EXPLOITING LANGUAGE MODELS FOR ANNOTATION-EFFICIENT KNOWLEDGE DISCOVERY

BY

JIAXIN HUANG

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, 2023

Urbana, Illinois

Doctoral Committee:

Professor Jiawei Han, Chair
Professor Chengxiang Zhai
Professor Tarek Abdelzaher
Dr. Jianfeng Gao, Microsoft Research

ABSTRACT

With tremendous amounts of texts across the Internet nowadays, it is incredibly difficult for people to manually seek for valuable knowledge from massive corpora, thus automatic knowledge acquisition systems are becoming highly desirable. Various text mining techniques are built for machines to perform text retrieval, concept understanding, commonsense reasoning, and question answering to solve for downstream tasks proposed by practitioners from different areas.

Existing intelligent systems are mostly based on deep learning models that generally require enormous amounts of annotations in downstream domains, which is expensive and time-consuming. Furthermore, they tend to assume a pre-defined task and label space, and are hard to handle unseen tasks that contain new emerging concepts in new domains. For more challenging knowledge utilization tasks such as commonsense reasoning, simple annotations of the final answer is not sufficient to reflect the complex reasoning process for performing the task.

My research aims to design text mining approaches for weakly-supervised knowledge extraction and utilization on domain-specific corpus, by leveraging the strong representation and generative power of pre-trained language models. Specifically, my work can be divided into the following three parts:

1. **Seed-Guided Hierarchical Concept Organization.** I build a systematic framework [79, 78, 125] that takes user-given seed hierarchy, and **constructs task-specific concept ontology** from domain text corpora. Traditional ontologies often fall short in capturing user-specific interests and relations, leading to potential irrelevance in specialized domains. I introduce a fully automated approach to address semantic drift in entity set expansion and present a new framework for seed-guided topical taxonomy construction.
2. **Fine-Grained Entity Extraction.** Extracting key information from text corpora serve as foundational steps in text mining applications. I first provide a comprehensive overview [76] of the few-shot learning methodologies for Named Entity Recognition and identify useful techniques, and then introduce a novel approach [77] for Fine-Grained Entity Typing that harnesses the representation and generation capabilities of PLMs.
3. **Knowledge-Guided Commonsense Reasoning.** Entity knowledge can be used to enhance more complex user-oriented tasks such as commonsense reasoning that relies on dynamically utilizing the static knowledge. I design methods [75] to bridge the gap

between the inherent capabilities of Large Language Models (LLMs) and the human brain's metacognitive processes, offering insights into how LLMs can self-enhance their reasoning abilities without the need for supervised data.

To my family for their love and support.

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my esteemed advisor, Professor Jiawei Han, for his unwavering support, insightful guidance and constant encouragement throughout the course of my PhD journey. Professor Han's professionalism and commitment to research excellence are unparalleled. Five years ago, I cannot even believe that Professor Han took me as one of his PhD student. Since then, Professor Han has always encouraged me to pursue interesting research problems, address real-world challenges, and to never lose sight of the bigger picture. Professor Han is always accessible whenever we want to discuss with him, and during our brainstorming discussions, his vast experience and wisdom shared from past students' stories have continually inspired me. I still remember that during my first year of PhD, when our paper on category-guided word embedding faced multiple rejections, Professor Han strongly believes in the potential of our work, and kept reminding us not to be discouraged, and should keep pushing the direction until people realize its power. Finally, our paper is accepted by NeurIPS and we also publish a series of follow-up works. Beyond concrete research, Professor Han has been instrumental in shaping my academic trajectory, encouraging me to seize opportunities like the Microsoft Research PhD Fellowship and guiding me through the intricate process of faculty applications. Professor Han is a very nice and warm person in daily life but has a high standard on doing research, where I am deeply impressed and will continue to work hard to strive for excellence.

I would like to extend my gratitude to the other members in my Thesis Committee: Professor Chengxiang Zhai, Professor Tarek Abdelzaher, and Dr. Jianfeng Gao. Their invaluable feedback and constructive critiques have greatly enhanced the quality of my thesis. Despite their demanding schedules, they still dedicate a lot to my thesis, take consecutive hours to attend my thesis presentation, carefully read my thesis, and provide very detailed feedbacks.

I wish to express my profound gratitude to the Defense Advanced Research Projects Agency, the National Science Foundation, the Army Research Laboratory, and Microsoft Research for their invaluable support throughout my Ph.D. journey. Their generous funding and support have been crucial to my research outcomes and this thesis.

A special thank should be given to all my coauthors, for their continued support during our research collaborations. I feel so lucky to have the opportunity to work with such a diverse and talented group. They inspire me a lot on research ideas, broaden my research scope, and offer help to me whenever I need. Our brainstorming discussions, late-night experiments, and

paper writing together have been truly memorable experiences. I am deeply grateful to these lovely people: Yu Meng, Yu Zhang, Chunyuan Li, Le Hou, Siru Ouyang, Suyu Ge, Yiqing Xie, Hongkun Yu, Xuezhi Wang, Jiaming Shen, Shobana Balakrishnan, Weizhu Chen, Shane Gu, Fang Guo, Damien Jose, Lance Kaplan, Martin Michalski, Baolin Peng, Jingbo Shang, Krishan Subudhi, Xuan Wang, Zihan Wang, Guangyuan Wang, Yuexin Wu, Chenyan Xiong, Frank Xu, Yunyi Zhang, Chao Zhang, and Honglei Zhuang.

I would also like to thank my colleagus in Data Mining Group for their generous support, which has always brought enlightening moments to me. We have had chats and spring outings together, fighting for the same conferenece deadline together, and had hotpots together on cold winter nights of Urbana-Champaign. They have provided important advice and suggestions for me, from how to read a paper, to building connections and hunting jobs. I would like to sincerely thank these people: Shivam Agarwal, Xiusi Chen, Xiaotao Gu, Huan Gui, Yizhu Jiao, Bowen Jin, Priyanka Kargupta, Sha Li, Qi Li, Liyuan Liu, Ruiliang Lyu, Yuning Mao, Chanyong Park, Wenda Qiu, Xiang Ren, Yanzhen Shen, Yu Shi, Xiangchen Song, Fangbo Tao, Ellen Wu, Carl Yang, Susik Yoon, Xinyang Zhang, Shi Zhi, Ming Zhong, Xianrui Zhong, Sizhe Zhou, Wanzheng Zhu, Qi Zhu, Jinfeng Xiao.

My deepest appreciation goes to my family. Their constant love has been always been my strength to face difficulties. Their belief and understanding supported me to proceed during challenging times, and their care and joy in my moments of achievements have led to many cherished memories.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Nowadays, the proliferation of digital content has transformed the way we access and disseminate information. With overwhelming amounts of textual data being created online everyday, there is an increasing demand for intelligent systems to assist people by efficiently extract, organize, and utilize knowledge from vast textual data. Existing intelligent systems are often constrained by the need for extensive annotations along with sophisticated pre-defined task and label space as well as the complexity of tasks like commonsense reasoning. Therefore, it is crucial to develop few-shot or even zero-shot approaches for knowledge extraction tasks, especially in domain-specific areas where annotations by experts are limited.

Recently, pre-trained language models (*e.g.,* BERT-base [37] model with 110M parameters) have shown that they are powerful at generalizing to various NLP tasks, even under low-resource settings. This is because these models gain strong text representative power during self-supervised pre-training on large corpus. Moreover, people have been pursuing even larger-sized language models, such as GPT-3 [13] with 175B parameters and PaLM [24] with 540B parameters. It has been observed that as the model size scales up, it has stronger generalization ability and performs well on complex tasks that smaller language models cannot fulfill [194], such as multi-step reasoning, program execution, and model calibration (the ability to predict whether its answer is correct or not).

Though these PLMs or even LLMs can reach remarkable performance on many difficult tasks, they still suffer from several **drawbacks**: (1) Language models are known to perform well on knowledge pieces related to popular/frequent entities, but suffer from long-tailed entities which only appear a few times in the pre-trained corpus [216]. Therefore, they could generate hallucinations (factual errors or inconsistent logic) in their responses. (2) Language model pre-training requires a large amount of text data, which could raise copyright and privacy issues. Public users may not want to generate a piece of text that could be identified as private data. On the other hand, some organizations wish to use language models with their own private data (*e.g.,* a domain-specific corpus). (3) It would be hard to fix errors made by language models since it is not as interpretable as explicit knowledge forms, and continuing the fine-tuning process may lead to catastrophic forgetting on preivous tasks.

My research provides a solution that leverage the advantages of both **language models** and the long-tailed knowledge in **domain-specific corpus** for knowledge extraction. For example, if some researchers have access to a domain-specific corpus and want to extract useful knowledge from it, my approaches can satisfy their needs by integrating both the LM techniques and long-tailed knowledge inherited in the corpus, such as by regularizing the

LM space on domain-specific concepts, as well as using the regularized LMs to self-generate domain-specific augmented data.

In my thesis, I mainly discuss three representative knowledge discovery applications: concept ontology construction, named entity recognition, and multi-step reasoning. The contents of this thesis can be organized and illustrated in Fig. 1.1.

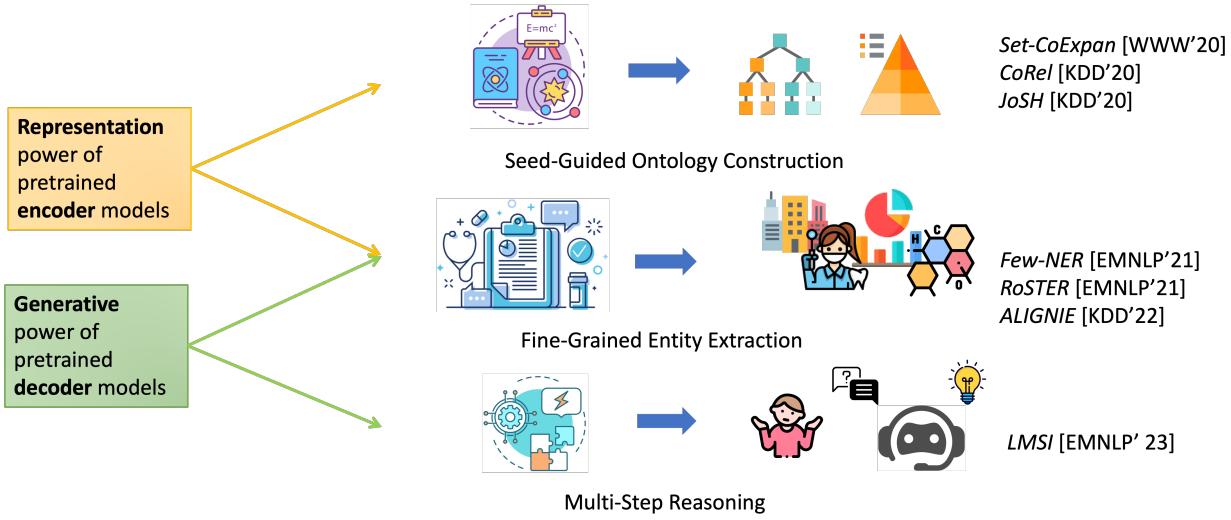


Figure 1.1: Overview of my research framework.

