|} \Big(\sum_{c_j \in \mathbb{C}_i^{pos}} \log p_{ij} + \sum_{c_j \in \mathbb{C}_i^{neg}} \log(1 - p_{ij}) \Big), \tag{5.8}$$

where "$\emptyset$" indicates an empty set and we exclude the documents without any confident core class from the loss calculation.

5.4.4 Multi-label Self-Training

After training the text classifier based on document core classes, we propose to further refine the model via self-training on the entire unlabeled corpus $\mathcal{D}$ for better generalization. The idea of self-training (ST) [156] is to iteratively use the model's current prediction P to compute a target distribution Q which guides the model for refinement. In general, the ST objective is expressed with the KL divergence loss as below:

$$\mathcal{L}_{ST} = \mathrm{KL}(Q||P) = \sum_{i=1}^{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{C}|} q_{ij} \log \frac{q_{ij}}{p_{ij}}. \tag{5.9}$$

The target distribution Q is constructed by enhancing high-confidence predictions while down-weighting low-confidence ones:

$$q_{ij} = \frac{p_{ij}^2 / (\sum_i p_{ij})}{p_{ij}^2 / (\sum_i p_{ij}) + (1 - p_{ij})^2 / (\sum_i (1 - p_{ij}))}. \tag{5.10}$$

Different from the previous studies [8], our target distribution Q can be applied to multi-label classification problem as it normalizes the current predictions P for each individual class. Intuitively, this equation can enhance high-confidence predictions while down-weighting low-confidence predictions. This is because if example i is more confidently labeled with class j than other examples, we will have a large p_{ij} that dominates the $\sum_i p_{ij}$ term. Consequently, Eq. (5.10) computes a large q_{ij}, which pushes the model to predict class j for example i.

Algorithm 5.1: TaxoClass Framework.

Input: An unlabeled corpus $\mathcal{D}$, a class taxonomy $\mathcal{T}$ with class names $\mathcal{S}$, an entailment model $\mathcal{M}$, total number of batches B.

Output: A trained classifier $f(\cdot)$.

Use model $\mathcal{M}$ to compute document-class similarity (c.f. Sect. 5.4.1);
Obtain document core classes $\{(D_i, \mathbb{C}_i) \text{ — } D_i \in \mathcal{D}\}$ (c.f. Sect. 5.4.2);
Train classifier $f(\cdot)$ with Eq. (5.8);
for *i from 1 to B* **do**
    if *i mod 25 = 0* **then**
        Update Q with Eq. (5.10);
    Train classifier $f(\cdot)$ with Eq. (5.9);
Return $f(\cdot)$;

In practice, instead of updating the target distribution Q for every training example, we update it every 25 batches[4] and train the model with Eq. (5.9), which makes the self-training process more efficient and robust. We summarize our TaxoClass framework in Algorithm 5.1.

5.5 EXPERIMENTS

5.5.1 Datasets

We use two public datasets from different domains to evaluate our method: (1) **Amazon-531** [157] contains 49,145 product reviews and a three-level class taxonomy consisting of 531 classes; and (2) **DBPedia-298** [158] includes 245,832 Wikipedia articles and a three-level class taxonomy with 298 classes. Documents in both datasets are lower-cased and truncated to has maximum 500 tokens. We list the data statistics in Table 5.1.

Table 5.1: Dataset statistics. Supervised methods are trained on the entire training set. Weakly-supervised methods are trained by treating the training set as unlabeled data. All methods are evaluated on the test set.

Dataset	# Train	# Test	# Classes
Amazon-531	29,487	19,685	531
DBPedia-298	196,665	49,167	298

[4]This hyper-parameter controls the update frequency. Empirically, we find our model is insensitive to this hyper-parameter (in the typical value range of 10-100).

5.5.2 Compared Methods

To the best of our knowledge, we are the first to study *weakly-supervised* HMTC problem and there is no directly comparable baseline under the exact same setting as ours. Therefore, we choose a wide range of representative methods that are most related to TaxoClass and adapt them to our problem setting, described as follows.

- **Hier-doc2vec** [159][5]: This *weakly-supervised* method first embeds documents and classes into a shared semantic space, and then recursively selects the class of the highest embedding similarity with the document in a top-down fashion. We set the embedding dimensionality to be 100 and use the default value for all other hyper-parameters.[6]
- **WeSHClass** [9][7]: Another *weakly-supervised* method that generates pseudo documents to pre-train a text classifier and bootstraps the pre-trained classifier on unlabeled documents with self-training. The class surface names are treated as the "class-related keywords" in this method. For the pseudo document generation step, we use its internal LSTM language model. We treat all classes in its returned class path as the output classes.
- **SS-PCEM** [160][8]: This *semi-supervised* method uses a generative model to generate documents based on a class path sampled from the class taxonomy. Both labeled and unlabeled documents are used to fit this generative model via the EM algorithm. Finally, it uses the posterior probability of a test document to predict its labeled classes. Among different base classifiers, we choose their author reported best variant PCEM in this study. We use 30% of labeled training documents for this method.
- **Hier-0Shot-TC** [142][9]: This *zero-shot* method uses a pre-trained textual entailment model to predict to what extent a document (as the premise text) can entail a template filled with the class name (as the hypothesis text). Similar to **Hier-doc2vec**, we select the class with the highest entailment score at each level in a top-down recursive fashion. For fair comparison, we change its internal `BERT-base-uncased` model to `RoBERTa-large-mnli` model as is used in our method.
- **TaxoClass**[10]: Our proposed *weakly-supervised* framework that identifies document core classes, leverages core classes to train a taxonomy-enhanced text classifier, and generalizes the classifier using multi-label self-training. We also evaluate two ablations: **TaxoClass-**

[5] `https://radimrehurek.com/gensim/models/doc2vec.html`

[6] We also test the Flat-doc2vec variant which directly ranks all classes in the taxonomy and returns top ranked classes. Its performance is significantly worse than Hier-doc2vec and thus we only report Hier-doc2vec results.

[7] `https://github.com/yumeng5/WeSHClass`

[8] `https://github.com/HKUST-KnowComp/PathPredictionForTextClassification`

[9] `https://github.com/yinwenpeng/BenchmarkingZeroShot`

[10] `https://github.com/mickeystroller/TaxoClass`

NoST which removes the multi-label self-training step, and **TaxoClass-NoGNN** which replaces the GNN-based class encoder with a simple embedding layer initialized with pre-trained word embeddings.

5.5.3 Evaluation Metrics

We follow previous studies [161, 134] and evaluate the multi-label classification results from different aspects using various metrics. The first metric is **Example-F1**[11] which calculates the average F1 scores for all documents as follows:

$$\text{Example-F1} = \frac{1}{N}\sum_{i=1}^{N}\frac{2|\mathbb{C}_i^{true} \cap \mathbb{C}_i^{pred}|}{|\mathbb{C}_i^{true}| + |\mathbb{C}_i^{pred}|}, \tag{5.11}$$

where $\mathbb{C}_i^{true}$ ($\mathbb{C}_i^{pred}$) is the true (model predicted) class set of document D_i.

Moreover, as many applications formalize the HMTC as a class ranking problem [161, 162], we convert predicted class set $\mathbb{C}_i^{pred}$ into a rank list $\mathbb{R}_i^{pred}$ based on each class's model predicted probability and calculate **Precision at k** (**P@k**) as follows:

$$P@k = \frac{1}{N}\sum_{i=1}^{N}\frac{|\mathbb{C}_i^{true} \cap \mathbb{R}_{i,1:k}^{pred}|}{min(k, |\mathbb{C}_i^{true}|)}, \tag{5.12}$$

where $\mathbb{R}_{i,1:k}^{pred}$ is each method predicted top k most likely classes for D_i. Finally, for methods able to return the probability of a document being tagged with each class in the taxonomy, we calculate their **Mean Reciprocal Rank (MRR)** as follows:

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\mathbb{C}_i^{true}|}\sum_{c_j \in \mathbb{C}_i^{true}}\frac{1}{R_{ij}}, \tag{5.13}$$

where R_{ij} is the "rank" of document D_j's true class c_j in model predicted rank list.

5.5.4 Implementation Details

For all baseline methods except Hier-doc2vec, we use the public implementations from their authors and leave the hyper-parameters unchanged. For both Hier-0Shot-TC and our method, we adopt the same public `Roberta-Large-MNLI` model as the textual entailment model and use the same hypothesis template: "*this product is about* __ ." for Amazon-531 dataset and "*this example is* __." for DBPedia-298 dataset. We use AdamW optimizer to train our model with batch size 64, learning rate 5e-5 for all parameters in BERT document

[11] This metric is also called "micro-Dice coefficient".

Table 5.2: Evaluation of all compared text classification methods on two datasets. For some methods predicting a class path in a top-down fashion rather than returning all classes' probabilities, we cannot compute their MRR scores and indicate this using "N/A".

Method	Amazon-531				DBPedia-298			
	Example-F1	P@1	P@3	MRR	Example-F1	P@1	P@3	MRR
Hier-doc2vec [159]	0.3157	0.5805	0.3115	N/A	0.1443	0.2635	0.1443	N/A
WeSHClass [9]	0.2458	0.5773	0.2517	N/A	0.3047	0.5359	0.3048	N/A
TaxoClass-NoST	0.5431	0.7918	0.5414	0.5911	0.7712	0.8621	0.7712	0.8221
TaxoClass-NoGNN	0.5271	0.7642	0.5213	0.5621	0.7241	0.8154	0.7241	0.7692
TaxoClass	**0.5934**	**0.8120**	**0.5894**	**0.6332**	**0.8156**	**0.8942**	**0.8156**	**0.8762**
SS-PCEM [160]	0.2921	0.5369	0.2948	0.3004	0.3845	0.7424	0.3845	0.4032
Hier-0Shot-TC [142]	0.4742	0.7144	0.4610	N/A	0.6765	0.7871	0.6765	N/A

encoder and learning rate 4e-3 for all remaining parameters. During the multi-label self-training stage, we use learning rate 1e-6 for all parameters in the BERT document encoder and 5e-4 for all remaining parameters. We run all experiments on a single cluster with 80 CPU cores and a Quadro RTX 8000 GPU. All deep learning models are moved to the GPU for faster inference speed. With batch size 64, the TaxoClass framework consumes about 10GB GPU memory. In principle, all methods should be runnable on CPU.

5.5.5 Overall Performance Comparison

Table 5.2 presents the overall results of all compared methods. First, we find most weakly-supervised (*i.e.*, WeSHClass, TaxoClass and its variants) and zero-shot method (*i.e.*, Hier-0Shot-TC) can outperform the semi-supervised method SS-PCEM even the later has access to 30% of labeled documents. Second, we can see that TaxoClass has the overall best performance across all the metrics and defeats the second best method by a large margin. Comparing TaxoClass with TaxoClass-NoGNN, we show the importance of incorporating taxonomy structure into the class encoder. Moreover, the improvement of TaxoClass over TaxoClass-NoST demonstrates the effectiveness of our multi-label self-training.

5.5.6 Effectiveness of Core Class Mining

We evaluate the effectiveness of our core class mining method as follows. First, we define a set of rival methods and use them to generate various sets of "core classes". Then, we derive pseudo-training data for each generated core class set and use it to learn a text classifier with the same architecture as the one in TaxoClass. Finally, we report each model's performance

Table 5.3: Evaluation of core class mining algorithms on Amazon-531 dataset. We train the classifier using different training sets derived from different core class mining algorithm outputs.

Core Class Mining Method	Example-F1	P@1	P@3	MRR
Explicit Mention	0.1611	0.2168	0.1564	0.2045
0Shot	0.4793	0.7361	0.4782	N/A
Ours	**0.5431**	**0.7918**	**0.5414**	**0.5911**
Ours-NoCS	0.3812	0.6254	0.3831	0.4366
Ours-NoConf	0.2603	0.4431	0.2521	0.3014

on the test set. Note here we skip the self-training step to ensure the "core class based pseudo-training data" is the only variable.

Table 5.3 lists all the results. First, we find that the "Explicit Mention" method, which treats all classes with names explicitly appear in the corpus as the core classes, does not perform well for our HMTC problem. One reason could be many class names are human-curated summarizing phrases that do not appear in the corpus naturally. Second, the "0Shot" method views the output classes of baseline method Hier-0Shot-TC as the core classes and trains a new classifier. Interestingly, this new classifier performs better than the original Hier-0Shot-TC classifier, which shows that transferring knowledge from a general zero-shot classifier to a domain-specific classifier is a possible and promising direction. Finally, we compare variants of our own methods. The "Ours-NoCS" method removes the candidate core class selection step and treats all classes with high confidence scores as core classes. The "Ours-NoConf" method skips the confident core class identification step and views all candidate core classes as the final output core classes. We can see a significant performance drop on both ablations, which shows the importance of our two core class mining steps.