The first part (Chapter 2) takes user-given seed entities or seed ontology, expand them both according to the entity type and entity relations, and finally forms a **task-specific concept ontology**. Ontology is a fundamental form of knowledge representation to organize concepts, entities and documents in a hierarchical way, from coarse-grained levels to fine-grained levels. To avoid overlapping entities from similar categories, controllable expansion is conducted by learning a concept-regularized embedding space. This is done by jointly learning the vocabulary and user-interested topics in the semantic space. To capture and transfer the potential relations in the seed ontology, I develop methods [79, 78, 125] that utilize generic linguistic knowledge preserved in Pre-trained Language Models into hierarchical inductive bias among entities.

The second part (Chapter 3) develops few-shot methods for **knowledge acquisition in entity level** [76, 77], by studying on Named Entity Recognition and Fine-Grained Entity Typing. These tasks serve as the important first component for downstream tasks such as information extraction [157], information retrieval [60], question answering [135], task-oriented dialogues [145, 50] and other language understanding applications [137, 163]. I will introduce two studies on Named Entity Recognition (NER) and Fine-grained Entity Typing (FET), especially in scenarios with limited labeled data. I will first provide a comprehensive

overview of the few-shot learning methodologies for NER and identify useful techniques, and then introduce novel approaches for FET that harness the representation and generation capabilities of PLMs.

The third part (Chapter 4) focuses on more complex user-oriented tasks such as **commonsense reasoning** that relies on dynamically utilizing the static knowledge. My research aims to outputting a generalized model that can answer various styles of reasoning questions given a few examples of in-domain questions and answers with explanations. Specifically, I will introduce a novel approach [75] aiming to bridge the gap between the inherent capabilities of language models and the human brain's metacognitive processes, offering insights into how language models can self-enhance their reasoning abilities without the need for supervised data.

The subsequent chapters provide a detailed exploration of these methodologies, from seed-guided hierarchical concept organization to fine-grained entity extraction and knowledge-guided commonsense reasoning, offering a holistic view and pave the way for more adaptable, label-efficient, and user-centric knowledge acquisition and utilization systems.

CHAPTER 2: SEED-GUIDED HIERARCHICAL CONCEPT ORGANIZATION

Ontology is a fundamental form of knowledge representation to organize concepts, entities and documents in a hierarchical way, from coarse-grained levels to fine-grained levels. For downstream knowledge-rich applications such as topic discovery, personalized recommendation and question answering, an ontology can serve as a hierarchical label space to pinpoint groups of text segments to a related concept or label. However, traditional ontologies often fall short in capturing user-specific interests and relations, leading to potential irrelevance in specialized domains. Such challenges arise from the uncontrolled expansion of ontologies and concepts without proper guidance, leading to the inclusion of cross-category entities. Moreover, existing ontology construction techniques typically requires heavy human efforts. In this chapter, we focus on these challenges and propose novel approaches to make ontologies more adaptable and user-centric. Specifically, we introduce a fully automated approach to address semantic drift in entity set expansion and present a new framework for seed-guided topical ontology construction.

2.1 OVERVIEW

Ontology is an essential form of knowledge representation and plays an important role in a wide range of applications [223, 207, 129]. An ontology constructed from a large corpus organizes a set of concepts into a hierarchy, making it clear for people to understand relations between concepts. While most ontology construction methods focus on organizing hypernym-hyponym pairs into tree structures, these "universal" taxonomies might not meet specific user needs. For instance, a generic ontology might be overwhelmed with irrelevant terms or have nodes represented by single words, limiting semantic understanding. Our goal is to construct user-guided topical ontology in an automated way. To construct or expand the tree structure of an ontology, can be mainly divided into two dimensions: width expansion and depth expansion. Therefore, entity set expansion can serve as a first step of ontology construction, where parallel nodes on the same level are expanded. A second step is to extract potential user-interested relations on the seed ontology, before expanding it to a more complete ontology. We will discuss techniques for these two steps sequentially.

Existing set expansion methods that rely on a large corpus typically bootstrap the initial seeds by refining the context feature pool and candidate term pool in an iterative manner. Though achieving reasonably good results, they still suffer from the problems of semantic drifting and entity intrusion in the expansion process. Typical errors come from closely

related entities of different granularity or semantic types. As shown in Figure 2.1: despite the difference in granularity of "Canada" as a country and "Ontario" as a Canadian province, they share some local contexts. These shared contexts, when introduced into the context feature pool, result in cross-category expansion and harm the final output continuously in the iterative process.

Such kind of semantic drift is caused by the uncontrolled expansion without guidance. Some previous studies [106, 179, 86] have explored to incorporate external knowledge in set expansion using either implicit supervision from other queries or human given negative examples. The applicability of these methods is often hindered by their prerequisites as they are not adaptable to specialized domains with very few experts.

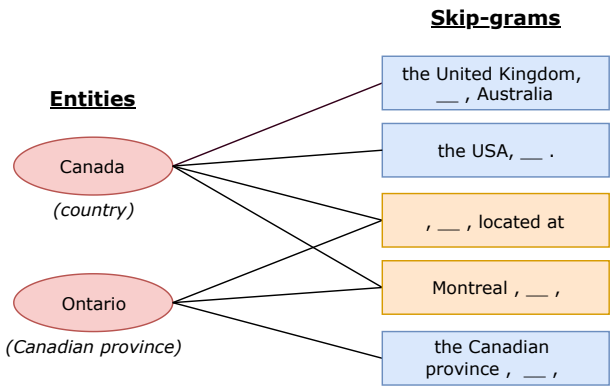


Figure 2.1: Examples of shared skip-grams between country and province/states entities.

To tackle the problem of semantic drift in set expansion, I propose a fully automated approach Set-CoExpan without negative sets or assumptions explicitly provided by human experts. This is done by capturing parallel relations between entities: word embedding has shown effective in capturing certain relations between entities, such as the parallelism of $v(Berlin) - v(Germany)$ and $v(Paris) - v(France)$ in the embedding space, which can also be used to distinguish *City* from *Country*. Recognizing relationships between entities from different semantic classes can help in set co-expansion. This co-expansion uses signals from related classes to guide the expansion direction, avoiding overlapping of entities from related classes by leveraging mutually exclusive signals. Specifically, the method contains an auxiliary sets generation module that extracts entities that share certain relations to thus different from the target semantic class, and a co-expansion module that utilizes the mutual exclusive signals to extract features with the most discriminative power to expand multiple sets simultaneously. Comprehensive experiments prove the effectiveness of Set-CoExpan and justify its generalizability.

As for topical ontology construction, I focus on a weakly-supervised setting, where the user

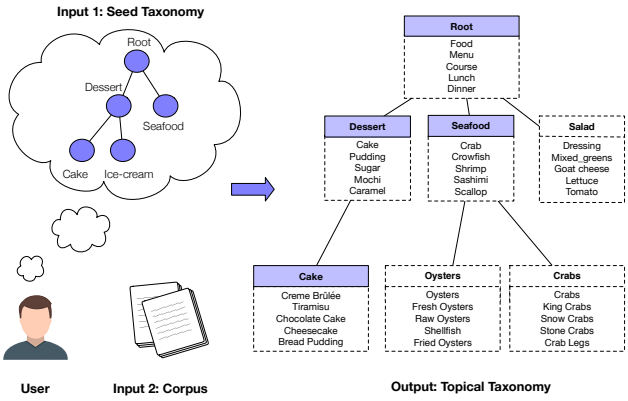


Figure 2.2: Seed-guided topical ontology construction. User inputs a partial ontology, and CoRel extracts a more complete topical ontology based on user-interested aspect and relation, with each node represented by a cluster of words.

provides a seed ontology as guidance, and a more complete topical ontology is generated from text corpus, with each node represented by a cluster of terms (topics). As shown in Figure 2.2, a user provides a seed ontology and wants to generate a more complete food ontology from a given corpus. Such a more complete topical ontology can be hopefully constructed by expanding various types of food both in width and depth, with a cluster of descriptive terms for each concept node as a topic.

I proposed a novel framework CoRel, which consists of two modules: a relation transferring module that passes the user-interested relation along multiple paths in different directions for ontology structure completion; and a concept learning module that enriches the semantics for a ontology of words by extracting distinctive terms. Comprehensive experiments on real-world data with qualitative and quantitative studies that prove the effectiveness of CoRel.

2.2 RELATED WORK

Corpus-Based Set Expansion. Previous studies [46, 61, 62, 165, 122] on corpus-based set extraction or expansion bootstrap the initials seeds in an iterative way by refining the context feature pool and candidate pool. Specifically, context features narrowed down the candidates by requiring co-occurrence between candidates and patterns (like n-gram or skip-gram); candidates refine context features by providing supervision to learn the most representative features commonly shared by expanded seeds. SetExpan [165], which has the most similar setting with our work, uses a context feature selection module to select denoised context features and a rank ensemble module to combine skip-gram patterns with distributional similarity. setExpander [122] captures distributional similarity on five different

context types and trains a classifier to combine multiple contexts. The classifier is pre-trained on a labeled dataset with seeds and candidate terms. There are also other related methods that are designed for different principles. CaSE[214] pursues efficiency by incorporating lexical patterns and distributional similarity in candidate scoring function to make a one-time ranking result. Egoset [158] deals with multifaceted expansion that allows for each seed term to generate multiple sets belonging to different senses. [6] focuses on singleton expansion (i.e.. only one seed in the initial seed set). Above methods only pay attention to the user given query words, suffering from possible semantic drift while expanding the seed set. Our proposed method in this chapter focuses on this issue by expanding multiple exclusive sets simultaneously for distinction across different categories.