5.5.7 Analysis of Classifier Architecture

We study whether we can use the identified document core classes to train other text classifiers with different architectures such as fastText [163] and TextCNN [133]. As shown in Table 5.4, both methods achieve reasonable performance. We can also see that TaxoClass with and without GNN-enhanced class encoder can outperform both methods. This shows the effectiveness of our dual-encoder style classifier architecture.

Table 5.4: Performance of different classifiers on Amazon-531 dataset. All methods use the same training set derived from our identified document core classes.

Method	Example-F1	P@1	P@3	MRR
fastText	0.4472	0.7515	0.4521	0.4587
TextCNN	0.4787	0.7694	0.4771	0.4827
TaxoClass-NoGNN	0.5271	0.7642	0.5213	0.5621
TaxoClass	0.5934	0.8120	0.5894	0.6332

5.5.8 Supervision Signals in Class Names

We vary the percentage of labeled documents on Amazon-531 dataset for training a supervised fastText classifier and present its corresponding performance in Figure 5.5. We can see the performance of our TaxoClass framework is equivalent to that of a supervised fastText model learned using roughly 70% of labeled documents in the training set (*i.e.*, about 20,000 labeled documents).

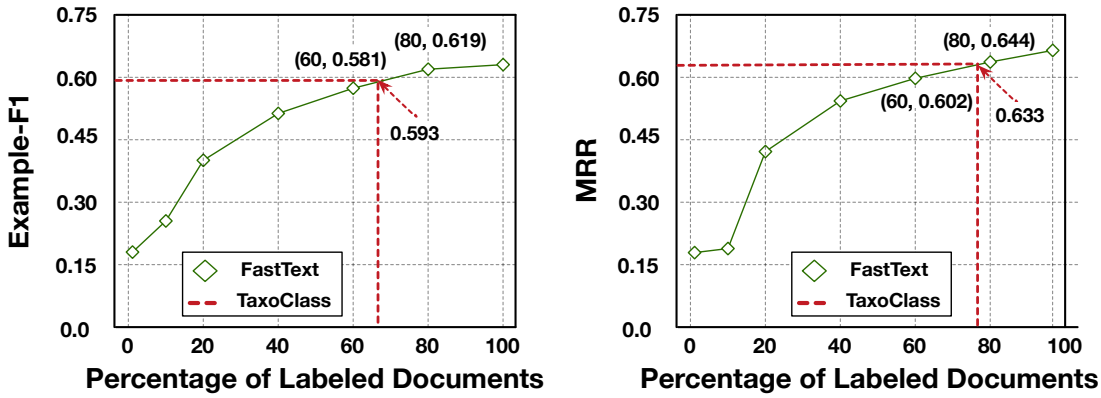


Figure 5.5: Comparison between TaxoClass and supervised fastText method on Amazon-531 dataset. We train the fastText model using on different percentages of labeled training documents.

5.6 SUMMARY

In this chapter, we study the hierarchical multi-label text classification problem when only class surface names, instead of massive labeled documents, are given. We propose a novel TaxoClass framework which leverages the class taxonomy to derive document core classes and learns taxonomy-enhanced text classifier for prediction. Extensive experiments demonstrate the effectiveness of TaxoClass on two real-world datasets from different domains.

In the future, we plan to explore how TaxoClass framework can be integrated with semi-supervised methods and data augmentation methods, when some class surface names are too ambiguous to indicate class semantics. Another another line, we may also interpret the document structuring task as a weakly-supervised hierarchical clustering problem where taxonomy nodes are viewed as seed guidances and potential cluster centers. Then, we can simultaneously cluster both documents and keywords to construct a document-allocated topic taxonomy. Finally, we consider extending our multi-label self-training method to other related NLP tasks such as fine-grained entity typing.

CHAPTER 6: INCORPORATING EVENT KNOWLEDGE IN TAXONOMY

6.1 OVERVIEW AND MOTIVATIONS

In the previous chapters, we have discussed how to construct, enrich, and apply a taxonomy to facilitate knowledge discovery from text data. Each node in this taxonomy is a concept represented by a single term. While being useful, this single term representation falls short of describing verbal and event-related concepts. For example, a single term "vaccine" cannot distinguish the events "*conduct research on vaccine*" versus "*distribute manufactured vaccines*". Therefore, if we can incorporate event knowledge into the taxonomy, we may significantly enlarge the application scope of taxonomy and unleash its full potentials.

To achieve the above goal, we need to first identify a set of salient events based on our given application-specific corpus. Most of traditional event extraction methods [164, 165, 166, 167] assume a set of predefined event types and their corresponding annotations are curated by human experts. This annotation process is expensive and time-consuming. Besides, those manually-defined event types often fail to generalize to new domains. For example, the widely used ACE 2005 event schemas[1] do not contain any event type about `Transmit Virus` or `Treat Disease` and thus cannot be readily applied to extract pandemic events. These prerequisites are often hard to be satisfied in real-world applications.

To automatically induce event types from raw text, researchers have studied ad-hoc clustering-based algorithms [168, 169] and probabilistic generative methods [170, 171, 172] to discover a set of event types and argument roles. These methods typically utilize bag-of-word text representations and impose strong statistical assumptions. Follow up work [173] relax those restrictions using a pipelined approach that leverages extensive lexical and semantic resources (*e.g.*, FrameNet [174], VerbNet [175], and PropBank [176]) to discover event schemas. While being effective, this method is limited by the scope of external resources and accuracies of its preprocessing tools. Recently, some studies [177, 178] have used transfer learning to extend traditional event extraction models to new types without explicitly deriving schemas of new event types. Nevertheless, these methods still require many annotations for a set of seen types.

In this chapter, we study the problem of *event type induction* which aims to discover a set of salient event types based on a given corpus. We observe that about 90% of event types can be frequently triggered by predicate verbs (c.f. Table 6.1) and thus propose to take a *verb-centric view* toward inducing event types. We use the five sentences (S1-S5) in

[1] `https://www.ldc.upenn.edu/collaborations/past-projects/ace`

Table 6.1: Statistics of verb triggered event types in three popular event extraction datasets. Event types triggered by verbs more than 5 times are considered as "Verb Frequently Triggered Event Types".

Datasets	ACE	ERE	RAMS
# of All Event Types	33	38	138
# of Verb Triggered Event Types	33	38	133
# of Verb Frequently Triggered Event Types	28	36	124

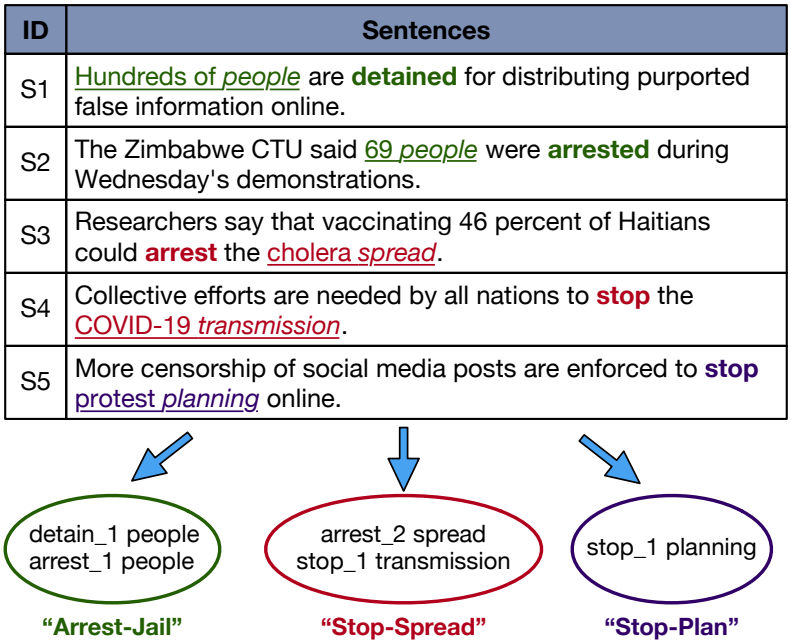


Figure 6.1: Motivating example sentences and induced event types. Predicates are in bold. Objects are underlined and *object heads* are in italics. Colors indicate event types. The suffix number followed by each predicate verb lemma indicates the predicate verb sense.

Figure 6.1 to motivate our design of event type representation. First, we observe that verb lemma itself might be ambiguous. For example, the two mentions of lemma "*arrest*" in S2 and S3 have different senses and indicate different event types. Second, even for predicates with the same sense, their different associated object heads[2] could lead them to express different event types. Taking S4 and S5 as examples, two "*stop*" mentions have the same sense but belong to different types because of their corresponding object heads. Finally, we can see that people have multiple ways to communicate the same event type due to the language variability.

[2]Intuitively, the object head is the most essential word in the object such as "*people*" in object "*hundreds of people*".

From the above observations, we propose to represent an event type as a cluster of ⟨predicate sense, object head⟩ pairs (P-O pairs for short)[3]. We present a new event type induction framework ETypeClus [180] to automatically discover event types, customized for a specific input corpus. ETypeClus requires no human-labeled data other than an existing general-domain verb sense dictionary such as VerbNet [175] and OntoNotes Sense Groupings [181]. ETypeClus contains four major steps. First, it extracts ⟨predicate, object head⟩ pairs from the input corpus based on sentence dependency tree structures. As some extracted pairs could be too general (*e.g.*, ⟨say, it⟩) or too specific (*e.g.*, ⟨document, microcephaly⟩), the second step of ETypeClus will identify salient predicates and object heads in the corpus. After that, we disambiguate the sense of each predicate verb by comparing its usage with those example sentences in a given verb sense dictionary. Finally, ETypeClus clusters the remaining salient P-O pairs into event types using a latent space generative model. This model jointly embeds P-O pairs into a latent spherical space and performs clustering within this space. Thus, we can guide the latent space learning with the clustering objective and enable the clustering process to benefit from the well-separated structure of the latent space.

We show our ETypeClus framework can save annotation cost and output corpus-specific event types on three datasets. The first two are benchmark datasets ACE 2005 and ERE (Entity Relation Event) [182]. ETypeClus can successfully recover predefined types and identify new event types such as `Build` in ACE and `Bombing` in ERE. Furthermore, to test the performance of ETypeClus in new domains, we collect a corpus about the disease outbreak scenario. Results show that ETypeClus can identify many interesting fine-grained event types (*e.g.*, `Vaccinate`, `Test`) that align well with human annotations.

The major contributions of this chapter are summarized as follows:

- We present a new event type representation as a ⟨predicate sense, object head⟩ cluster.
- We propose a novel event type induction framework ETypeClus that automatically disambiguates predicate senses and learns a latent space with desired event cluster structures.
- We conduct extensive experiments on three datasets to verify the effectiveness of ETypeClus in terms of both automatic and human evaluations.

The rest of this chapter is presented as follows. We first discuss the related work in Section 6.2. Then, we formalize our task in Section 6.3 and present our ETypeClus framework in Section 6.4. After that, we present the experiment results in Section 6.5 and finally

[3]Subjects are intentionally left here because [179] finds objects play a more important role in determining predicate semantics. Also, many P-O pairs indicate the same event type but share different subjects (*e.g.*, "*police capture X*" and "*terrorists capture X*" are considered as two different events but belong to the same event type `Capture Person`. Adding subjects may help divide current event types into more fine-grained types and we leave this for future work.

conclude this chapter in Section 6.6.

6.2 RELATED WORK

There are two major lines of work related to our study.

Event Schema Induction. Early studies on event schema induction adopt rule-based approaches [183] and classification-based methods [184, 185] to induce templates from labeled corpus. Later, unsupervised methods are proposed to leverage relation patterns [168] and coreference chains [169] for event schema induction. Typical approaches use probabilistic generative models [170, 171, 172, 186, 167] or ad-hoc clustering algorithms [173, 187] to induce predicate and argument clusters. In particular, [188] takes an entity-centric view toward event schema induction. It clusters entities into semantic slots and finds predicates for entity clusters in a post-processing step. [189] studies the event profiling task and includes one module that leverages a Bayesian generative model to cluster ⟨predicate:role:label⟩ triplets into event types. These methods typically rely on discrete hand-crafted features derived from bag-of-word text representations and impose strong statistics assumptions; whereas our method uses pre-trained language models to reduce the feature generation complexity and relaxes stringent statistics assumptions via latent space clustering.

Weakly-Supervised Event Extraction. Some studies on event extraction [190, 191, 192] propose to leverage annotations for a few seen event types to help extract mentions of new event types specified with just a few keywords. These methods reduce the annotation efforts but still require all target new types to be given. Recently, some studies [177, 178] use transfer learning techniques to extend traditional event extraction models to new types without explicitly deriving schemas of new event types. Compared to our study, these methods still require many annotations for a set of seen types and their resulting vector-based event type representations are less human interpretable. Another related work by [193] uses GAN to extract events from an open domain corpus. It clusters ⟨entity:location:keyword:date⟩ quadruples related to the same event rather than finds event types.

6.3 PROBLEM FORMULATION

In this section, we first introduce some important concepts and then formalize our task.

Notations. A **corpus** $\mathcal{S} = \{S_1, \ldots, S_N\}$ is a set of sentences where each sentence $S_i \in \mathcal{S}$ is a word sequence $[w_{i,1}, \ldots, w_{i,n}]$. A **predicate** is a *verb mention in a sentence* and can optionally have an associated **object** in the same sentence. We follow previous studies [194,

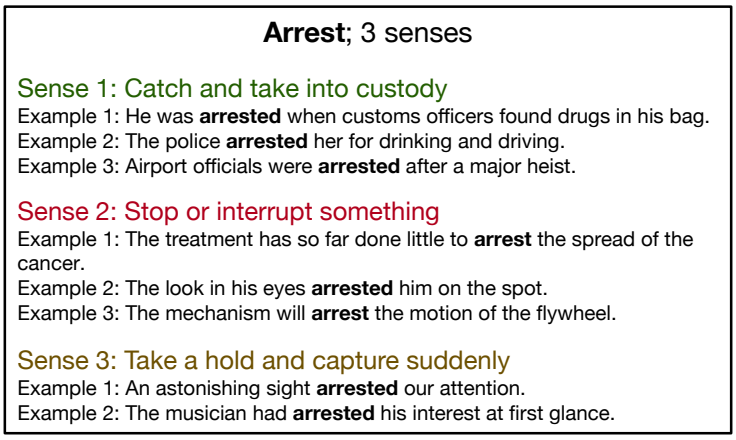


Figure 6.2: One example in verb sense dictionary $\mathcal{V}$.

195] and refer to the most important word in the object as the **object head**. For example, one predicate from the first sentence in Figure 6.1 is "*detain*" and its corresponding object is "*hundreds of people*" with the word "*people*" being the object head.

As predicates with the same lemma may have different senses, we disambiguate each predicate verb based on a **verb sense dictionary** $\mathcal{V}$ wherein each verb lemma has a list of candidate senses with example usage sentences. One illustrative example of our verb sense dictionary is shown in Figure 6.2. We refer to the sense of predicate verb lemma as the **predicate sense**.

Task Definition. Given a corpus $\mathcal{S}$ and a verb sense dictionary $\mathcal{V}$, our task of *event type induction* is to identify a set of K event types where each type T_j is represented by a cluster of $\langle$predicate sense, object head$\rangle$ pairs.

6.4 THE ETYPECLUS FRAMEWORK

Our proposed ETypeClus framework (outlined in Figure 6.3) induces event types in four major steps: (1) predicate and object head extraction, (2) salient predicate lemma and object head selection, (3) predicate sense disambiguation, and (4) latent space joint predicate sense and object head clustering.

6.4.1 Predicate and Object Head Extraction

We propose a lightweight method to extract predicates and object heads in sentences without relying on manually-labeled training data. Specifically, given a sentence S_i, we first

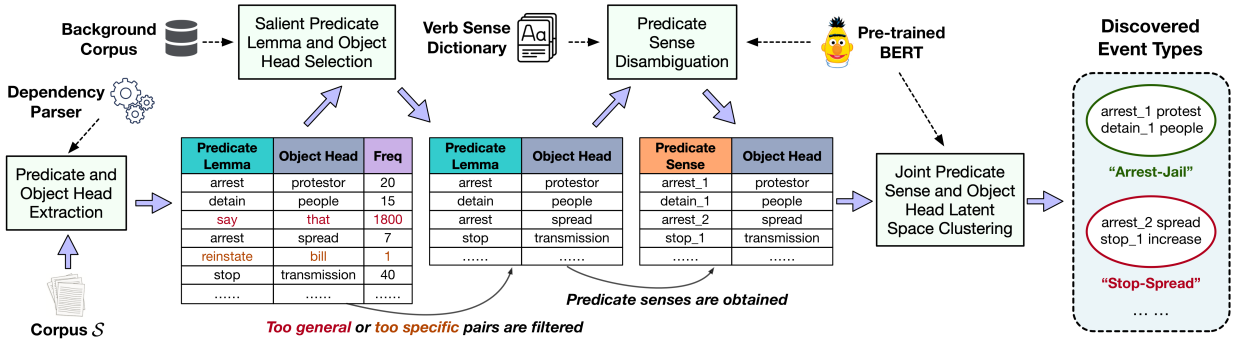


Figure 6.3: Our ETypeClus framework overview.

use a dependency parser[4] to obtain its dependency parse tree and select all non-auxiliary verb tokens[5] as our candidate predicates. Then, for each candidate predicate, we check its dependent words and if any of them has a dependency label `auxpass`, we believe this predicate verb is in passive voice and find its object heads within its syntactic children that occur before it and have a dependency label in SUBJECT label set[6]. Otherwise, we consider this predicate is in active voice and identify its object heads within its dependents that occur after it and have a dependency label in OBJECT label set[7]. Finally, we aggregate all ⟨predicate, object head⟩ pairs along with their frequencies in the corpus.

6.4.2 Salient Predicate Lemma and Object Head Selection

The above extracted ⟨predicate, object head⟩ pairs have different qualities. Some are too general and contain little information, while others are too specific and hard to generalize. Thus, this step of ETypeClus tries to select those salient predicate lemmas and object heads from our input corpus.

We compute the salience of a word (either a predicate lemma or an object head) based on two criteria. First, it should appear frequently in our corpus. Second, it should not be too frequent in a large general-domain background corpus[8]. Computationally, we follow the TF-IDF idea and define the word salience as follows:

$$Salience(w) = \big(1 + \log(freq(w))^2\big) \log\Big(\frac{N\_bs}{bsf(w)}\Big), \tag{6.1}$$

where $freq(w)$ is the frequency of word w, N_bs is the number of background sentences, and

[4]We use the Spacy `en_core_web_lg` model.
[5]A token with part-of-speech tag `VERB` and dependency label not equal to `aux` and `auxpass`.
[6]{`nsubj(pass)`, `csubj(pass)`, `agent`, `expl`}
[7]{`dobj`, `dative`, `attr`, `oprd`}
[8]We use the English Wikipedia 20171201 dump as our background corpus.

Table 6.2: Top 5 salient predicate lemmas (PredL) and object heads (ObjH) in three datasets.

ACE		ERE		Pandemic	
PredL	ObjH	PredL	ObjH	PredL	ObjH
kill	weapon	pay	money	infect	virus
pay	iraqis	kill	people	suspect	outbreak
guess	nations	rape	kid	sicken	vaccine
convict	states	send	weapon	test	case
fire	marines	attack	cadre	circulate	infection

$bsf(w)$ is the background sentence frequency of word w. Finally, we select those terms with salience scores ranked in top 80% as our salient predicate lemmas and object heads. Table 6.2 lists the top 5 most salient predicate lemmas and object heads in three datasets. The first two datasets contain news articles about wars and thus terms like "*kill*" and "*weapon*" are ranked top. The third dataset includes articles about disease outbreaks and thus most salient terms include "*infect*", "*virus*", and "*outbreak*".

6.4.3 Predicate Sense Disambiguation

As verbs typically exhibit large sense ambiguities, we disambiguate each predicate's sense in the sentence. Huang *et al.* [173] achieves this goal by utilizing a supervised word sense disambiguation tool [196] to link each predicate to a WordNet sense [197] and then mapping that sense back to an OntoNotes sense grouping [181]. In this work, we propose to remove such extra complexity and present a lightweight sense disambiguation method that requires only a verb sense dictionary.

The key idea of our method is to compare the usage of a predicate with each verb sense's example sentences in the dictionary. Given a predicate verb v in sentence S_i, we compute two types of features to capture both its *content* and *context* information. The first one, denoted as $\mathbf{v}^{emb}$, is obtained by feeding the sentence S_i into the BERT-Large model [55] and retrieving the predicate's corresponding contextualized embedding. The second feature $\mathtt{v}^{mwp}$ is a rank list of 10 alternative words that can be used to replace v in sentence S_i. Specifically, we replace the original word v in S_i with a special `[MASK]` token and feed the masked sentence S_i^{mask} into BERT-Large for masked word prediction. From the prediction results, we select the top 10 most likely words and sort them into $\mathtt{v}^{mwp}$.

After obtaining the predicate representation, we compute the representations of its candidate senses in the dictionary. Suppose the lemma of this predicate v has N_v candidate senses in the dictionary and each sense E_j, $j \in [1, \ldots, N_v]$ has N_j example sentences $\{S_{j,k}\}_{k=1}^{N_j}$ in

the dictionary. Then, within each example sentence $S_{j,k}$, we locate where the predicate lemma v occurs and compute its corresponding feature $\mathbf{v}_{j,k}^{emb}$ and $\mathsf{v}_{j,k}^{mwp}$ similarly as discussed before. After that, we obtain two types of features for each sense E_j as follows:

$$\mathbf{E}_j^{emb} = \frac{1}{N_j}\sum_{k=1}^{N_j}\mathbf{v}_{j,k}^{emb}, \quad \mathsf{E}_j^{mwp} = RA(\{\mathsf{v}_{j,k}^{mwp}\}|_{k=1}^{N_j}), \tag{6.2}$$

where $RA(\cdot)$ stands for the rank aggregation operation based on mean reciprocal rank. This method is widely used in previous literature [1, 198, 2, 17] for fusing ranked lists. Finally, we choose the sense that is most similar to the predicate v as follows:

$$j^* = \underset{j\in[1,\ldots,N_v]}{\arg\max}\ \cos(\mathbf{v}^{emb}, \mathbf{E}_j^{emb}) \cdot \mathrm{rbo}(\mathsf{v}^{mwp}, \mathsf{E}_j^{mwp}), \tag{6.3}$$

where $\cos(\mathbf{x}, \mathbf{y})$ is the cosine similarity between two vectors $\mathbf{x}$ and $\mathbf{y}$, and $\mathrm{rbo}(\mathsf{a}, \mathsf{b})$ is the rank-biased overlap similarity [199] between two ranked lists.

We evaluate our method on the verb subset of standard word sense disambiguation benchmarks [200]. Our method achieves 55.7% F1 score. In comparison, the supervised IMS method in [173] gets a 56.9% F1 score. Thus, we think our method is comparable to supervised IMS but being more lightweight and requires no training data.

6.4.4 Latent Space Joint Predicate Sense and Object Head Clustering

After obtaining salient ⟨predicate sense, object head⟩ pairs (P-O pairs for short), we aim to cluster them into event types. Below, we first discuss how to obtain the initial features for predicate senses and object heads. As those predicate senses and object heads are living in two separate spaces, we aim to fuse them into one joint feature space wherein the event cluster structures are better preserved. We achieve this goal by proposing a latent space generative method that jointly embeds P-O pairs into a unified spherical space and performs clustering in this space. Finally, we discuss how to train this generative model.

Initial Feature Acquisition. We obtain two types of features for each term w (either a predicate sense w_p or an object head w_o) by first locating its mentions in the corpus and then aggregating mention-level representations into term-level features. Suppose term w appears M_w times, for each of its mentions $m_{w,l}, l \in [1, \ldots, M_w]$, we extract this mention's content feature $\mathbf{m}_{w,l}^{emb}$ and context feature $\mathsf{m}_{w,l}^{mwp}$, following the same process discussed in Section 6.4.3. Then, we average all mentions' content features into this term's content feature $\mathbf{m}_w^{emb} = \frac{1}{M_w}\sum_{l=1}^{M_w}\mathbf{m}_{w,l}^{emb}$.

The aggregation of mention context features is more difficult as each $\mathsf{m}_{w,l}^{mwp}$ is not a numerical vector but instead a set of words predicted by BERT to replace $m_{w,l}$. In this work,

we propose the following aggregation scheme. For each term w, we first construct a pseudo document D_w using the bag union operation[9]. Then, we obtain the vector representations of pseudo documents based on TF-IDF transformation and apply Principal Component Analysis (PCA) to reduce the dimensionality of document vectors. The resulting vector will be considered as the term's context feature vector $\mathbf{m}_w^{mwp}$. Finally, we concatenate $\mathbf{m}_w^{emb}$ with $\mathbf{m}_w^{mwp}$ to obtain the initial feature vector of predicate senses (denoted as $\mathbf{h}_p$) and object heads (denoted as $\mathbf{h}_o$).

Latent Space Generative Model. To cluster P-O pairs into K event types based on two separate feature spaces ($\mathbf{H}_p$ for predicate sense and $\mathbf{H}_o$ for object head), one straightforward approach is to represent each P-O pair $x = (p, o)$ as $\mathbf{x} = [\mathbf{h}_p, \mathbf{h}_o]$ and directly applying clustering algorithms to all pairs. However, this approach cannot guarantee the concatenated space $\mathbf{H} = [\mathbf{H}_p, \mathbf{H}_o]$ will be naturally suited for clustering. Therefore, we propose to jointly embed and cluster P-O pairs in latent space $\mathbf{Z}$. By doing so, we can unify two feature spaces $\mathbf{H}_p$ and $\mathbf{H}_o$. More importantly, the latent space learning is guided by the clustering objective, and the clustering process can benefit from the well-separated structure of the latent space, which achieves a mutually-enhanced effect.