Unsupervised Ontology Construction. Instance-based ontology construction algorithms perform a two-step approach: Hypernym-hyponym pairs are first extracted from a corpus and then organized into a tree structure. Hearst patterns [68] like "NP such as NP" are used to acquire parent-child pairs that satisfy the "is-a" relation. Later, researchers design more lexical patterns [140, 138] or extract such patterns automatically [173, 170] in a bootstrapping method. These pattern-based methods suffer from low recall due to the diversity of expressions. Distributional methods alleviate the problem of sparsity by representing each word as a low-dimensional vector to capture their semantic similarity. There exist approaches [192, 17] inspired by Distributional Inclusion Hypothesis [227] (the context of a hyponym should be the subset of that of a hypernym) to detect hypernym-hyponym pairs without supervision. As another line of work, clustering-based ontology construction methods first learn a representation space for terms, then perform clustering to separate terms into different topics by different measures [26, 114, 187, 205]. TaxoGen [220] finds fine-grained topics by spherical clustering and local-corpus embedding. Hierarchical topic modeling algorithms [11, 132] are comparable to these methods, since they organize terms to form a ontology of topics, each represented by a word distribution.

Seed-Guided Ontology Construction. For seed-guided ontology construction, HiExpan [166] integrates a two-step approach into a tree expansion process by width and depth expansion of the original seed ontology. Specifically, for width expansion that adds sibling nodes to those sharing the same parent, the method uses a set expansion algorithm [165] that leverages skip-gram features to calculate similarity between terms. For depth expansion that attaches children nodes to new node (e.g., attaching "oyster" to "seafood"), they use word analogy [131] to capture relations between parent-child pairs. Our proposed method in this chapter integrates the advantages of both clustering-based and instance-based methods, by transferring relations within node pairs and enriching concept semantics with word embedding

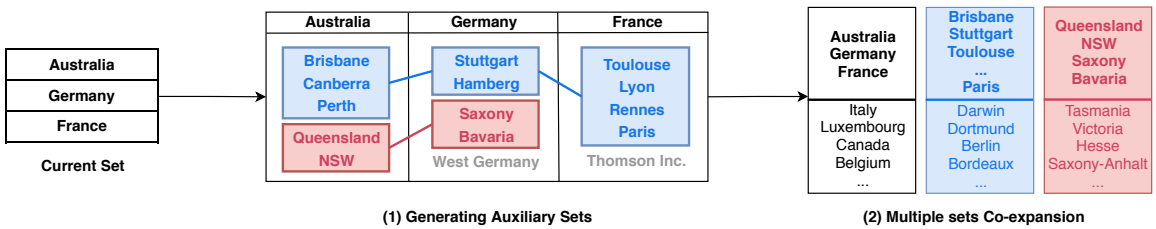


Figure 2.3: Workflow of Set-CoExpan in one iteration: For user-input country names, related terms such as provinces and cities are retrieved and clustered into auxiliary sets. Multiple sets are then co-expanded by extracting discriminative context features.

clustering.

2.3 DISCOVERING PARALLEL INSTANCES THROUGH SET EXPANSION

In this section, we introduce our proposed method Set-CoExpan for entity set expansion, which aims to enrich same-type entities, and could be a crucial first step for ontology construction. For example, given "*United States*", "*Canada*", and "*China*", set expansion algorithms would extend the set by terms such as "*France*", "*Germany*" and "*Australia*", which also belong to the semantic class of *Country*.

Framework. The workflow of Set-CoExpan is shown in Figure 2.3. It consists of two modules that are operated iteratively: Auxiliary sets generation module that finds auxiliary sets holding certain relations with the target set in an unsupervised way, and 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 unsupervisedly by intra-seed clustering and inter-seed merging. The latter step captures parallel relations among inter-seed terms by imposing a parallelogram in the embedding space. The second module, multiple sets co-expansion, 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.

Semantic Space Learning. We first learn a semantic space for related terms retrieval for core entities based on Word2Vec [131] which is commonly used to retrieve relevant words of

a certain seed in the embedding space. training objective is

$$L_l = \sum_{d \in D} \sum_{i=1}^{|d|} \sum_{0<|j-i|\leq h} P(w_j|w_i) = \sum_{d \in D} \sum_{i=1}^{|d|} \sum_{0<|j-i|\leq h} \frac{v_{w_i} u_{w_j}}{\sum_{w_{j'} \in V} v_{w_i} u_{w'_j}} \tag{2.1}$$

Besides this local context, we also add global context in embedding training, which assumes that words appeared in the same document tend to be semantically similar. A distributed representation for document as context is added to fulfill this goal. The training objective resembles the above one while the goal is to present which document is correct for the target word to appear in.

$$L_g = \sum_{d \in D} \sum_{i=1}^{|d|} P(d|w_i) = \sum_{d \in D} \sum_{i=1}^{|d|} \frac{v_{w_i} u_d}{\sum_{d' \in D} v_{w_i} u_{d'}} \tag{2.2}$$

The global context ensures that words close in the embedding space are topically similar, thus complement the conventional word embedding like Word2Vec that only looks into a very small window size. At the end of each epoch in embedding training, for each entity e, we can retrieve the k most similar words by computing the following score for each entity e' in the vocabulary and get the k highest ones, denoted as Z_e.

$$Sim(e', e) = \cos(w_{e'}, w_e) \tag{2.3}$$

We refine the embedding space by adding a regularization term to push words in Z_e toward their relevant seed entity e while away from the other ones in each epoch. Let $\overrightarrow{E}$ be a list of entities in the target seed set, $\overrightarrow{E} = [e_1, e_2, \cdots, e_{|S|}]$, $e_i \in S$.

$$L_r = \sum_{e \in S} \sum_{w \in Z_e} KL(\mathbb{I}[w \in Z_{\overrightarrow{E}}] \| Sim(w, \overrightarrow{E})) \tag{2.4}$$

By minimizing the above KL-divergence, the most similar words are trained to be closer to their relevant entities, and will gradually impact other similar words to be clustered around the relevant entity in later iterations of embedding training.

The overall training objective is the sum of global, local and regularization loss. Finally, the most relevant words for each entity is retrieved at the last epoch of embedding training.

Auxiliary Sets Generation. Here we introduce how we generate multiple auxiliary sets that are formed by entities of parallel relation to the seed entities in the target semantic class.

Figure 2.4 shows an example of how new semantic classes are confirmed by the constraint of

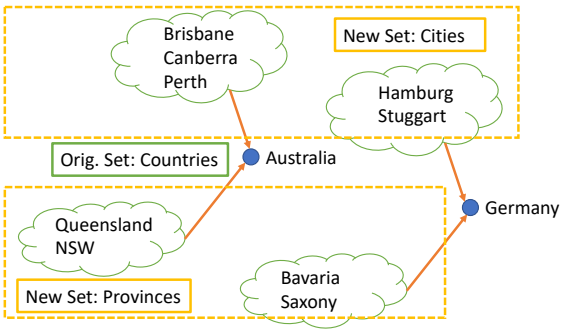


Figure 2.4: Generating Sets of entities from different semantic classes.

cross-seed parallel relations. Specifically, for seed countries "Australia" and "Germany", their retrieved relevant words can be first clustered into initial groups belonging to certain seed entities: $G_{Australia} = \{g^1_{Australia}, g^2_{Australia}\}$, where $g^1_{Australia} = \{\text{Brisbane, Canberra, Perth}\}$ and $g^2_{Australia} = \{\text{Queensland, NSW}\}$. For entity "Germany", several groups can also be formed to generate $G_{Germany}$. Then the groups with same cross-entity relations will be detected as $g^1_{Australia} = \{\text{Brisbane, Canberra, Perth}\}$ and $g^1_{Germany} = \{\text{Hamburg, Stuggart}\}$ are both cities in their subscript countries. Then a new parallel set could be confirmed by merging $g^1_{Australia}$ and $g^1_{Germany}$ to S_1, where S_1 is the subset of C_1, a semantic class of cities.

We use constrained agglomerative clustering to merge similar entites, with similarity measure as the Euclidean distance between terms in the BERT [38] embedding space, where the representation of each entity is calculated by averaging the contextualized embedding of each occurrence in the corpus. We choose *"complete"* linkage for constructing more compact clusters.

Inspired by the word analogy examples in [131] that shows the embedding space hold the property of capturing the relations between words, studies on relation extraction [12, 191, 108] relies on the idea that for a triplet of head <entity, relation, tail entity> and their vector representation $<h, r, t>$, it holds that:

$$h + r \approx t \tag{2.5}$$

This equation basically shows that if two pairs of entities have same relations, then their difference in the vector space are parallel. The theoretical explanation is also given in [3].

By now we have generated seeds for multiple parallel sets that hold strong relations with the original input set S. These strong relations ensure that they are related but different from the original semantic class. We will show how to utilize seeds from these sets to benefit our model in distinguishing between these different semantic classes.

Co-Expanding Core Set and Auxiliary Sets. Conventional Set Expansion methods often iteratively refine two elements: feature pool and candidate pool. Feature pool stores common context features of seed entities, which best describe the target semantic class. When more entities are added into the original set, the pattern pool will be refined by re-selecting the context features (e.g., skip-gram) that maximize the similarity of user given and currently expanded seeds. Candidate pool stores the possible candidates to be expanded, and they are narrowed down by co-occurrence with patterns in the pattern pool. Weights of connecting an entity and a pattern include TF-IDF transformation, PMI, discounted PMI , BM-25 scoring and etc. In our model, we choose TF-IDF transformation as the skip-gram weights, specifically, the score of a skip-gram to an entity is defined as

$$f_{e,c} = \log(1 + X_{e,c})[\frac{\log |E|}{\log X_{e',c}}] \tag{2.6}$$

where $X_{e,c}$ is the number of co-occurrence between entity e and context c, and $|E|$ is the number of candidate entities. This score function resembles TF-IDF in that entities can be seen as *"documents"*, while skip-grams are the *"terms"* that they contain.

To capturing contrastive skip-gram contexts for multiple sets co-expansion, we can score skip-grams shared by more pairs of entities belonging to the same set higher, while penalize skip-grams shared by arbitrary pairs from different sets. Then we have the following criterion to select the most contrastive skip-gram context features.

$$F^* = \arg\max_F \sum_{S_k} \sum_{e_i,e_j \in S_k} Sim(e_i, e_j|F) - \sum_{S_k,S_{k'}} \sum_{e_i \in S_k, e_j \in S_{k'}} Sim(e_i, e_j|F) \tag{2.7}$$

The context dependent similarity function $Sim(e_i, e_j|F)$ is defined as the same in [165] using the weighted Jaccard similarity measure:

$$Sim(e_1, e_2|F) = \frac{\sum_{f \in F} \min(f_{e_1,c}, f_{e_2,c})}{\sum_{f \in F} \max(f_{e_1,c}, f_{e_2,c})} \tag{2.8}$$

To solve the NP-hard optimization problem in Equation 2.7, we apply a greedy selection process that increments the score the most, and would benefit us in selecting complimentary features instead of focusing on strong but similar signals. The above criterion chooses skip-gram contexts that force satellite sets as well as the core set to be coherent, so that each of them will expand in a right direction that avoid penetrating each other's territories.