We design the latent space to have a spherical topology because cosine similarity more naturally captures word/event semantic similarities than Euclidean/L2 distance. Previous studies [100, 201] also show that learning spherical embeddings directly is better than first learning Euclidean embeddings and normalizing them later. Thus, we assume there is a spherical latent space $\mathbf{Z}$ with K clusters[10]. Each cluster in this space corresponds to one event type and is associated with a von Mises-Fisher (vMF) distribution [202] from which event type representative P-O pairs are generated. The vMF distribution of an event type c is parameterized by a mean vector $\mathbf{c}$ and a concentration parameter κ. A unit-norm vector $\mathbf{z}$ is generated from $\text{vMF}_d(\mathbf{c}, \kappa)$ with the probability as follows:

$$p(\mathbf{z}|\mathbf{c}, \kappa) = n_d(\kappa) \exp(\kappa \cdot \cos(\mathbf{z}, \mathbf{c})), \tag{6.4}$$

where d is the dimensionality of latent space $\mathbf{Z}$ and $n_d(\kappa)$ is a normalization constant.

Each P-O pair (p_i, o_i) with the initial feature $[\mathbf{h}_{p_i}, \mathbf{h}_{o_i}] \in \mathbf{H}_p \times \mathbf{H}_o$ is assumed to be generated as follows: (1) An event type c_k is sampled from a uniform distribution over K types; (2) a latent embedding $\mathbf{z}_i$ is generated from the vMF distribution associated with c_k; and (3) a function g_p (g_o) maps the latent embedding $\mathbf{z}_i$ to the original embedding $\mathbf{h}_{p_i}$ ($\mathbf{h}_{o_i}$)

[9]Namely, D_w contains a word T times if this word appears in T different $\mathsf{m}_{w,l}^{mwp}$, $l \in [1, \ldots, M_w]$.

[10]K is a hyper-parameter. We can either set K to the true number of event types (if it is known) or directly set K based on application-specific knowledge or adopt statistical methods to estimate K. In practice, we can set it to a large number and the resulting event types are still useful.

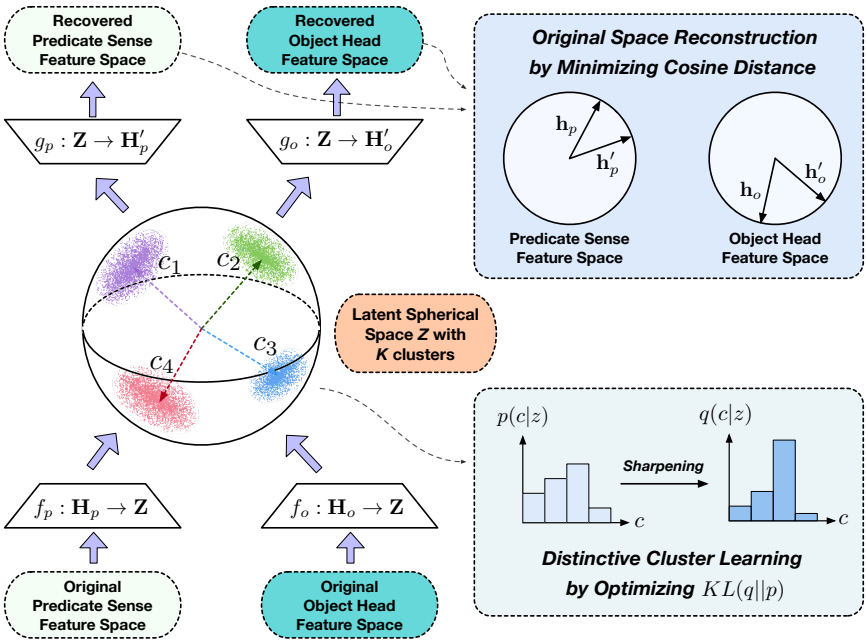


Figure 6.4: Overview of joint predicate sense and object head latent spherical space clustering. Detailed descriptions in Section 6.4.4.

corresponding to the predicate sense p_i (object head o_i). Namely, we have:

$$\begin{aligned} c_k \sim \text{Uniform}(K), \quad & \mathbf{z}_i \sim \text{vMF}_d(\mathbf{c}_k, \kappa), \\ \mathbf{h}_{p_i} = g_p(\mathbf{z}_i), \quad & \mathbf{h}_{o_i} = g_o(\mathbf{z}_i). \end{aligned} \tag{6.5}$$

We parameterize g_p and g_o as two deep neural networks and jointly learn the mapping function $f_p : \mathbf{H}_p \to \mathbf{Z}$ as well as $f_o : \mathbf{H}_o \to \mathbf{Z}$ from the original space to the latent space. Such a setup closely follows the autoencoder architecture [203] which is shown to be effective for preserving input information.

Model Training. We learn our generative model by jointly optimizing two objectives. The first one is a *reconstruction objective* defined as follows:

$$\mathcal{O}_{\text{rec}} = \sum_{i=1}^{N} \Big(\cos(\mathbf{h}_{p_i}, g_p(f_p(\mathbf{h}_{p_i}))) + \cos(\mathbf{h}_{o_i}, g_o(f_o(\mathbf{h}_{o_i}))) \Big) \tag{6.6}$$

This objective encourages our model to preserve input space semantics and generate the original data faithfully.

The second *clustering-promoting objective* enforces our model to learn a latent space with K well-separated cluster structures. Specifically, we use an expectation-maximization (EM) algorithm to sharpen the posterior event type distribution of each input P-O pair. In the expectation step, we first compute the posterior distribution based on current model param-

Algorithm 6.1: Latent Space Generative Model Training.

Input: A set of P-O pairs $\{x_i\}_{i=1}^N$; Initial feature spaces $\mathbf{H}_p$ and $\mathbf{H}_o$; Number of discovered event types K.

Output: Event-pair distributions $p(x_i|c_k)$.

```
f_o, f_p, g_o, g_p ← max O_rec in Eq. (6)   // Pretraining;
Initialize C = {c_k}_{k=1}^K;
while not converaged do
    // Update cluster assignment estimation;
    q(c_k|z_i) ← Eq. (8);
    // Update model parameteres;
    f_o, f_p, g_o, g_p, C ← max O_rec + λO_clus;
Return p(x_i|c_k) = p(z_i|c_k);
```

eters as follows:

$$p(c_k|\mathbf{z}_i) = \frac{p(\mathbf{z}_i|c_k)p(c_k)}{\sum_{k'=1}^{K} p(\mathbf{z}_i|c_{k'})p(c_{k'})} = \frac{\exp(\kappa \cdot \cos(\mathbf{z}_i, \mathbf{c}_k))}{\sum_{k'=1}^{K} \exp(\kappa \cdot \cos(\mathbf{z}_i, \mathbf{c}_{k'}))}. \tag{6.7}$$

We then compute a new estimate of each P-O pair's cluster assignment $q(c_k|\mathbf{z}_i)$ and use it to update the model in the maximization step. Instead of making hard cluster assignments like K-means which directly assigns each $\mathbf{z}_i$ to its closest cluster, we compute a soft assignment $q(c_k|\mathbf{z}_i)$ as follows:

$$q(c_k|\mathbf{z}_i) = \frac{p(c_k|\mathbf{z}_i)^2/s_k}{\sum_{k'=1}^{K} p(c_{k'}|\mathbf{z}_i)^2/s_{k'}}, \tag{6.8}$$

where $s_k = \sum_{i=1}^{N} p(c_k|\mathbf{z}_i)$. This squaring-then-normalizing formulation has a sharpening effect that skews the distribution towards its most confident cluster assignment, as shown in [156, 8, 9]. The formulation encourages unambiguous assignment of P-O pairs to event types so that the learned latent space will have gradually well-separated cluster structures. Finally, in the maximization step, we update the model parameters to maximize the expected log-probability of the current cluster assignments under the new cluster assignment estimates as follows:

$$\mathcal{O}_{\text{clus}} = \sum_{i=1}^{N} \sum_{k=1}^{K} q(c_k|\mathbf{z}_i) \log p(c_k|\mathbf{z}_i), \tag{6.9}$$

where p is updated to approximate fixed target q.

We summarize our training procedure in Algorithm 6.1. We first pretrain the model using only the reconstruction objective, which provides a stable initialization of all parameterized mapping functions. Then, we apply the EM algorithm to iteratively update all mapping functions and event type parameters $\mathbf{C}$ with a joint objective $\mathcal{O}_{\text{rec}} + \lambda\mathcal{O}_{\text{clus}}$ where the hyper-parameter λ balances two objectives. The algorithm is considered converged if less than 5%

of the P-O pairs change cluster assignment between two iterations or a maximum iteration number is reached[11]. Finally, we output each P-O pair's distribution over K event types.

6.5 EXPERIMENTS

We conduct two sets of experiments to verify the effectiveness of ETypeClus framework. Section 6.5.1 presents our results on two widely used benchmark datasets, ACE 2005 and ERE (Entity Relation Event) [182]. Section 6.5.2 presents the results on our newly created Pandemic dataset.

6.5.1 Experiments on ACE/ERE Datasets

Datasets. We first evaluate ETypeClus on two widely used event extraction datasets: ACE (Automatic Content Extraction) 2005[12] and ERE (Entity Relation Event) [182]. For both datasets, we follow the same preprocessing steps from [204, 167] and use sentences in the training split as our input corpus. The ACE dataset contains 17,172 sentences with 33 event types and the ERE dataset has 14,695 sentences with 38 types. We test the performance of ETypeClus on event type discovery and event mention clustering.

Event Type Discovery. We apply ETypeClus on each input corpus to discover 100 candidate event clusters and follow [173] to manually check whether discovered clusters can reconstruct ground truth event types. On ACE, we recover 24 out of 33 event types (19 out of 20 most frequent types) and 7 out of 9 missing types have a frequency less than 10. On ERE, we recover 28 out of 38 event types (18 out of 20 most frequent types). We show some example clusters in Table 6.3 which includes top ranked P-O pairs and their occurring sentences. We observe that ETypeClus successfully identifies human defined event types (*e.g.*, `Arrest-Jail` in ACE and `Transfer-Money` in ERE). It can also identify finer-grained types compared with the original ground truth types (e.g., the 4th row of Table 6.3 shows one discovered event type `Bombing` in ERE which is in finer scale than "Conflict:Attack", the closest human-annotated type in ERE). Further, ETypeClus is able to identify new salient event types (e.g., finding new event type `Build` in ACE). Finally, ETypeClus not only induces event types but also provides their example sentences, which serve as the *corpus-specific* annotation guidance.

Event Mention Clustering. We evaluate the effectiveness of our latent space generative

[11] In practice, this will certainly happen because our learning rate is gradually decreasing.

[12] `https://www.ldc.upenn.edu/collaborations/past-projects/ace`

Table 6.3: Example outputs of ETypeClus discovered event types with their associated sentences in ACE and ERE datasets. The first two types come from ACE and the remaining two are from ERE. The event types with superscript "$^{\nabla}$" originally do not exist in human-labeled schemas and are discovered by ETypeClus framework. Predicates are in bold and *object heads* are underlined and in italics.

Event Type	Top Ranked P-O Pairs	Example Sentences in Corpus
Arrest-Jail	⟨arrest_0, protester⟩ ⟨arrest_0, militant⟩ ⟨arrest_0, suspect⟩	• For the most part the marches went off peacefully, but in New York a small group of *protesters* were **arrested** after they refused to go home at the end of their rally, police sources said. • On Tuesday, Saudi security officials said three suspected al-Qaida *militants* were **arrested** in Jiddah, Saudi Arabia, in sweeps following the near-simultaneous suicide attacks on three residential compounds on the outskirts of Riyadh on May 12. • can owe tell us exactly the details, the precise details of how you **arrested** the *suspect*?
Build$^{\nabla}$	⟨build_0, facility⟩ ⟨build_0, center⟩ ⟨build_0, housing⟩	• Plans were underway to **build** destruction *facilities* at all other locations but now the Bush junta has removed from its proposed defense budget for fiscal year 2006 all but the minimum funding for these destruction projects. • Virginia is apparently going to be **build** a data *center* in Richmond, a back-up data center , and a help desk/call center as a follow-on to the creation of VITA, the Virginia Information Technology Agency. • The Habitat for Humanity might be a good one to consider, since their expertise is in **building** *housing*, which of course is so beadly needed over there at this time.
Transfer-Money	⟨fund_0, activity⟩ ⟨fund_0, operation⟩ ⟨fund_0, people⟩	• The grants will **fund** advisory *activities*, including local capacity building, infrastructure development, product development, and development of local insurance companies' capacity to provide index-based insurance products. • The White House had hoped to hold off asking for more money to **fund** military *operations* in Iraq and Afghanistan until after the election, but with costs rising faster than expected, it sent a request for an early installment of $25 billion to Congress this week. • Watch 'Secret Pakistan' on the BBC iPlayer , it's an awesome two part documentary about how Pakistan has been supporting and **funding** these *people* for years.
Bombing$^{\nabla}$	⟨bomb_0, factory⟩ ⟨bomb_0, checkpoint⟩ ⟨bomb_0, base⟩	• He **bombed** the Aspirin *factory* in 1998 (which turned out to have nothing to do with Bin Laden) the week he revealed he had been lying to us for eight months about Lewinsky. • Prosecutors then also pointed to the men's suicide bomber training in 2011 in Somalia and association with Beledi, who prosecutors said **bombed** a government *checkpoint* in Mogadishu that year. • Once the war breaks out, Iran will immediately use all kinds of missiles to **bomb** the military *bases* of the United States in the Gulf and Israel to pieces.

model via the event mention clustering task. We first match each event mention with one extracted P-O pair if possible, and select 15 event types with the most matched results. Then, for each selected type, we collect its associated mentions and add them into a candidate pool. We represent each mention using the feature of its corresponding P-O pair. Finally, we cluster all mentions in the candidate pool into 15 groups and evaluate whether they

align well with the original 15 types. The event mention clustering quality also serves as a good proxy of the event type quality. This is because if a method can discover good event types from a corpus, it should also be able to generate good event mention clusters when the ground truth number of clusters is given.

Compared Methods. We compare the following methods:

- **Kmeans**: A standard clustering algorithm that works in the Euclidean feature space. We run this algorithm with the ground truth number of clusters.
- **sp-Kmeans**: A variant of Kmeans that clusters event mentions in a spherical space based on the cosine similarity.
- **AggClus**: A hierarchical agglomerative clustering algorithm with Euclidean distance function and Ward linkage. A stop criterion is set to be reaching the target number of clusters.
- **Triframes** [205]: A graph-based clustering algorithm that constructs a k-NN event mention graph and uses a fuzzy graph clustering algorithm WATSET to generate the clusters.
- **JCSC** [173]: A joint constrained spectral clustering method that iteratively refines the clustering result with a constraint function to enforce inter-dependent predicates and objects to have coherent clusters.
- **ETypeClus**: Our proposed latent space joint embedding and clustering algorithm. For fair comparison, all methods start with the same $[\mathbf{h}_p, \mathbf{h}_o]$ embeddings.

We implement Kmeans and AggClus based on the Scikit-learn codebase [206]. We use L_2 distance for both methods. For Kmeans, we use k-means++ strategy for model initialization, and each time the result with the best inertia is used within 10 initializations. We use ward linkage for AggClus and set the stop criterion to be reaching the target number of clusters. For spherical Kmeans, we use an open source implementation[13]. Similar to Kmeans, we use k-means++ to initialize the model and select the best results among 10 initializations. For Triframes [205], we use its authors' original implementation[14], and tune the parameter k in the k-NN graph construction step for different tasks and datasets to get a reasonable number of clusters. Specifically, we use $k = 30$ for the event mention clustering task, which gives us the overall best evaluation results on both ACE and ERE. On the Pandemic corpus, we take $k = 100$, which generates 35 clusters that contain at least 40 tuples. For JCSC, we implement the clustering algorithm based on Algorithm 1 in [173]. The spectral clustering used in JCSC is based on Scikit-learn's implementation, and the label assigning strategy is K-means with 30 random initializations each time.

[13] `https://github.com/jasonlaska/spherecluster`

[14] `https://github.com/uhh-lt/triframes`

Evaluation Metrics. We evaluate clustering results with the following several standard metrics. We denote the ground truth clusters as C^*, the predicted clusters as C, and the total number of event mentions as N.

- **ARI** measures the similarity between two cluster assignments. Let $TP(TN)$ denote the number of element pairs in the same (different) cluster(s) in both C^* and C. Then, ARI is calculated as follows:

$$\text{ARI} = \frac{\text{RI} - \mathbb{E}(\text{RI})}{\max \text{RI} - \mathbb{E}(\text{RI})}, \quad \text{RI} = \frac{TP + TN}{N}, \tag{6.10}$$

  where $\mathbb{E}(\text{RI})$ is the expected RI of random assignments.

- **NMI** denotes the normalized mutual information between two cluster assignments and is widely used in previous studies. Let $\text{MI}(\cdot;\cdot)$ be the Mutual Information between two cluster assignments, and $\text{H}(\cdot)$ denote the Entropy. Then the NMI is formulated as follows:

$$\text{NMI} = \frac{2 \times \text{MI}(C^*; C)}{\text{H}(C^*) + \text{H}(C)}. \tag{6.11}$$

- **BCubed** estimates the quality of the generated cluster assignment by aggregating the precision and recall of each element. Specifically, we have:

$$\text{BCubed-P} = \frac{1}{N}\sum_{i=0}^{N} \frac{|C(e_i) \cap C^*(e_i)|}{|C(e_i)|}, \quad \text{BCubed-R} = \frac{1}{N}\sum_{i=0}^{N} \frac{|C(e_i) \cap C^*(e_i)|}{|C^*(e_i)|}, \tag{6.12}$$

  where $C^*(\cdot)$ ($C(\cdot)$) is the mapping function from an element to its ground truth (predicted) cluster. BCubed-F1 is the harmonic mean of BCubed-P and BCubed-R.

- **ACC** measures the clustering quality by finding the permutation function from predicted cluster IDs to ground truth IDs that gives the highest accuracy. Let y_i (y_i^*) denote the i-th element's predicted (ground truth) cluster ID, the ACC is formulated as follows:

$$\text{ACC} = \max_{\sigma \in Perm(k)} \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(y_i^* = \sigma(y_i)) \tag{6.13}$$

  where k is the number of clusters for both C^* and C, $Perm(k)$ is the set of all permutation functions on the set $\{1, 2, \ldots, k\}$, and $\mathbf{1}(\cdot)$ is the indicator function.

Experiment Results. Table 6.4 shows ETypeClus outperforms all the baselines on both datasets in terms of all metrics. The major advantage of ETypeClus is the latent event space: different types of information can be projected into the same space for effective clustering. We also observe that JCSC is the strongest among all baselines. We think the

Table 6.4: Event mention clustering results. All values are in percentage. We run each method 10 times and report its averaged result for each metric with the standard deviation. Note that ACC is not applicable for Triframes because it assumes the equal number of clusters in ground truth and generated results.

Methods	ACE				ERE			
	ARI (std)	NMI (std)	ACC (std)	BCubed-F1 (std)	ARI (std)	NMI (std)	ACC (std)	BCubed-F1 (std)
Kmeans	26.27 (1.60)	48.02 (1.55)	41.57 (3.07)	41.33 (1.75)	11.17 (1.83)	35.10 (2.36)	31.65 (1.82)	29.97 (1.79)
sp-Kmeans	26.06 (2.12)	47.30 (1.65)	40.41 (2.46)	39.52 (1.42)	13.62 (2.14)	37.33 (2.25)	33.28 (3.12)	30.73 (2.03)
AggClus	24.45 (0.00)	45.71 (0.00)	41.00 (0.00)	40.20 (0.00)	6.07 (0.00)	29.62 (0.00)	30.84 (0.00)	29.90 (0.00)
Triframes [205]	19.35 (6.60)	36.38 (4.91)	—	38.91 (2.36)	10.89 (2.51)	34.94 (2.54)	—	33.53 (4.47)
JCSC [173]	36.10 (4.96)	49.50 (2.70)	46.17 (3.64)	43.83 (3.17)	17.07 (4.40)	39.50 (3.97)	33.76 (2.43)	34.04 (2.23)
ETypeClus	**40.78 (3.20)**	**57.57 (2.40)**	**48.35 (2.55)**	**51.58 (2.50)**	**24.09 (1.93)**	**49.40 (1.37)**	**41.19 (1.87)**	**39.78 (1.45)**

reason is that it uses a joint clustering strategy where event types are defined as predicate clusters and the constraint function enables objects to refine predicate clusters. Thus, a predicate-centric clustering algorithm can outperform all other baselines, which supports our verb-centric view of events.

6.5.2 Experiments on Pandemic Dataset

Dataset. To evaluate the portability of ETypeClus to a new open domain, we collect a new dataset that includes 98,000 sentences about disease outbreak events. We follow a similar approach in [167] to construct our Pandemic Dataset. First, we resort to Wikipedia lists to get a set of Wikipedia articles related to disease outbreaks[15]. Then, we extract the news article links from the "references" section of those Wikipedia article pages. Finally, we crawl these news articles based on their above extracted links[16] and construct a corpus related to disease outbreaks.

Intrusion Test. We run the top-3 performing baselines (in ACE/ERE datasets) and ETypeClus to generate 30 candidate event types and evaluate their quality using intrusion test. Specifically, given the top-5 tuples of each detected type, we inject a randomly sampled tuple from the top results of other types to serve as a negative sample. For methods that have cluster centers, we rank tuples within each cluster by their distances to the center. Otherwise, we rank tuples according to their frequencies in the corpus. Then, the intrusion questions from all compared methods are randomly shuffled to avoid bias. Three annotators[17] are asked to identify the injected tuples independently, and we take the average of their labeling accuracy to show the quality of the generated event types. The intuition be-

[15]Specifically, we use the list `https://en.wikipedia.org/wiki/List_of_epidemics`

[16]We use the crawler tool at `https://github.com/codelucas/newspaper`.

[17]All three annotators are not in the author list of this paper and provide independent judgements of the tuple quality.

Table 6.5: ETypeClus discovered event types with their associated sentences in the Pandemic corpus. Predicates are in bold and *object heads* are underlined and in italics.

Event Type	Top Ranked P-O Pairs	Example Sentences in Corpus
Spread Virus	⟨spread_2, virus⟩ ⟨spread_2, disease⟩ ⟨spread_2, coronavirus⟩	• What is the best way to keep from **spreading** the *virus* through coughing or sneezing? • Farmers quickly mobilized to fight the misperceptions that pigs could **spread** the *disease*. • In the UK, Asians have been punched in the face, accused of **spreading** *coronavirus*.
Prevent Spread	⟨prevent_1, spread⟩ ⟨mitigate_1, spread⟩ ⟨mitigate_1, transmission⟩	• Infection prevention and control measures are critical to **prevent** the possible *spread* of MERS-CoV in health care facilities . • A vaccine can **mitigate** *spread*, but not fully prevent the virus circulating. • Asymptomatic infection could also potentially be directly harnessed to **mitigate** *transmission*.
Delay Gathering	⟨delay_1, gathering⟩ ⟨postpone_1, gathering⟩ ⟨suspend_1, gathering⟩	• The 2020 edition of the Cannes Film Festival, was left in limbo following an announcement from the festival's organizers that the *gathering* could be **delayed** until late June or early July. • States with EVD should consider **postponing** mass *gatherings* until EVD transmission is interrupted. • On Thursday, leaders of The Church of Jesus Christ of Latter - day Saints told its 15 million members worldwide all public *gatherings* would be **suspended** until further notice .
Provide Testing	⟨provide_1, testing⟩ ⟨conduct_1, testing⟩ ⟨perform_1, testing⟩	• Governments are racing to buy medical equipment as a debate intensifies over **providing** adequate *testing*, when it 's advisable to wear masks, and whether stricter lockdowns should be imposed. • Additional *testing* is being **conducted** to confirm that the family members had H1N1 and to try to verify that the flu was transmitted from human to cat. • Additional laboratories **perform** antiviral *testing* and report their results to CDC .
Warn Country	⟨warn_1, country⟩ ⟨warn_1, authority⟩ ⟨warn_1, government⟩	• WHO uses six phases of alert to communicate the seriousness of infectious threats and to **warn** *countries* of the need to prepare and respond to outbreaks. • The message showed a photo of a letter, written by the operators of the hospital's oxygen supply plant, **warning** the *authorities* that the supply was running dangerously low . • WHO staff concluded there was a high risk of further spread, and issued a global alert to **warn** all member *governments* of the existence of a new and highly infectious form of "atypical pneumonia" on March 12th .
Vaccinate People	⟨vaccinate_0, person⟩ ⟨immunize_0, people⟩ ⟨vaccinate_0, family⟩	• All *persons* in a recommended vaccination target group should be **vaccinated** with the 2009 H1N1 monovalent vaccine and the seasonal influenza vaccine. • U.K. Will Start **Immunizing** *People* Against COVID-19 On Tuesday, Officials Say. • "In the Samoan language there is no word for bacteria or virus" says Henrietta Aviga, a nurse travelling around villages to **vaccinate** and educate *families*.