Methods	*Wiki*			*APR*		
	MAP@10	MAP@20	MAP@50	MAP@10	MAP@20	MAP@50
EgoSet	..	..	..	..	..	..
CaSE	0.897	0.806	0.588	0.619	0.494	0.330
SetExpander	0.499	0.439	0.321	0.287	0.208	0.120
SetExpan	0.944	0.921	0.720	0.789	0.763	0.639
BERT	0.850	0.830	0.760	..	..	..
Full Model (no aux.)	0.964	0.950	0.861	..	..	..
Full Model (no flex.)	0.973	0.961	0.886	..	..	..
Full Model	0.976	0.964	0.899	0.933	0.915	0.830

Table 2.1: Mean Average Precision across all queries on *Wiki* and *APR*.

2.4 EVALUATION OF ENTITY SET EXPANSION

We conduct our experiments on two large corpus also used in previous studies [165]: (1) **Wiki** is a subset of Wikipedia English dump from May 2011, which contains 780,556 entries. (2) **APR** contains news articles published by two news platforms, Associated Press and Reuters in the year of 2015, and include a total of xxx articles. We use the same queries of seed sets and ground truth of each semantic classes with previous studies as well. For **Wiki** dataset, there are 8 semantic classes, and 5 queries per class. For **APR** dataset, there are 4 semantic classes, and 5 queries for each class. Each query contains three seeds belonging to the same semantic class.

Evaluation Metric. We use Mean Average Precision (MAP) to evaluate the ranking result of our methods and other baseline methods. For a query q with 3 seeds, we take the first 10, 20, 50 entities of each algorithm as the output list L and calculate Average Precision at k, defined as

$$AP_k(L) = \sum_{i=1}^{k} P(k)\Delta r(k) \tag{2.9}$$

where $P(k)$ denotes precision@k and $Deltar(k)$ denotes the change in recall from the $k-1^{th}$ term to the k^{th} term. Mean Average Precision at k is defined as the mean of Average Precision at k over all queries.

$$MAP_k = \sum_{L} AP_k(L) \tag{2.10}$$

Results and Discussions. For our method and all baseline methods, we use the same queries in the datasets, and report $MAP@k$ $(k = 10, 20, 50)$ in Table 2.1. The result clearly shows that our model has the best performance over all the methods in both datasets, which

justifies that auxiliary sets are indeed improve the set expansion process by providing clearer discriminative context features. What's more, our method has a larger margin over other baselines when the ranking list is longer, which indicates that when the seed set gradually grows out of control and more noises appear, our method is able to steer the direction of expanding and setting barriers for out-of-category words to come in. We also observe that SetExpan is the strongest among the baselines and SetExpander has a rather low performance because its classifier is a pre-trained one, and might not adapt well to other corpus. CaSE provides a one-time ranking result, which sacrifices performance in pursuit of efficiency. BERT has a rather stable performance over different length of ranking lists, but it is still not better than our method, which shows that distributional similarity alone is not enough for generating accurate expansion.

Case Study. We further break down the results on *Wiki* Dataset into different semantic classes to demonstrate the power of adding auxiliary sets. In Table 2.2, we show the comparison of our full model and the ablation without auxiliary sets in each class. In the semantic class level, we can see for classes like "*Countries*", "*TV Channels*" and "*Companies*", the auxiliary sets help improve the performance to a great extent. On the other hand, for some categories like "*US States*", "*Diseases*" and "*Sports Leagues*", the improvement is marginal, which implies that in these situations our method does not generate meaningful auxiliary sets to learn from. Since our method of generating auxiliary sets is entirely unsupervised, it is not guaranteed that each semantic class can find some related satellite sets.

Semantic Class	Full Model	Full Model (no aux.)
Countries	0.990	0.968
TV Channels	0.901	0.847
US States	0.930	0.930
Diseases	0.979	0.963
Sports Leagues	0.879	0.860
Chinese Provinces	0.693	0.680
Political Parties	0.960	0.927
Companies	0.857	0.709

Table 2.2: MAP@50 of Full Model vs. Full Model (no aux.) on different semantic classes in *Wiki* Dataset.

Effects of Auxiliary Sets in Guiding Core Set Expansion. To justify that the auxiliary sets help guide the set expansion process, we randomly select three semantic class in the *Wiki* Dataset and show the core set expansion results under the guide of these auxiliary

sets in Table 2.3. In comparison, we also show the set expansion result without any auxiliary sets. In all three cases, our algorithm successfully extracts meaningful satellite sets that are relevant to the core set but belong to different types.

Dimension	Companies	Countries	TV Channels
Query	Myspace, Youtube, Twitter	Australia, France, Germany	ESPN News, ESPN Classic, ABC
Expansion result of core set (with satellite sets)	Facebook Ebay LiveJournal iTunes (×) New Statesman Daily Telegraph Amazon Border New York Times Napster	Italy Luxembourg Canada Belgium Australia Spain The Netherlands Denmark England Switzerland	Comedy Central STAR Plus PB TB CTV NBC Bravo the CBS the CW ABC Television
Expansion result of core set (without satellite sets)	Facebook iTunes (×) LiveJournal Digg (×) GMail (×) New Statesman Flickr (×) Daily Telegraph WordPress (×) Paypal	Italy Luxembourg Canada Belgium Spain Australia England Scandinavia (×) Switzerland Ireland	PB TB STAR Plus FOX Sports Net Comedy Central CBS NBC Bravo the Price is Right (×) Cartoon Network

Table 2.3: Satellite sets of different dimensions and their contribution to overall performance

2.5 SEED-GUIDED TOPICAL ONTOLOGY CONSTRUCTION

In this section, we introduce seed-guided topical ontology construction, where user provides a seed ontology as guidance, and a more complete topical ontology is generated from text corpus, with each node represented by a cluster of terms (topics).

My proposed framework CoRel has two modules as shown in 2.2: (i) A relation transferring module learns the specific relation preserved in seed ontology and attaches new concepts to existing nodes to complete the ontology structure.; and (ii) a concept learning module captures user-interested aspects and enriches the semantics of each concept node. Two challenges are met in the course: (1) Fine-grained concept names can be close in the embedding space, and enriching the concepts might result in relevant but not distinctive terms (e.g., "sugar" is relevant to both "cake" and "ice-cream"); and (2) with minimal user input, it is nontrivial to directly apply weakly supervised relation extraction methods to expand the ontology structure. Overall, noisy terms may harm the quality of new topics found.

To address these challenges, the relation transferring module first captures the relation preserved in seed parent-child pairs and transfers it upwards and downwards for finding the

first-layer topics and subtopics, attached by a co-clustering technique to remove inconsistent subtopics. The concept learning module learns a discriminative embedding space by jointly embedding the ontology with text and separating close concepts in the embedding space.

The two modules are introduced in the following paragraphs.

Relation Classifier Training. The relation transferring module first captures the relation between user given $\langle p,c \rangle$ pairs by training a relation classifier on the given corpus. The relation classifier takes a relation statement of a pair of terms as input, and judges whether there exists user-interested relation and which direction it is between the pair. After training the relation classifier, we transfer the relation upwards for root node discovery, and then transfer the relation downwards to find new topics/subtopics as the child of root/topic node. In both the example and evaluation we construct two layers of topics, though this module can be applied to discover more fine-grained topics by further going down.

Previous studies show that word analogy can capture relations between words to some extent [131, 166], but vector offset is only preserved in a local area in the embedding space [49]. To deal with more complicated relations, our choice of the model is inspired by the effectiveness of the pre-trained deep language model, BERT [38], on wide downstream applications. [174] also shows its power in few-shot relation learning by training the model in a distant supervised setting using large amounts of pairs of entities on Wikipedia corpus. In our setting, user gives minimal seeds which limits the potential to train such deep language model, thus we only employ the pre-trained BERT model and train a relation classifier as shown in Figure 2.6.

We assume that if a pair of $\langle p,c \rangle$ co-occurs in a sentence in the corpus, then that sentence implies their relation. We refer to sentences containing $\langle p,c \rangle$ as their relation statements and leverage a pre-trained deep language model to understand the relation statements. To learn the user-interested specific relation, we extract all the relation statements of user-given

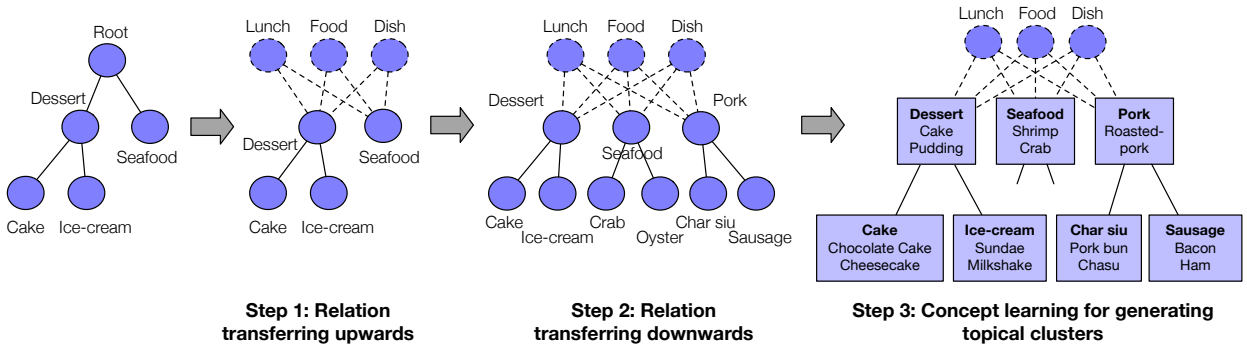


Figure 2.5: Workflow of CoRel.