Table 6.6: Intrusion test results in percentage.

Methods	K-Menas	AggClus	JCSC	ETypeClus
Accuracy	86.7	64.4	54.4	**91.1**

hind this test is that the annotators will be easier to identify the intruders if the clustering results are clean and tuples are semantically coherent. As shown in Table 6.6, ETypeClus achieves the highest accuracy among all the baseline methods, indicating that it generates semantically coherent types in each cluster.

Case Studies. Table 6.5 shows some discovered event types of ETypeClus. Interesting examples include tuples with the same predicate sense but object heads with different granularities (*e.g.*, ⟨spread_2, virus⟩ and ⟨spread_2, coronavirus⟩ for `Spread-Virus` type), tuples with same object head but different predicate senses (*e.g.*, ⟨prevent_1, spread⟩, and ⟨mitigate_1, spread⟩ for `Prevent-Spread` type), and event types with predicate verb lemmas that are not directly linkable to OntoNotes Senses grouping (*e.g.*, "*immunize*" and "*vaccinate*" for `Vaccinate` type).

6.6 SUMMARY

In this chapter, we study the event type induction problem that aims to automatically generate salient event types for a given corpus. We define a novel event type representation as a ⟨predicate sense, object head⟩ cluster, and propose ETypeClus that can extract and select salient predicates and object heads, disambiguate predicate senses, and jointly embed and cluster P-O pairs in a latent space. Experiments on three datasets show that ETypeClus can recover human curated types and identify new salient event types.

In the future, we propose to explore the following directions: (1) improve predicate and object extraction quality with tools of higher semantic richness (*e.g.*, a SRL labeler or an AMR parser); (2) leverage more information from lexical resources to enhance event representation; and (3) cluster objects into argument roles for each discovered event type.

CHAPTER 7: CONCLUSIONS

7.1 SUMMARY

In this thesis, we have proposed a principled framework that automatically constructs, enriches, and applies taxonomies for unleashing hidden knowledge in unstructured text. The whole automated framework requires minimum human labeled data and obtains good performances by (1) leveraging user-provided seed information as weak supervision, (2) utilizing salient statistical signals in unlabeled data as self supervision, and (3) resorting to existing knowledge repositories as distant supervision. Those obtained taxonomies, either a concept taxonomy or an event taxonomy, not only contain useful knowledge in themselves but also can be used to organize and index knowledge, help users search and comprehend knowledge stored in a large corpus, and provide structured guidance in many other knowledge engineering tasks.

7.2 FUTURE WORK

With the mounting big data and diverse applications in today's information-based society, my proposed taxonomy-centric framework will play an increasingly important role in organizing concepts, data, and knowledge. My long-term research goal is to create data-driven methods that ingest massive heterogeneous data, organize machine-actionable knowledge into taxonomies, and utilize taxonomies to facilitate human decision-making. Many unique challenges arise in this context and call for collaborative research efforts from multiple areas, including data mining, machine learning, natural language processing, computer vision, human-computer interaction, computer security, and much more. Below are some specific directions that I am excited to explore in the near future:

7.2.1 Integrate heterogeneous modalities and sources

While my current research primarily focuses on text data, recent years have witnessed a trend in the confluence of multiple data modalities (*e.g.*, graphs, image, time series) gathered from heterogeneous sources. On one hand, introducing multi-modal data opens the gate to more labeled datasets for training models. On the other hand, human's understanding is often grounded on multiple data modalities. A multi-modal multi-task system can reflect a better understanding of the world. Some study [4] has already shown that more

meaningful taxonomic relations can be recovered from text-rich networks (*e.g.*, a citation network with paper abstracts, a social network with user posts, *etc.*) from their corresponding corpora without metadata. Furthermore, people find that leveraging the information in taxonomy can help learn better representations of knowledge graph (modeled as a heterogeneous information network) [207]. My proposed framework, if properly extended, could well accommodate such heterogeneity. I plan to develop novel embedding methods that organize heterogeneous data into a shared latent space for bridging different modalities.

7.2.2 Engage with human behaviors and interactions

For many complex analytical tasks, humans and machines need to collaborate to acquire necessary task-specific knowledge. There is great potential to adopt my proposed taxonomy-centric framework to facilitate such a human-in-the-loop process: (1) machines input user-provided seed data as weak supervision and task guidance, perform data analysis by learning task-specific models, and return interpretable patterns and visualizations; and (2) humans make sense of the resultant patterns and visual cues, adjust or provide additional seed data, and give feedback to guide the machines to extract more useful knowledge. I am interested in working with researchers in HCI, visualization, and machine learning to address fundamental challenges for realizing this goal. Example research problems include: How to design intuitive interfaces to help users provide seed data for various applications? How to train a predictive and interpretable model using limited labeled data from the user input? How to develop taxonomy-tailored visualization techniques to help users more easily gain useful knowledge? How to update a machine learning model continuously based on user feedback to better satisfy users' information needs? To answer some of the above questions, I plan to leverage the prompt-based few-shot learning techniques to incorporate sparse example-based user feedbacks into our current models.

7.2.3 Preserve data privacy and model security

While the knowledge discovery process from multiple data sources through collective learning is powerful, the uncensored information extraction and exchanging of multi-party data may be prone to adversarial attacks and threaten users' privacy. For example, an online retailer may provide false or exaggerated product descriptions to fool a category-guided recommender system to rank their products in top positions. An identity thief may conduct differential privacy attacks to infer sensitive user information, which is extremely dangerous and concerning. My future work on taxonomy will also study privacy-preserving data

mining methods and federated learning techniques to prevent such attacks. Several research directions include: (1) enforcing the differential privacy principle to select and anonymize personal identifiable features for achieving the optimal trade-off between model utility and data privacy, and (2) leverage the multi-party computation techniques to achieve secure data and knowledge exchange and prevent information leakage propagation.

REFERENCES

[1] J. Shen, Z. Wu, D. Lei, J. Shang, X. Ren, and J. Han, "SetExpan: Corpus-based set expansion via context feature selection and rank ensemble," in *Proceedings of the 2017 Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2017.

[2] Y. Zhang, J. Shen, J. Shang, and J. Han, "Empower entity set expansion via language model probing," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[3] J. Shen, Z. Wu, D. Lei, C. Zhang, X. Ren, M. Vanni, B. M. Sadler, and J. Han, "HiExpan: Task-guided taxonomy construction by hierarchical tree expansion," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[4] Y. Shi, J. Shen, Y. Li, N. Zhang, X. He, Z. Lou, Q. Zhu, M. Walker, M.-H. Kim, and J. Han, "Discovering hypernymy in text-rich heterogeneous information network by exploiting context granularity," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019.

[5] J. Shen, Z. Shen, C. Xiong, C. Wang, K. Wang, and J. Han, "TaxoExpan: Self-supervised taxonomy expansion with position-enhanced graph neural network," in *Proceedings of the 2020 Web Conference*, 2020.

[6] J. Zhang, X. Song, Y. Zeng, J. Chen, J. Shen, Y. Mao, and L. Li, "Taxonomy completion via triplet matching network," in *Proceedings of the 2021 AAAI Conference on Artificial Intelligence*, 2021.

[7] X. Song, J. Shen, J. Zhang, and J. Han, "Who should go first? a self-supervised concept sorting model for improving taxonomy expansion," in *Proccedings of the International Workshop on Self-Supervised Learning for the Web*, 2021.

[8] Y. Meng, J. Shen, C. Zhang, and J. Han, "Weakly-supervised neural text classification," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018.

[9] Y. Meng, J. Shen, C. Zhang, and J. Han, "Weakly-supervised hierarchical text classification," in *Proceedings of the 2019 AAAI Conference on Artificial Intelligence*, 2019.

[10] J. Shen, J. Xiao, X. He, J. Shang, S. Sinha, and J. Han, "Entity set search of scientific literature: An unsupervised ranking approach," in *Proccedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018.

[11] J. Shen, J. Xiao, Y. Zhang, C. Yang, J. Shang, J. Han, S. Sinha, P. Ping, R. Weinshilboum, Z. Lu, and J. Han, "SetSearch+: Entity-set-aware search and mining for scientific literature," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Demo)*, 2018.

[12] C. E. Lipscomb, "Medical subject headings (MeSH)," in *Bulletin of the Medical Library Association*, 2000.

[13] L. N. Cassel, S. Palivela, S. Marepalli, A. Padyala, R. Deep, and S. Terala, "The new ACM CCS and a computing ontology," in *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital libraries*, 2013.

[14] R. S. Gonçalves, M. Horridge, R. Li, Y. Liu, M. A. Musen, C. I. Nyulas, E. Obamos, D. Shrouty, and D. Temple, "Use of OWL and semantic web technologies at Pinterest," in *Proceedings of 2019 International Semantic Web Conference*, 2019.

[15] W. Zhu, H. Gong, J. Shen, C. Zhang, J. Shang, S. Bhat, and J. Han, "FUSE: Multi-faceted set expansion by coherent clustering of skip-grams," in *Proceedings of 2020 Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2020.

[16] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.

[17] J. Huang, Y. Xie, Y. Meng, J. Shen, Y. Zhang, and J. Han, "Guiding corpus-based set expansion by auxiliary sets generation and co-expansion," in *Proceedings of the 2020 Web Conference*, 2020.

[18] W. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: A probabilistic taxonomy for text understanding," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 2012.

[19] P. Velardi, S. Faralli, and R. Navigli, "Ontolearn reloaded: A graph-based algorithm for taxonomy induction," in *Computational Linguistics*, 2013.

[20] C. Wang, X. He, and A. Zhou, "A short survey on taxonomy learning from text corpora: Issues, resources and recent advances," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

[21] Y. Yu, Y. Li, J. Shen, H. Feng, J. Sun, and C. Zhang, "STEAM: Self-supervised taxonomy expansion with mini-paths," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.

[22] H. Zha, J. Shen, K. Li, W. Greiff, M. Vanni, J. Han, and X. Yan, "FTS: Faceted taxonomy construction and search for scientific publications," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Demo)*, 2018.

[23] J. Shen, W. Qiu, Y. Meng, J. Shang, X. Ren, and J. Han, "TaxoClass: Hierarchical multi-label text classification using only class names," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

[24] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. P. Hsu, and K. Wang, "An Overview of Microsoft Academic Service (MAS) and Applications," in *Proceedings of the 2015 International Conference on World Wide Web*, 2015.

[25] R. C. Wang and W. W. Cohen, "Language-independent set expansion of named entities using the web," in *Proceedings of the 7th IEEE International Conference on Data Mining*, 2007.

[26] S. Gupta and C. D. Manning, "Improved pattern learning for bootstrapped entity extraction," in *Proceedings of the 18th Conference on Computational Natural Language Learning*, 2014.

[27] Z. Chen, M. Cafarella, and H. Jagadish, "Long-tail vocabulary dictionary extraction from the web," in *Proceedings of the 9th ACM international conference on Web search and data mining*, 2016.

[28] S. Tong and J. Dean, "System and methods for automatically creating lists," 2008, US Patent 7,350,187.

[29] P. Pantel, E. Crestan, A. Borkovsky, A.-M. Popescu, and V. Vyas, "Web-scale distributional similarity and entity set expansion," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 2009.

[30] S. Shi, H. Zhang, X. Yuan, and J.-R. Wen, "Corpus-based semantic class mining: Distributional vs. pattern-based approaches," in *Proceedings of the 23rd International Conference on Computational Linguistics*, 2010.

[31] Y. He and D. Xin, "SEISA: Set expansion by iterative similarity aggregation," in *Proceedings of the 20th International Conference on World Wide Web*, 2011.

[32] C. Wang, K. Chakrabarti, Y. He, K. Ganjam, Z. Chen, and P. A. Bernstein, "Concept expansion using web tables," in *Proceedings of the 24th International Conference on World Wide Web*, 2015.

[33] X. Rong, Z. Chen, Q. Mei, and E. Adar, "Egoset: Exploiting word ego-networks and user-generated ontology for multifaceted set expansion," in *Proceedings of the 9th ACM international conference on Web search and data mining*, 2016.

[34] B. Shi, Z. Zhang, L. Sun, and X. Han, "A probabilistic co-bootstrapping method for entity set expansion," in *Proceedings of the 25th International Conference on Computational Linguistics*, 2014.

[35] S. Gupta and C. D. Manning, "Distributed representations of words to guide bootstrapped entity classifiers," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[36] Z. Ghahramani and K. A. Heller, "Bayesian sets," in *Proceedings of the 19th Conference on Neural Information Processing Systems*, 2005.

[37] S. Gupta, D. L. MacLean, J. Heer, and C. D. Manning, "Research and applications: Induced lexico-syntactic patterns improve information extraction from online medical forums," in *Journal of the American Medical Informatics Association*, 2014.

[38] E. Riloff, "Automatically generating extraction patterns from untagged text," in *Proceedings of the 1996 AAAI Conference on Artificial Intelligence*, 1996.

[39] P. P. Talukdar, J. Reisinger, M. Pasca, D. Ravichandran, R. Bhagat, and F. Pereira, "Weakly-supervised acquisition of labeled class instances using graph random walks," in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 2008.

[40] Y.-Y. Wang, R. Hoffmann, X. Li, and J. Szymanski, "Semi-supervised learning of semantic classes for query understanding: from the web and for the web," in *Proceedings of the 18th ACM International Conference on Information and Knowledge Management*, 2009.

[41] R. Balasubramanyan, B. B. Dalvi, and W. W. Cohen, "From topic models to semi-supervised learning: Biasing mixed-membership models to exploit topic-indicative features in entity clustering," in *Proceedings of 2013 Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2013.

[42] D. Lin and X. Wu, "Phrase clustering for discriminative learning," in *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*, 2009.

[43] T. McIntosh and J. R. Curran, "Weighted mutual exclusion bootstrapping for domain independent lexicon and template acquisition," in *Proceedings of the Australasian Language Technology Association Workshop 2008*, 2008.

[44] X. Ren, Y. Lv, K. Wang, and J. Han, "Comparative document analysis for large text corpora," in *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*, 2017.

[45] F. Chierichetti, R. Kumar, S. Pandey, and S. Vassilvitskii, "Finding the jaccard median," in *Proceedings of the 21st annual ACM-SIAM symposium on Discrete Algorithms*, 2010.

[46] P. Jindal and D. Roth, "Learning from negative examples in set-expansion," in *Proceedings of IEEE 11th International Conference on Data Mining*, 2011.

[47] J. R. Curran, T. Murphy, and B. Scholz, "Minimising semantic drift with mutual exclusion bootstrapping," in *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, 2007.

[48] X. Ling and D. S. Weld, "Fine-grained entity recognition," in *Proceedings of the 2012 AAAI Conference on Artificial Intelligence*, 2012.

[49] J. Liu, J. Shang, C. Wang, X. Ren, and J. Han, "Mining quality phrases from massive text corpora," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015.

[50] X. Ren, A. El-Kishky, C. Wang, F. Tao, C. R. Voss, and J. Han, "ClusType: Effective entity recognition and typing by relation phrase-based clustering," in *Proceedings of the 24th International Conference on World Wide Web*, 2015.

[51] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 27th Conference on Neural Information Processing Systems*, 2013.

[52] J. Tang, M. Qu, and Q. Mei, "PTE: Predictive text embedding through large-scale heterogeneous text networks," in *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[53] M. Thelen and E. Riloff, "A bootstrapping method for learning semantic lexicons using extraction pattern contexts," in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, 2002.

[54] W. Lin, R. Yangarber, and R. Grishman, "Bootstrapped learning of semantic classes from positive and negative examples," in *Proceedings of ICML-2003 Workshop on The Continuum from Labeled to Unlabeled Data*, 2003.

[55] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[56] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Proceedings of the 33rd Conference on Neural Information Processing Systems*, 2019.

[57] J. Mamou, O. Pereg, M. Wasserblat, A. Eirew, Y. Green, S. Guskin, P. Izsak, and D. Korat, "Term Set Expansion based NLP Architect by Intel AI Lab," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[58] P. Yu, Z. Huang, R. Rahimi, and J. D. Allan, "Corpus-based set expansion with lexical features and distributed representations," in *Proceedings of the 42nd International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2019.

[59] L. Yan, X. Han, L. Sun, and B. He, "Learning to bootstrap for entity set expansion," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

[60] S. Yang, L. Zou, Z. Wang, J. Yan, and J.-R. Wen, "Efficiently answering technical questions - a knowledge graph approach," in *Proceedings of the 2017 AAAI Conference on Artificial Intelligence*, 2017.

[61] W. Hua, Z. Wang, H. Wang, K. Zheng, and X. Zhou, "Understand short texts by harvesting and analyzing semantic knowledge," in *IEEE Transactions on Knowledge and Data Engineering*, 2017.

[62] Y. Zhang, A. Ahmed, V. Josifovski, and A. J. Smola, "Taxonomy discovery for personalized recommendation," in *Proceedings of the 7th ACM international conference on Web search and data mining*, 2014.

[63] C. Wang, M. Danilevsky, N. Desai, Y. Zhang, P. Nguyen, T. Taula, and J. Han, "A phrase mining framework for recursive construction of a topical hierarchy," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013.

[64] J. Shang, X. Zhang, L. Liu, S. Li, and J. Han, "NetTaxo: Automated topic taxonomy construction from text-rich network," in *Proceedings of the 2020 Web Conference*, 2020.

[65] D. Downey, C. Bhagavatula, and Y. Yang, "Efficient methods for inferring large sparse topic hierarchies," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, 2015.

[66] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proceedings of the 15th International Conference on Computational inguistics*, 1992.

[67] R. Snow, D. Jurafsky, and A. Y. Ng, "Learning syntactic patterns for automatic hypernym discovery," in *Proceedings of the 18th Conference on Neural Information Processing Systems*, 2004.

[68] A. Ritter, S. Soderland, and O. Etzioni, "What is this, anyway: Automatic hypernym discovery," in *Papers from the 2009 AAAI Spring Symposium: Learning by Reading and Learning to Read*, 2009.

[69] A. T. Luu, J. jae Kim, and S.-K. Ng, "Taxonomy construction using syntactic contextual evidence," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

[70] R. Navigli and P. Velardi, "Learning word-class lattices for definition and hypernym extraction," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010.

[71] N. Nakashole, G. Weikum, and F. M. Suchanek, "Patty: A taxonomy of relational patterns with semantic types," in *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing*, 2012.

[72] M. Jiang, J. Shang, T. Cassidy, X. Ren, L. M. Kaplan, T. P. Hanratty, and J. Han, "MetaPAD: Meta pattern discovery from massive text corpora," in *Proceedings of the 23th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[73] D. Lin, "An information-theoretic definition of similarity," in *Proceedings of the 15th International Conference on Machine Learning*, 1998.

[74] J. Weeds, D. J. Weir, and D. McCarthy, "Characterising measures of lexical distributional similarity," in *Proceedings of the 20th international conference on Computational Linguistics*, 2004.

[75] S. Roller, K. Erk, and G. Boleda, "Inclusive yet selective: Supervised distributional hypernymy detection," in *Proceedings of the 25th International Conference on Computational Linguistics*, 2014.

[76] M. G. Baroni, R. Bernardi, N.-Q. Do, and C. chieh Shan, "Entailment above the word level in distributional semantics," in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 2012.

[77] R. Fu, J. Guo, B. Qin, W. Che, H. Wang, and T. Liu, "Learning semantic hierarchies via word embeddings," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.

[78] J. Weeds, D. Clarke, J. Reffin, D. J. Weir, and B. Keller, "Learning to distinguish hypernyms and co-hyponyms," in *Proceedings of the 25th International Conference on Computational Linguistics*, 2014.

[79] A. T. Luu, Y. Tay, S. C. Hui, and S.-K. Ng, "Learning term embeddings for taxonomic relation identification using dynamic weighting neural network," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[80] L. E. Anke, J. Camacho-Collados, C. D. Bovi, and H. Saggion, "Supervised distributional hypernym discovery via domain adaptation," in *Proceedings of the 26th International Conference on Computational Linguistics*, 2016.

[81] Z. Kozareva and E. H. Hovy, "A semi-supervised method to learn and construct taxonomies using the web," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 2010.

[82] M. Bansal, D. Burkett, G. de Melo, and D. Klein, "Structured learning for taxonomy induction with belief propagation," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.

[83] A. Gupta, R. Lebret, H. Harkous, and K. Aberer, "Taxonomy induction using hypernym subsequences," in *Proceedings of the 26th ACM International Conference on Information and Knowledge Management*, 2017.

[84] D. Blei, T. Griffiths, M. I. Jordan, and J. Tenenbaum, "Hierarchical topic models and the nested chinese restaurant process," in *Proceedings of the 17th Conference on Neural Information Processing Systems*, 2003.

[85] S. Shin and I. Moon, "Guided HTM: Hierarchical topic model with dirichlet forest priors," *IEEE Transactions on Knowledge and Data Engineering*, 2017.

[86] X. Liu, Y. Song, S. Liu, and H. Wang, "Automatic taxonomy construction from keywords," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012.

[87] Y. Song, S. Liu, X. Liu, and H. Wang, "Automatic taxonomy construction from keywords via scalable bayesian rose trees," in *IEEE Transactions on Knowledge and Data Engineering*, 2015.

[88] C. Zhang, F. Tao, X. Chen, J. Shen, M. Jiang, B. M. Sadler, M. T. Vanni, and J. Han, "Taxogen: Constructing topical concept taxonomy by adaptive term embedding and clustering," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.

[89] J. Chang, J. L. Boyd-Graber, S. Gerrish, C. Wang, and D. Blei, "Reading tea leaves: How humans interpret topic models," in *Proceedings of the 23rd Conference on Neural Information Processing Systems*, 2009.

[90] M. Qu, X. Ren, Y. Zhang, and J. Han, "Weakly-supervised relation extraction by pattern-enhanced embedding learning," in *Proceedings of the 27th International Conference on World Wide Web*, 2018.

[91] D. Zeng, K. Liu, Y. Chen, and J. Zhao, "Distant supervision for relation extraction via piecewise convolutional neural networks," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

[92] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*, 2009.

[93] S. Riedel, L. Yao, A. McCallum, and B. M. Marlin, "Relation extraction with matrix factorization and universal schemas," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013.

[94] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

[95] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, 2015.

[96] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han, "Automated phrase mining from massive text corpora," in *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[97] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proceedings of the 17th Conference on Neural Information Processing Systems*, 2003.

[98] G. Bordea, P. Buitelaar, S. Faralli, and R. Navigli, "Semeval-2015 Task 17: Taxonomy Extraction Evaluation (TExEval)," in *Proceedings of the 9th International Workshop on Semantic Evaluation*, 2015.

[99] G. Bordea, E. Lefever, and P. Buitelaar, "Semeval-2016 Task 13: Taxonomy extraction evaluation (TExEval-2)," in *Proceedings of the 10th International Workshop on Semantic Evaluation*, 2016.

[100] Y. Meng, J. Huang, G. Wang, C. Zhang, H. Zhuang, L. M. Kaplan, and J. Han, "Spherical text embedding," in *Proceedings of the 33rd Conference on Neural Information Processing Systems*, 2019.

[101] Y. Meng, J. Huang, G. Wang, Z. Wang, C. Zhang, Y. Zhang, and J. Han, "Discriminative topic mining via category-name guided text embedding," *Proceedings of the 2020 Web Conference*, 2020.

[102] G. H. Yang, "Constructing task-specific taxonomies for document collection browsing," in *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing*, 2012.

[103] J. Huang, Z. Ren, W. X. Zhao, G. He, J.-R. Wen, and D. Dong, "Taxonomy-aware multi-hop reasoning networks for sequential recommendation," in *Proceedings of the 12ND ACM international conference on Web search and data mining*, 2019.

[104] B. W. Liu, W. Guo, D. Niu, C. Wang, S.-Z. Xu, J. Lin, K. Lai, and Y. W. Xu, "A user-centered concept mining system for query and document understanding at tencent," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[105] D. Jurgens and M. T. Pilehvar, "Reserating the awesometastic: An automatic extension of the WordNet taxonomy for novel terms," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[106] D. Jurgens and M. T. Pilehvar, "SemEval-2016 task 14: Semantic taxonomy enrichment," in *Proceedings of the 10th International Workshop on Semantic Evaluation*, 2016.

[107] M. S. Schlichtkrull and H. M. Alonso, "MSejrKu at SemEval-2016 task 14: Taxonomy enrichment by evidence ranking," in *Proceedings of the 10th International Workshop on Semantic Evaluation*, 2016.

[108] Q. Zeng, J. Lin, W. Yu, J. Cleland-Huang, and M. Jiang, "Enhancing taxonomy completion with concept generation via fusing relational representations," in *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2021.

[109] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Procedings of the 5th International Conference on Learning Representations*, 2017.

[110] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[111] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[112] A. Toral, R. Muñoz, and M. Monachini, "Named entity WordNet," in *Proceedings of the 6th international conference on language resources and evaluation*, 2008.

[113] L. Bentivogli, A. A. Bocco, and E. Pianta, "ArchiWordNet: Integrating WordNet with domain-specific knowledge," in *Proceedings of the 2nd International Global Wordnet Conference*, 2003.

[114] C. Fellbaum, U. Hahn, and B. D. Smith, "Towards new information resources for public health - from WordNet to MedicalWordNet," *Journal of biomedical informatics*, 2006.