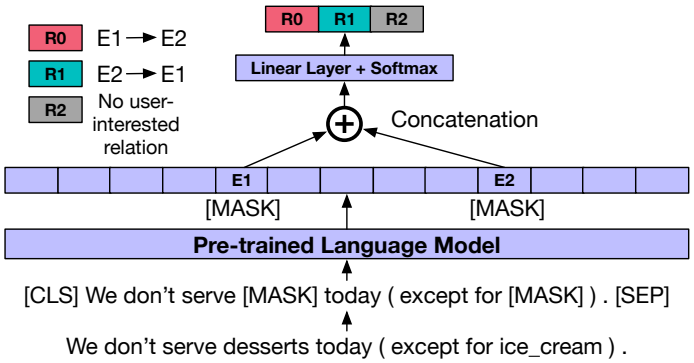


Figure 2.6: Our weakly supervised relation classifier.

$\langle p_0, c_0 \rangle$ as positive training samples. We collect negative training sentences in two ways: (1) relation statements of sibling nodes, thus avoiding the model to only find closely related terms; and (2) random sentences from the corpus, so the model can learn from irrelevant contexts to avoid overfitting.

We take the output of two "[MASK]" tokens from the last layer of the pre-trained language model, and concatenate them to be the input of the classification layer, where we use a simple linear layer before the softmax layer. The output label chooses the relation from e_1 to e_2 among three classes in the relation set $\mathcal{Q}$: $\langle e_1 \rightarrow e_2 \rangle$ (i.e., e_1 is a parent node of e_2), $\langle e_2 \rightarrow e_1 \rangle$, or non-user interested relation.

To fully utilize the asymmetric property along the ontology edges, we augment each input training sequence by reversing the order of concatenation of $e1$ and $e2$. Then the label would switch if user-interested relation exists between the pair, but will not change otherwise.

After deriving a relation classifier, we can easily transfer the user-interested relation along the paths in the ontology. This is done by targeting an existing node and finding entities to be its potential parent node (transferring upwards) or child node (transferring downwards).

Ontology Completion by Relation Transferring. To find the parent or child of an existing node, we extract relation statements of a concept e and a candidate term w into the relation classifier to judge sentence-based relations. Corpus-based relation between w and e is then averaged over confident sentence-based results over the corpus, with the confidence threshold being δ.

$$\text{Score}(w \rightarrow e) = \frac{\sum_{s_{w \rightarrow e}} \mathbb{I}\left(KL\left(\boldsymbol{l} \| \boldsymbol{p}_w\right) > \delta\right)}{\sum_{q \in \mathcal{Q}} \sum_{s_q} \mathbb{I}\left(KL\left(\boldsymbol{l} \| \boldsymbol{p}_w\right) > \delta\right)} \tag{2.11}$$

where s_q denotes relation statements in which the relation of q exists, $\boldsymbol{p}_w$ denotes the output probability from the relation classifier, and l is the uniform distribution vector among three

classes of relations. Thus if the KL divergence between the two distributions is larger than a threshold δ, we treat the prediction as a confident one. Eq. (2.11) calculates the portion of term w being the parent of concept e among all the confident predictions, and we confirm w as the parent node of e if the portion is larger than a threshold. For each user-given first-layer topic, we can generate a list of parent nodes, and their common parent nodes are treated as root nodes R.

We apply the relation classifier to extract child terms for each root node $r \in R$. This is done in a similar way as root node discovery, but we only reverse the direction of relation. Thus we need to replace $(w \rightarrow e)$ in Eq. (2.11) with $(r \rightarrow w)$. New topics are selected by their average score over all root nodes.

$$\text{Score}(R \rightarrow w) = \frac{\sum_{r \in R} \text{Score}(r \rightarrow w)}{|R|} \tag{2.12}$$

Generating Topical Clusters by Concept Learning. Our concept learning module is used to learn a discriminative embedding space, so that each concept is surrounded by its representative terms. Within this embedding space, subtopic candidates are also clustered to form coherent subtopic nodes. This is motivated by the fact that relevant concept names can be close to each other in the embedding space, and directly using unsupervised word embedding such as Word2Vec [131] might result in relevant but not distinctive terms (e.g., "food'' is relevant to both "seafood'' and "dessert''). Thus we use the expanded ontology as input to guide the word embedding learning process.

We want to regularize the embedding space to be discriminative among the concepts in the expanded ontology, we wish to form topical clusters in the embedding space where each concept embedding is surrounded by its representative terms. We apply similar semantic learning methods as mentioned in section 2.3.

To find subtopics for existing concepts in the seed ontology, we apply two constraints for generating potential candidates for subtopics: (1) Topical constraints: candidates should belong to the topic cluster C_e of that concept e; and (2) relational constraints: candidates should bear user-interested relation with the concept. The two constraints can be applied by using the learned relation classifier and concept embedding. However, directly using the few-shot relation classifier can still include noisy and non-consistent terms, thus we carry out a co-clustering method to further filter out those noisy terms.

An example is shown in Table 2.4, where we use a Topic-Type table to organize the valid terms from the relation classifier. Valid terms are divided in columns by semantic meaning and in rows by semantic type (e.g., *food*, *cooking style* or *sauces*). It is easily observable that the fourth subtopic of "pieces, slices" is an outlier sharing little type similarity with other

subtopics. Thus we apply co-clustering method to retain those subtopics sharing similar semantic type distribution.

Beef		**Pork**	**Bread**	
sliced beef	sirloin, rare beef	roasted pork		flat bread, wheat
stewed				toasted
			pieces, slices	
	black pepper	spicy sauce		buttery

Table 2.4: An example of a Topic-Type table.

We construct an indicative (0/1) Topic-Type matrix to represent the joint distribution of subtopics and types of candidates from the Topic-Type table as shown in Table 2.4, and the table is created by the following process: The topic-wise clustering is done by affinity propagation (AP) clustering [47] in the discriminative embedding space trained by concept learning, where the concepts are separated away from each other to avoid overlapping. The type-wise clustering is conducted by AP on the average BERT embedding space: We first retrieve the contextualized embedding of each candidate mention using the last layer output of BERT, and then average over the mentions to get the embedding for each candidate.

Finally, to extract high quality subtopics, we apply co-clustering [90] on the indicative Topic-Type matrix M and define a consistency score for each cluster.

$$\text{Consistency}(\text{Cluster}_k) = \frac{\sum_{\text{row label[i]==k}} \sum_{\text{col. label[j]==k}} M_{i,j}}{\sum_{\text{row label[i]==k}} \sum_{\text{col. label[j]==k}} 1} \tag{2.13}$$

If the consistency score of a cluster is high, then the cluster consists of multiple subtopics that share similar semantic types. We retain high quality subtopics by setting a threshold for the consistency score.

2.6 EVALUATION OF TOPICAL ONTOLOGY CONSTRUCTION

In this section we demonstrate the evaluation of our proposed method CoRel. Our experiments are conducted on two large real-world datasets: (1) **DBLP** contains around 157 thousand abstracts from publications in the field of computer science. For preprocessing, we use AutoPhrase [164] to extract meaningful phrases to serve as our vocabulary, we further discard infrequent terms occuring less than 50 times, resulting in 16650 terms. (2) **Yelp** is collected from the recent released *Yelp Dataset Challenge*[1], containing around 1.08 million

[1]https://www.yelp.com/dataset/challenge

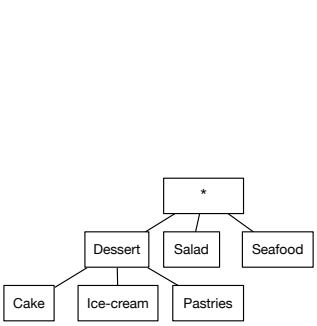
Topics	Method	All Subtopics found
DBLP-Machine Learning	Hi-Expan	**Support vector machine**, **Decision trees**, **Neural networks**, Regression models, Genetic algorithm, Naive Bayes, Classification, Random forest, Markov random field, Nearest Neighbor, Unsupervised learning, Artificial neural networks (✂), Multilayer perceptron (✂), Support vector regression (✂), Conditional random fields (✂), hidden markov models (✂), Radial basis function(✂), Self-organizing map (✂), Recurrent neural networks (✂), Extreme learning machine(✂), Particle swarm optimization (×),
	CoRel	**Support vector machine**, **Decision trees**, **Neural networks**, Regression, Genetic algorithm, Bayesian networks, Classification, Random forest, Inductive logic programming, Reinforcement learning, Active learning, Boosting algorithms, Transfer learning, object recognition (×), Text classification (×)
DBLP-Data Mining	Hi-Expan	**association rule mining**, **text mining**, **web mining**, outlier detection, anomaly detection, spectral clustering, social network analysis, density estimation,(✂) association rules (✂)
	CoRel	**association rule mining**, **text mining**, **web mining**, outlier detection, anomaly detection, clustering algorithms, social network analysis, data stream mining, data visualization, online analytical processing, rule discovery, predictive modeling, sequence analysis (✂)
Yelp-Dessert	Hi-Expan	**cake**, **ice cream**, **pastries**, latte, cream, gelato, sauce (×), cheesecake (✂), taste (×), pancake (✂)
	CoRel	**cake**, **ice cream**, **pastries**, milk tea, cream, fruit, juice, cooked (×), sugar (×)
Yelp-Seafood	Hi-Expan	fish, shrimps, salmon (✂), meat (×), chicken (×), beef (×), steak (×), pork (×), rice (×)
	CoRel	fish, shrimps, crab, scallop, oysters, mussel, camarones (✂), soy sauce (×), pho (×), low mein (×)

Table 2.5: Comparison of subtopics found under the same topics.

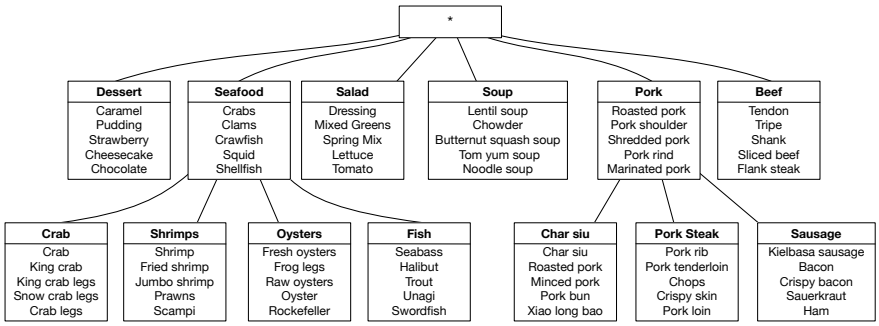
restaurant reviews. Similarly, we extract meaningful phrases and remove infrequent terms, resulting in 14619 terms.

Qualitative Results. Figure 2.7 shows the input seed ontology and parts of the topical ontology generated by CoRel on both datasets. For the **Yelp** dataset, we use minimal user input by only giving child nodes for one input topic to test the robustness of our method. The output in Figure 2.7b shows that we can find new food types such as "soup", "pork" and "beef". For subtopic finding, we can also distinguish between various western and eastern cooking style of pork. The word clusters for each topic/subtopic are obtained by the concept learning module trained on the corpus and the whole ontology. For **DBLP**, we use the same input seed as Hi-Expan [166]. We show that CoRel successfully finds various computer science fields in Figure 2.7d other than user's input, such as "information retrieval" and "pattern recognition". CoRel is also capable of finding separate fields for seed and new research areas found.

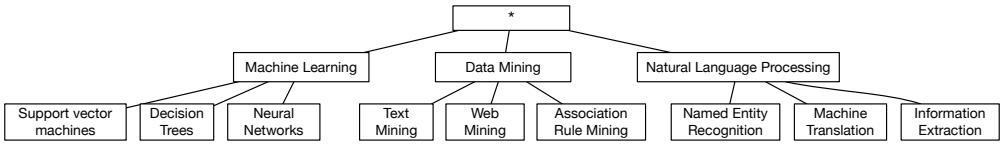
We exhibit the effectiveness of our relation learning module by comparing the subtopics we found with those of HiExpan (seed-guided tree expansion baseline). We randomly choose the common topics shared by both methods, and show all subtopics found by each of them in Table 2.5. The bold ones are seeds from the input ontology, while the wrong subtopics are marked by (×). Since generating synonyms or included concepts at the same level harms the overall quality of a ontology, we mark these terms as redundant (✂). For example, under the topic of "**DBLP**-Machine Learning", HiExpan generates lots of synonyms and subconcepts for "neural networks" at the same level, such as "artificial neural networks" and "multilayer perceptron". These terms should be formed in the topic of "neural networks" instead of its sibling nodes. Our design of concept learning puts these terms into the distinctive term



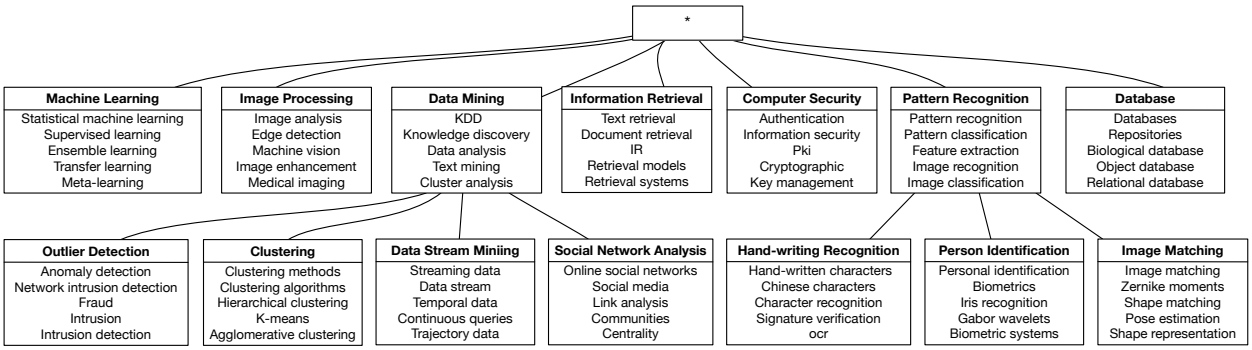
(a) Seed of Yelp



(b) Parts of the ontology generated by CoRel on the Yelp dataset



(c) Seed of DBLP



(d) Parts of the ontology generated by CoRel on the DBLP dataset

Figure 2.7: Input and part of output ontology generated by CoRel on **DBLP** and **Yelp**.

clusters of corresponding topics, thus avoiding such pitfalls. Under the topic of "**Yelp**-Seafood", we show that simply using word analogy is not robust in capturing parallel relations in the global embedding space, and it is essential to utilize more advance text representations and operations upon them as in our relation learning module.

Quantitative Results. The evaluation of the quality of a topical ontology is a challenging task since there are different aspects to be considered, and it is hard to construct a gold standard ontology that contains all the correct child nodes under each parent node. Following [166, 220], we propose three evaluation metrics: *Term Coherency*, *Relation F1* and *Sibling Distinctiveness* in this study.

- **Term Coherency (TC)** measures the semantic coherence of words in a topic.
- **Relation F1** measures the portions of correct parent-children pairs in a ontology that

preserve user-interested relation.

- **Sibling Distinctiveness (SD)** measures how well the topics are distinctive from their siblings.

We calculate the three metrics as follows. For *TC*, we recruited 5 Computer Science students to judge the results. Specifically, we extract the top 10 representative words under each topic, ask the evaluators to divide these words into different clusters by concepts and compute the size of the cluster with the most words, thus a mixture of terms from different concepts scores lower. Then we take the mean of the results given by all the evaluators as the *TC* of this topic, and average the *TC* of all the topics in a ontology.

For *Relation F1*, we show the evaluators all the parent-children pairs of topics in a ontology and ask them to judge independently whether each pair truly holds user-interested relation. Then we use majority votes to label the pairs and use all the true parent-children pairs from different methods to construct a gold standard ontology. Since each topic is represented by a cluster of words, for simplicity, we consider two clusters as the same if they share the same concept. The *Relation F1* is computed as follow:

$$\begin{aligned} P_r &= \frac{|is\_ancestor_{pred}| \bigcap |is\_ancestor_{gold}|}{|is\_ancestor_{pred}|}, \\ R_r &= \frac{|is\_ancestor_{pred}| \bigcap |is\_ancestor_{gold}|}{|is\_ancestor_{gold}|}, \\ F1_r &= \frac{2P_r * R_r}{P_r + R_r} \end{aligned} \tag{2.14}$$

where P_r, R_r and $F1_r$ denote the *Relation Precision*, *Relation Recall* and *Relation F1*, respectively.

Finally, we calculate *Sibling Distinctiveness (SD)* as follows: we compute the similarity between a topic cluster C_i and each of its sibling topics C_j by Jaccard index [64]. Then we calculate *SD* of C_i as 1 minus the largest similarity score among all C_j. A larger *SD* means the sibling topics sharing a common parent are truly separate from each other.

Table 2.6 shows the *Term Coherency (TC)*, *Relation F1*, and *Sibling Distinctiveness (SD)* of different methods. For unsupervised baselines, we only take relevant topics (topics with more than half terms belonging to *food* or *research areas*) into account. Overall, weakly-supervised methods (Hi-Expan and CoRel) outperform unsupervised methods by a large margin, which shows the constructed taxonomies are well guided by the user given seeds. We can see that CoRel achieves the best performance under all evaluation metrics, especially in terms of the *Relation F1*, showing that CoRel is able to find related terms for each concept and retain ones holding certain relations with current topics in relation transferring module. For *TC*,

CoRel also significantly outperforms HLDA, HPAM and TaxoGen, which model the intrinsic distribution of terms in documents or corpus, and might generate topics as mixtures of terms relevant but not distinctive of user-interested topics. They also have inferior performance in *SD* since they do not enforce distinctiveness when forming topics.

Methods	DBLP					Yelp				
	TC	SD	Precision$_r$	Recall$_r$	F1-score$_r$	TC	SD	Precision$_r$	Recall$_r$	F1-score$_r$
HLDA	0.582	0.981	0.188	0.577	0.283	0.517	0.991	0.135	0.387	0.200
HPAM	0.557	0.905	0.362	0.538	0.433	0.687	0.898	0.173	0.615	0.271
TaxoGen	0.720	0.979	0.450	0.429	0.439	0.563	0.965	0.267	0.381	0.314
Hi-Expan + CoL.	0.819	0.996	0.676	0.532	0.595	0.815	**1.000**	0.429	0.677	0.525
CoRel	**0.855**	**1.000**	**0.730**	**0.607**	**0.663**	**0.825**	**1.000**	**0.564**	**0.710**	**0.629**

Table 2.6: Quantitative evaluation on topical taxonomies.

Case Study. We are interested in what kind of root nodes could be generated for different user inputs, since the root node discovery is crucial for the connection to find new topics. In Table 2.7 we show the top ranked terms scored by Equation (2.12) as well as bottom ranked ones. It is interesting that we are able to find terms that can summarize the user's interested topics from different aspects or granularity. For example, in the **DBLP** dataset, we observe that "major", "ai", and "science" rank top, providing summarization for user given topics in different granularity. we observe that "major" ranks top as the root node, which might appear in terms like "major interest" or "major in", and this word could cover very broad subjects. The second ranked term "ai", however, shows a rather narrow though precise scope to summarize the user given topics. On the other hand, bottom ranked words like "database" and "multimedia" are not good at capturing the broader field of given seeds, thus will not be selected as root nodes. On the **Yelp** dataset, we can also find highly concluding terms like "lunch special" or "chinese food", each potential root node partially summarizes the user-input topics. Due to the different granularity and aspects of possible root nodes, we do not put them in the final output of our topical ontology since they cannot form coherent clusters, but they have shown to be powerful in voting for new topics from different aspects.

2.7 SUMMARY

In this chapter we introduce two fundamental tasks in concept organization: entity set expansion and topical ontology construction. We focus on a low-resource setting where only keywords/seed ontology is given from the user. Our proposed frameworks Set-CoExpan and CoRel both utilizes a semantic learning method that captures both local and global context similarities for terms. The learned semantic space helps enrich the semantics of each concept.

DBLP		Yelp	
Term	Score	Term	Score
major	0.963	lunch special	0.950
ai	0.892	chinese food	0.800
science	0.877	main dish	0.788
research	0.826	daily special	0.750
journal	0.667	signature	0.725
...	...	...	...
multimedia	0.286	wheat	0.250
information extraction	0.250	freshness	0.250
database	0.242	sushi	0.188

Table 2.7: Top and bottom terms in root node discovery.

The proposed frameworks complete the ontology structure by relying on the relation or analogy between nodes, then transfers it along multiple paths to expand the ontology in width and depth. Extensive experiments show that both frameworks work effectively in generating a high-quality topical ontology based on user-given seeds.

With the recent development of Large Language Models (LLMs), new approaches [182] have been developed to construct explicit knowledge forms such as ontology and knowledge graphs. However, LLM-based methods tend to make mistakes in fine-grained entities, as these entities appear less in the LLM pre-training data. From this perspective, our approach can better leverage domain-specific corpus to recognize fine-grained entities, by training the model to recognize potential user-interested relations. Fine-tuning LLMs on domain-specific corpus for each new task, on the other hand, could be computationally inefficient. However, LLMs can still be integrated with my methods for improving the task. Though language models may not perform well on generating fine-grained entities, they have been proven as good text evaluators [115], which can be used to decide whether new terms identified by my methods are suitable to be attached to the ontology.