[115] L. E. Anke, J. Camacho-Collados, S. Rodríguez-Fernández, H. Saggion, and L. Wanner, "Extending WordNet with fine-grained collocational information via supervised distributional learning," in *Proceedings of the 26th International Conference on Computational Linguistics*, 2016.

[116] J. Wang, C. Kang, Y. Chang, and J. Han, "A hierarchical dirichlet model for taxonomy expansion for search engines," in *Proceedings of the 23rd International Conference on World Wide Web*, 2014.

[117] V. Plachouras, F. Petroni, T. Nugent, and J. L. Leidner, "A comparison of two paraphrase models for taxonomy augmentation," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

[118] N. Vedula, P. K. Nicholson, D. Ajwani, S. Dutta, A. Sala, and S. Parthasarathy, "Enriching taxonomies with functional domain knowledge," in *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2018.

[119] R. Aly, S. Acharya, A. Ossa, A. Köhn, C. Biemann, and A. Panchenko, "Every child should have parents: A taxonomy refinement algorithm based on hyperbolic term embeddings," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[120] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st Conference on Neural Information Processing Systems*, 2017.

[121] J. J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," in *Procedings of the 6th International Conference on Learning Representations*, 2018.

[122] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proceedings of the 32nd Conference on Neural Information Processing Systems*, 2018.

[123] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the 2018 AAAI Conference on Artificial Intelligence*, 2018.

[124] J. B. Lee, R. A. Rossi, and X. Kong, "Graph classification using structural attention," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.

[125] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[126] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. W. Battaglia, "Learning deep generative models of graphs," in *Procedings of the 6th International Conference on Learning Representations*, 2018.

[127] W. Jin, R. Barzilay, and T. S. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," in *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[128] J. You, B. Liu, Z. Ying, V. S. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," in *Proceedings of the 32nd Conference on Neural Information Processing Systems*, 2018.

[129] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proceedings of 31st Conference on Neural Information Processing Systems*, 2017.

[130] Z. Shen, H. Ma, and K. Wang, "A web-scale system for scientific knowledge exploration," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

[131] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[132] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.

[133] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

[134] R. You, S. Dai, Z. Zhang, H. Mamitsuka, and S. Zhu, "Attentionxml: Extreme multi-label text classification with multi-label attention based recurrent neural networks," in *Proceedings of the 33rd Conference on Neural Information Processing Systems*, 2019.

[135] W. Huang, E. Chen, Q. Liu, Y. Chen, Z. Huang, Y. Liu, Z. Zhao, D. Zhang, and S. Wang, "Hierarchical multi-label text classification: An attention-based recurrent network approach," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019.

[136] S. Gururangan, T. Dang, D. Card, and N. A. Smith, "Variational pretraining for semi-supervised text classification," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[137] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Proceedings of the 33rd Conference on Neural Information Processing Systems*, 2019.

[138] Z. Zeng, W. Zhou, X. Liu, and Y. Song, "A variational approach to weakly supervised document-level multi-aspect sentiment classification," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[139] D. Mekala and J. Shang, "Contextualized weak supervision for text classification," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[140] Y. Meng, Y. Zhang, J. Huang, C. Xiong, H. Ji, C. Zhang, and J. Han, "Text classification using label names only: A language model self-training approach," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

[141] Z. Wang, D. Mekala, and J. Shang, "X-class: Text classification with extremely weak supervision," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

[142] W. Yin, J. Hay, and D. Roth, "Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

[143] M.-W. Chang, L.-A. Ratinov, D. Roth, and V. Srikumar, "Importance of semantic representation: Dataless classification," in *Proceedings of the 2008 AAAI Conference on Artificial Intelligence*, 2008.

[144] Y. Song and D. Roth, "On dataless hierarchical text classification," in *Proceedings of the 2014 AAAI Conference on Artificial Intelligence*, 2014.

[145] K. Li, H. Zha, Y. Su, and X. Yan, "Unsupervised neural categorization for scientific publications," in *Proceedings of the 2018 SIAM International Conference on Data Mining*, 2018.

[146] K. Li, S. Li, S. Yavuz, H. Zha, Y. Su, and X. Yan, "Hiercon: Hierarchical organization of technical documents based on concepts," in *Proceedings of the 19th IEEE International Conference on Data Mining*, 2019.

[147] J. Nam, E. L. Mencía, and J. Fürnkranz, "All-in text: Learning document, label, and word representations jointly," in *Proceedings of the 2016 AAAI Conference on Artificial Intelligence*, 2016.

[148] A. Rios and R. Kavuluru, "Few-shot and zero-shot multi-label learning for structured label spaces," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[149] S. Srivastava, I. Labutov, and T. M. Mitchell, "Zero-shot learning of classifiers from natural language quantification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

[150] S. Banerjee, C. Akkaya, F. Perez-Sorrosal, and K. Tsioutsiouliklis, "Hierarchical transfer learning for multi-label text classification," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[151] T. Liu, Y. Yang, H. Wan, H. Zeng, Z. Chen, and W. Ma, "Support vector machines classification with a very large-scale taxonomy," in *Proceedings of the 11st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2005.

[152] J. Wehrmann, R. Cerri, and R. C. Barros, "Hierarchical multi-label classification networks," in *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[153] S. Gopal and Y. Yang, "Recursive regularization for large-scale classification with hierarchical and graphical dependencies," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013.

[154] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang, "Large-scale hierarchical text classification with recursively regularized deep graph-cnn," *Proceedings of the 2018 World Wide Web Conference*, 2018.

[155] J. Zhou, C. Ma, D. Long, G. Xu, N. Ding, H. Zhang, P. Xie, and G. Liu, "Hierarchy-aware global model for hierarchical text classification," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[156] J. Xie, R. B. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[157] J. J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text," in *Proceedings of the 7th ACM conference on Recommender Systems*, 2013.

[158] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer et al., "DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia," *Semantic web*, vol. 6, no. 2, pp. 167–195, 2015.

[159] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[160] H. Xiao, X. Liu, and Y. Song, "Efficient path prediction for semi-supervised and weakly supervised hierarchical text classification," in *Proceedings of the 2019 Web Conference*, 2019.

[161] H. Jain, Y. Prabhu, and M. Varma, "Extreme multi-label loss functions for recommendation, tagging, ranking, and other missing label applications," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[162] C. F. Guo, A. Mousavi, X. Wu, D. N. Holtmann-Rice, S. Kale, S. J. Reddi, and S. Kumar, "Breaking the glass ceiling for embedding-based classifiers for large output spaces," in *Proceedings of the 32nd Conference on Neural Information Processing Systems*, 2019.

[163] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, 2016.

[164] D. Ahn, "The stages of event extraction," in *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, 2006.

[165] H. Ji and R. Grishman, "Refining event extraction through cross-document inference," in *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, 2008.

[166] X. Du and C. Cardie, "Event extraction by answering (almost) natural questions," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

[167] S. Li, H. Ji, and J. Han, "Document-level event argument extraction by conditional generation," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

[168] S. Sekine, "On-demand information extraction," in *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, 2006.

[169] N. Chambers and D. Jurafsky, "Template-based information extraction without the templates," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, 2011.

[170] N. Chambers, "Event schema induction with a probabilistic entity-driven model," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013.

[171] J. C. K. Cheung, H. Poon, and L. Vanderwende, "Probabilistic frame induction," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013.

[172] K.-H. Nguyen, X. Tannier, O. Ferret, and R. Besançon, "Generative event schema induction with entity disambiguation," in *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics*, 2015.

[173] L. Huang, T. Cassidy, X. Feng, H. Ji, C. R. Voss, J. Han, and A. Sil, "Liberal event extraction and event schema induction," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[174] C. F. Baker, C. J. Fillmore, and J. B. Lowe, "The Berkeley FrameNet project," in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistic*, 1998.

[175] K. Schuler and M. Palmer, *VerbNet: A Broad-coverage, Comprehensive Verb Lexicon*. University of Pennsylvania, 2005.

[176] M. Palmer, P. R. Kingsbury, and D. Gildea, "The proposition bank: An annotated corpus of semantic roles," in *Computational Linguistics*, 2005.

[177] L. Huang, H. Ji, K. Cho, and C. R. Voss, "Zero-shot transfer learning for event extraction," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

[178] L. Huang and H. Ji, "Semi-supervised new event type induction and event detection," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

[179] D. J. Allerton, *Essentials of grammatical theory: A consensus view of syntax and morphology*. Routledge, 1979.

[180] J. Shen, Y. Zhang, H. Ji, and J. Han, "Corpus-based open-domain event type induction," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

[181] E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel, "OntoNotes: The 90% solution," in *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2006.

[182] Z. Song, A. Bies, S. Strassel, T. Riese, J. Mott, J. Ellis, J. Wright, S. Kulick, N. Ryant, and X. Ma, "From light to rich ERE: Annotation of entities, relations, and events," in *Proceedings of the 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, 2015.

[183] N. Chinchor, L. Hirschman, and D. D. Lewis, "Evaluating message understanding systems: An analysis of the third message understanding conference (MUC-3)," in *Computational Linguistics*, 1993.

[184] H. L. Chieu, H. T. Ng, and Y. K. Lee, "Closing the gap: Learning-based information extraction rivaling knowledge-engineering methods," in *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 2003.

[185] R. C. Bunescu and R. J. Mooney, "Collective information extraction with relational markov networks," in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 2004.

[186] M. Li, Q. Zeng, Y. Lin, K. Cho, H. Ji, J. May, N. Chambers, and C. R. Voss, "Connecting the dots: Event graph schema induction with path language modeling," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

[187] L. Sha, S. Li, B. Chang, and Z. Sui, "Joint learning templates and slots for event schema induction," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

[188] X. Liu, H. Huang, and Y. Zhang, "Open domain event extraction using neural latent variable models," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[189] Q. Yuan, X. Ren, W. He, C. Zhang, X. Geng, L. Huang, H. Ji, C.-Y. Lin, and J. Han, "Open-schema event profiling for massive news corpora," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018.

[190] O. Bronstein, I. Dagan, Q. Li, H. Ji, and A. Frank, "Seed-based event trigger labeling: How far can event descriptions get us?" in *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics*, 2015.

[191] J. Ferguson, C. Lockard, D. S. Weld, and H. Hajishirzi, "Semi-supervised event extraction with paraphrase clusters," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

[192] Y. S. Chan, J. Fasching, H. Qiu, and B. Min, "Rapid customization for event extraction," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[193] R. Wang, D. Zhou, and Y. He, "Open event extraction from online text using a generative adversarial network," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

[194] G. G. Corbett, N. M. Fraser, and S. McGlashan, *Heads in grammatical theory.* Cambridge University Press, 1993.

[195] T. J. O'Gorman, K. Wright-Bettner, and M. Palmer, "Richer event description: Integrating event coreference with temporal, causal and bridging annotation," in *Proceedings of the 2nd Workshop on Computing News Storylines*, 2016.

[196] Z. Zhong and H. T. Ng, "It makes sense: A wide-coverage word sense disambiguation system for free text," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2010.

[197] G. A. Miller, "WordNet: A lexical database for English," in *Communications of the ACM*, 1995.

[198] J. Shen, W. Qiu, J. Shang, M. Vanni, X. Ren, and J. Han, "SynSetExpan: An iterative framework for joint entity set expansion and synonym discovery," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

[199] W. Webber, A. Moffat, and J. Zobel, "A similarity measure for indefinite rankings," in *ACM Transactions on Information Systems*, 2010.

[200] R. Navigli, J. Camacho-Collados, and A. Raganato, "Word sense disambiguation: A unified evaluation framework and empirical comparison," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics*, 2017.

[201] Y. Meng, Y. Zhang, J. Huang, Y. Zhang, C. Zhang, and J. Han, "Hierarchical topic mining via joint spherical tree and text embedding," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.

[202] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra, "Clustering on the unit hypersphere using von mises-fisher distributions," in *Journal of Machine Learning Research*, 2005.

[203] G. E. Hinton and R. S. Zemel, "AutoEncoders, minimum description length and helmholtz free energy," in *Proceedings of the 7th Conference on Neural Information Processing Systems*, 1993.

[204] Y. Lin, H. Ji, F. Huang, and L. Wu, "A joint neural model for information extraction with global features," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[205] D. Ustalov, A. Panchenko, A. Kutuzov, C. Biemann, and S. P. Ponzetto, "Unsupervised semantic frame induction using triclustering," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

[206] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, G. Louppe, P. Prettenhofer, R. Weiss, R. J. Weiss, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, 2011.

[207] H. Xiao and Y. Song, "Manifold alignment across geometric spaces for knowledge base representation learning," in *Proceedings of the 3rd Conference on Automated Knowledge Base Construction*, 2021.