CHAPTER 3: FINE-GRAINED ENTITY EXTRACTION

Extracting key information from text corpora serve as foundational steps in text mining applications. Named Entity Recognition (NER) involves processing unstructured text, locating and classifying named entities (certain occurrences of words or expressions) into particular categories of pre-defined entity types, such as persons, organizations, locations, medical codes, dates and quantities. NER serves as an important first component for tasks such as information extraction [157], information retrieval [60], question answering [135], task-oriented dialogues [145, 50] and other language understanding applications [137, 163]. In this chapter, I will introduce two studies on Named Entity Recognition (NER) and Fine-grained Entity Typing (FET), especially in scenarios with limited labeled data. I will first provide a comprehensive overview of the few-shot learning methodologies for NER and identify useful techniques, and then introduce novel approaches for FET that harness the representation and generation capabilities of PLMs. The empirical results demonstrate the effectivenss of these methods and could shed light on the future entity recognition studies.

3.1 OVERVIEW

Named Entity Recognition (NER) and Fine-grained Entity Typing (FET) serve as important first steps in text processing, and benefit downstream tasks such as question answering [135], task-oriented dialogues [145, 50] and other language understanding applications [137, 163]. NER identifies named entities in unstructured text and classify them into predefined categories like persons, organizations, and locations. FET aims to infer types of named entity mentions in specific contexts, which serves as the essential component for many downstream text mining applications, such as entity linking [18], knowledge completion [42], text classification [73]. With the recent success of pre-trained language models, state-of-the-art (SoTA) NER models are often initialized with PLM weights, fine-tuned with standard supervised learning. One classic approach is to add a linear classifier on top of the representations provided by PLMs, and fine-tune the entire model using a cross-entropy objective on domain-specific labels [38].

Unfortunately, even with these PLMs, building NER and FET systems still remains a labor-intensive, time-consuming task. It requires rich domain knowledge and expert experience to annotate a large corpus of in-domain labeled tokens to teach the models to achieve a reasonable accuracy. However, this is in contrast to the real-world application scenarios, where only very small amounts of labeled data are available for new domains, such as medical [71]

domain.

To address this issue, my research focuses on few-shot NER and few-shot FET, aiming to improve the generalization ability of PLMs on these entity-related tasks. I first present a systematic study for few-shot NER, a problem that is little explored in the literature. Three distinctive schemes and their combinations are investigated, and erform comprehensive comparisons of these schemes on 10 public NER datasets from different domains.

I also explore hierarchical fine-grained FET with few-shot learning setting, where the entity labels form a hierarchy, and a few annotated mentions within a particular context are given for each entity type in the label hierarchy. Recent studies have shown that factual knowledge can be extracted from PLMs using natural language prompts, especially effective in few-shot tasks.

Recently, some studies [148] show that without any further training of PLMs, factual knowledge can be probed from PLMs via natural language prompts that query the PLM's MLM head for masked word prediction. Prompt-based tuning for few-shot FET has also been explored very recently [40]: A prompt template is concatenated with the original context containing the entity mention, and a PLM is used to predict the masked token in the template. Then a verbalizer maps the probability distribution over the vocabulary to a distribution over the entire label set. The verbalizer is typically constructed by extracting semantically relevant words to each label name from external knowledge bases. However, the above method still suffer from notable limitations that the verbalizer construction method cannot be applied to label names that have composite semantics or label names that share similar semantics. Moreover, it has not fully explored the *generation power* of PLMs acquired through extensive general-domain pre-training.

I propose a novel framework, named ALIGNE, which introduces an entity type label interpretation module to understand the relationship between type labels and vocabulary tokens. Additionally, a type-based contextualized instance generator is proposed to create new instances based on few-shot instances and their types, enhancing the training set for better generalization. In tests on three benchmark datasets, ALIGNE has outperformed existing methods significantly.

Below I will introduce these two works in detail.

3.2 RELATED WORK

General Named Entity Recognition. Named Entity Recognition(NER) is a long standing problem in NLP. Deep learning has significantly improved the recognition accuracy. Early efforts include exploring various neural architectures [96] such as Bidirectional LSTMs [23] and

adding CRFs to capture structures [119]. Early studies have noticed the importance of reducing the annotation labor, where semi-supervised learning is employed, such as clustering [104], and combining supervised objective with unsupervised word representations [183]. PLMs have recently revolutionized NER, where large-scale Transformer-based architectures [147, 38] are used as backbone network to extract informative representations. Contextualized string embedding [2] is proposed to capture subword structures and polysemous words in different usage. Masked words and entities are jointly trained for prediction in [204] with entity-aware self-attention. These methods are designed for standard supervised learning, and have a limited generalization ability in few-shot settings, as empirically shown in [48].

Pre-Trained Models. In computer vision, it is a *de facto* standard to transfer ImageNet-supervised pre-trained models to small image datasets to pursue high recognition accuracy [212]. The recent work named big transfer [92] has achieved SoTA on various vision tasks via pre-training on billions of noisily labeled web images. To gain a stronger transfer learning ability, one may combine supervised and self-supervised methods [100, 99]. In NLP, supervised/grounded pre-training have been recently explored for natural language generation (NLG) [88, 219, 146, 51, 98]. They aim to endow GPT models [151], an ability of enabling high-level semantic controlling in language generation, and are often pre-trained on massive corpus consisting of text sequences associated with prescribed codes such as text style, content description, and task-specific behavior. In contrast to NLG, to our best knowledge, large-scale supervised pre-training has been little studied for natural language understanding (NLU). There are early studies showing promising results by transferring from medium-sized datasets to small datasets in some NLU applications. For example, from MNLI to RTE for sentence classification [149, 27, 5], and from OntoNER to CoNLL for NER [208]. Our work further increases the supervised pre-training at the scale of web data [55], 1000 orders of magnitude larger than [208], showing consistent improvements.

Few-Shot Learning Methods with Pre-Trained Models. PLMs [38, 117, 27] have shown remarkable performance through fine-tuning on downstream tasks [72], thanks to their learned generic linguistic features [148] via pre-training. Prototype-based methods was firstly studied in the context of image classification [184, 177, 225], and has recently been adapted to different NLP tasks such as text classification [188, 53, 9], machine translation [57] and relation classification [66]. Some studies also show that PLMs are able to capture factual knowledge [148] by making predictions on manually curated "fill-in-the-blank" cloze-style prompts. Researchers use manually created prompts as task descriptions on GPT-3 [13] and found that prompts can guide the model to generate answers for specific tasks,

especially effective on few-shot tasks. Inspired by these observations, studies on prompt engineering [82, 169] aim to automatically search for optimal templates on specific tasks. In few-shot settings where training data is very limited, prompt-based tuning has surpassed standard model fine-tuning in a wide range of applications including text classification [65, 74], relation extraction [21], named entity recognition [20, 97], and fine-grained entity typing [40].

Fine-Grained Entity Typing. The goal of fine-grained entity typing (FET) is to determine the type of entity given a particular context and a label hierarchy [110, 211]. Several early studies [110, 56, 33] generate large training data by automatically labeling entity mentions using external knowledge bases, which was followed by a line of research known as distantly-supervised FET with the goal of denoising the automatically generated labels [156]. Some studies focus on leveraging type hierarchy and type inter-dependency [155, 19, 136, 107] for noise reduction, while other studies [85, 32, 202, 102, 127, 221] combine external entity information provided in knowledge bases and self-training techniques to automatically relabel the data. Recently, joint training with relation extraction [203] and searching for optimal templates under prompt-based framework [34] are also proposed in the distantly-supervised setting.

Zero-shot FET has also been explored to mitigate human annotation burden by transferring the knowledge learned from seen types to unseen ones. Multiple sources of information are used for zero-shot FET: [215] maps mention embedding to type embedding space by training a neural model to combine entity and context information. [222] captures hierarchical relationship between unseen types and seen types in order to generate unseen label representations. [120] further enhances label representations by incorporating hierarchical and prototypical information derived from around 40 manually selected context-free entities as prototypes for each type. [228, 141] define unseen types by generating type embeddings from Wikipedia descriptions. [22] fuses three distinct types of sources: contexutal, pre-defined hierarchy and label-knowledge (label prototype and descriptions), by training three independent modules to combine their results. Nonetheless, these zero-shot learning algorithms require extensive annotations on source domain or manually selected high-quality representative mentions.

3.3 FEW-SHOT NAMED ENTITY RECOGNITION

To deal with the challenge of few-shot learning, I focus on improving the generalization ability of PLMs for NER from three complementary directions, shown in Figure 3.1. Instead of limiting ourselves in making use of limited in-domain labeled tokens with the classic approach, (*i*) I create prototypes as the representations for different entity types, and assign

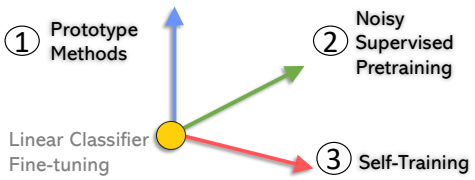


Figure 3.1: An overview of methods studied in our paper. Linear classifier fine-tuning is a default baseline that updates an NER model from pre-trained Roberta/BERT. I study three orthogonal strategies to improve NER models in the limited labeled data settings.

labels via the nearest neighbor criterion; (*ii*) I continuously pre-train PLMs using web data with noisy labels that is available in much larger quantities to improve NER accuracy and robustness; (*iii*) I tag the in-domain unlabeled data with soft labels via self-training [201], and perform semi-supervised learning in conjunction with the limited labeled data.

We provide a comprehensive study specifically for limited NER data settings, and explore three orthogonal directions shown in Figure 3.1: (*i*) How to adapt meta-learning such as prototype-based methods for few-shot NER? (*ii*) How to leverage freely-available web data as noisy supervised pre-training data? (*iii*) How to leverage unlabeled in-domain sentences in a semi-supervised manner? Note that these three directions are complementary to each other and can be used jointly to further extrapolate the methodology space. Below I will introduce the three directions.

Background on Few-Shot NER. Few-shot NER is a sequence labeling task, where the input is a text sequence (*e.g.,* sentence) of length T, $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T]$, and the output is a corresponding length-T labeling sequence $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_T]$, where $\boldsymbol{y} \in \mathcal{Y}$ is a one-hot vector indicating the entity type of each token from a pre-defined discrete label space. The training dataset for NER often consists of pair-wise data $\mathcal{D}^{\text{L}} = \{(\mathbf{X}_n, \mathbf{Y}_n)\}_{n=1}^{N}$, where N is the number of training examples. Traditional NER systems are trained in the standard supervised learning paradigms, which usually requires a large number of pairwise examples, *i.e.,* N is large. In real-world applications, the more favorable scenarios are that only a small number of labeled examples are given for each entity type (N is small), because expanding labeled data increases annotation cost and decreases customer engagement. This yields a challenging task *few-shot NER*.

Following the recent self-supervised PLMs [38, 117], a typical method for NER is to utilize a Transformer-based backbone network to extract the contextualized representation of each token $\boldsymbol{z} = f_{\boldsymbol{\theta}_0}(\boldsymbol{x})$. A linear classifier (*i.e.,* a linear layer with parameter $\boldsymbol{\theta}_1 = \{\mathbf{W}, \boldsymbol{b}\}$ followed by a Softmax layer) is applied to project the representation $\boldsymbol{z}$ into the label space $f_{\boldsymbol{\theta}_1}(\boldsymbol{z}) = \text{Softmax}(\mathbf{W}\boldsymbol{z} + \boldsymbol{b})$. In another word, the end-to-end learning objective for linear

classifier based NER can be obtained via a function composition $\boldsymbol{y} = f_{\boldsymbol{\theta}_1} \circ f_{\boldsymbol{\theta}_0}(\boldsymbol{x})$, with trainable parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_0, \boldsymbol{\theta}_1\}$. The pipeline is shown in Figure 3.2(a). The model is optimized by minimizing the cross-entropy:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(\mathbf{X},\mathbf{Y})\in\mathcal{D}^{\mathsf{L}}} \sum_{i=1}^{T} \mathrm{KL}(\boldsymbol{y}_i || q(\boldsymbol{y}_i|\boldsymbol{x}_i)), \tag{3.1}$$

where the KL divergence between two distributions is $\mathrm{KL}(p||q) = \mathbb{E}_p \log(p/q)$, and the prediction probability vector for each token is

$$q(\boldsymbol{y}|\boldsymbol{x}) = \mathrm{Softmax}(\mathbf{W} \cdot f_{\boldsymbol{\theta}_0}(\boldsymbol{x}) + \boldsymbol{b}) \tag{3.2}$$

In practice, $\boldsymbol{\theta}_1 = \{\mathbf{W}, \boldsymbol{b}\}$ is always updated, while $\boldsymbol{\theta}_0$ can be either frozen [113, 116, 84] or updated [38, 208].

Prototype-Based Methods. Meta-learning [154] has shown promising results for few-shot image classification [181] and sentence classification [213, 53]. It is natural to adapt this idea to few-shot NER. The core idea is to use episodic classification paradigm to simulate few-shot settings during model training. Specifically in each episode, M entity types (usually $M < |\mathcal{Y}|$) are randomly sampled from $\mathcal{D}^{\mathsf{L}}$, containing a *support* set $\mathcal{S} = \{(\mathbf{X}_i, \mathbf{Y}_i\}_{i=1}^{M\times K}$ (K sentences per type) and a *query* set $\mathcal{Q} = \{(\hat{\mathbf{X}}_i, \hat{\mathbf{Y}}_i\}_{i=1}^{M\times K'}$ (K' sentences per type).

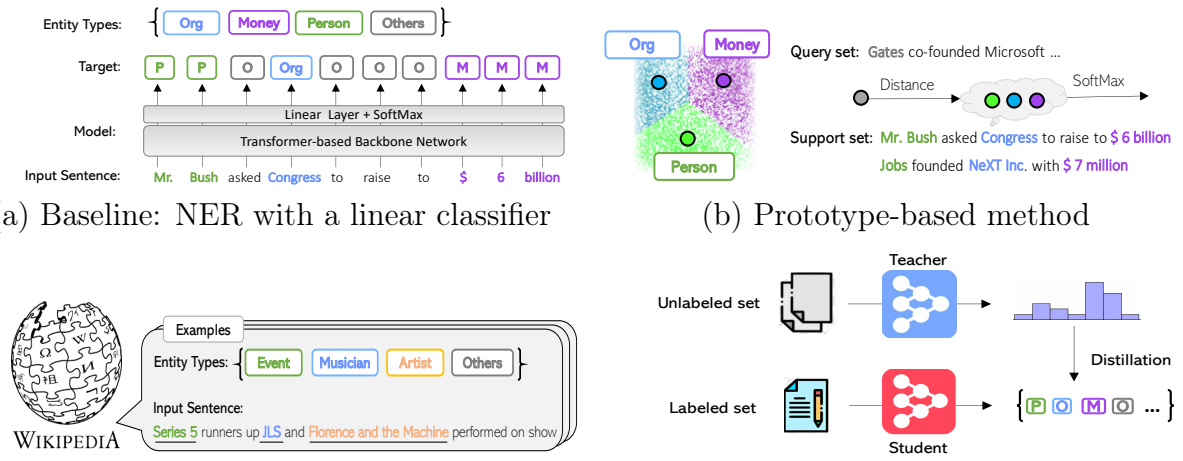
We build our method based on prototypical network [172], which introduces the notion of *prototypes*, representing entity types as vectors in the same representation space of individual tokens. To construct the prototype for the m-th entity type $\boldsymbol{c}_m$, the average of representations is computed for all tokens belonging to this type in the support set $\mathcal{S}$:

$$\boldsymbol{c}_m = \frac{1}{|\mathcal{S}_m|} \sum_{\boldsymbol{x}\in\mathcal{S}_m} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}), \tag{3.3}$$

where $\mathcal{S}_m$ is the token set of the m-th type in $\mathcal{S}$, and $f_{\boldsymbol{\theta}_0}$ is defined in (3.2). For an input token $\boldsymbol{x} \in \mathcal{Q}$ from the query set, its prediction distribution is computed by a softmax function of the distance between $\boldsymbol{x}$ and all the entity prototypes. For example, the prediction probability for the m-th prototype is:

$$q(\boldsymbol{y}=\mathbb{I}_m|\boldsymbol{x}) = \frac{\exp\left(-d(f_{\boldsymbol{\theta}_0}(\boldsymbol{x}), \boldsymbol{c}_m)\right)}{\sum_{m'} \exp\left(-d(f_{\boldsymbol{\theta}_0}(\boldsymbol{x}), \boldsymbol{c}_{m'})\right)} \tag{3.4}$$

where $\mathbb{I}_m$ is the one-hot vector with 1 for m-th coordinate and 0 elsewhere, and $d(f_{\boldsymbol{\theta}_0}(\boldsymbol{x}), \boldsymbol{c}_m) = \|f_{\boldsymbol{\theta}_0}(\boldsymbol{x}) - \boldsymbol{c}_m\|_2$ is used in our implementation. We provide a simple example to illustrate the



(a) Baseline: NER with a linear classifier

(b) Prototype-based method

(c) Noisy supervised pre-training

(d) Self-training

Figure 3.2: Illustration of different methods for few-shot NER. In this example, each token in the input sentence is categorized into one of the four entity types. (a) A typical NER system, where a linear classifier is built on top of unsupervised pre-trained Transformer-based networks such as BERT/Roberta. (b) A prototype set is constructed via averaging features of all tokens belonging to a given entity type in the support set (*e.g.,* the prototype for `Person` is an average of three tokens: *Mr.*, *Bush* and *Jobs*). For a token in the query set, its distances from different prototypes are computed, and the model is trained to maximize the likelihood to assign the query token to its target prototype. (c) The Wikipedia dataset is employed for supervised pre-training, whose entity types are related but different (*e.g.,* `Musician` and `Artist` are more fine-grained types of `Person` in the downstream task). The associated types on each token can be noisy. (d) Self-training: An NER system (teacher model) trained on a small labeled dataset is used to predict soft labels for sentences in a large unlabeled dataset. The joint of the predicted dataset and original dataset is used to train a student model.

prototype method in Figure 3.2(b). At each training iteration, a new episode is sampled, and the model parameter $\boldsymbol{\theta}_0$ is updated via plugging (3.4) into (3.1). In the evaluation phase, the label of a new token $\boldsymbol{x}$ is assigned using the nearest neighbor criterion $\arg\min_m d(f_{\boldsymbol{\theta}_0}(\boldsymbol{x}), \boldsymbol{c}_m)$.

Noisy Supervised Pre-Training. Generic representations via self-supervised pre-trained language models [38, 117] have benefited a wide range of NLP applications. These models are pre-trained with the task of randomly masked token prediction on massive corpora, and are agnostic to the downstream tasks. In other words, PLMs treat each token equally, which is not aligned with the goal of NER: identifying named entities as emphasized tokens and assigning labels to them. For example, for a sentence " *Mr. Bush asked Congress to raise to* $ *6 billion* ", PLMs treat *to* and *Congress* equally, while NER aims to highlight entities like *Congress* and downplay their collocated non-entity words like *to*.

This intuition inspires us to endow the backbone network with an ability to upweight the representations of entities for NER. Hence, we propose to employ the large-scale noisy web data `WiFiNE` [55] for noisy supervised pre-training (NSP). The authors automatically annotated the 2013 English Wikipedia dump by querying anchored strings as well as the coreference mentions in each wiki page to the Freebase. The `WiFiNE` dataset is of 6.8GB and contains 113 entity types along with over 50 million sentences. Though introducing inevitable noises (*e.g.,* a random subset of 1000 mentions are manually evaluated and the accuracy of automatic annotations reaches 77% as reported in the paper, due to the error of identifying coreferences), this automatic annotation procedure is highly scalable and affordable. The label set of `WiFiNE` covers a wide range of fine-grained entity types, which are often related but different from entity types in the downstream datasets. For example in Figure 3.2(c), the entity types `Musician` and `Artist` in Wikipedia are more fine-grained than `Person` in a typical NER dataset. The proposed NSP learns representations to distinguish entities from others. This particularly favors the few-shot settings, preventing over-fitting via the prior knowledge of extracting entities from various contexts in pre-training.

Two pre-training objectives are considered in NSP, respectively: the first one is to use the linear classifier in (3.2), the other is a prototype-based objective in (3.4). For the linear classifier, we found that the batch size of 1024 and learning rate of $1e^{-4}$ works best, and for the prototype-based approach, we use the episodic training paradigm with $M = 5$ and set learning rate to be $5e^{-5}$. For both objectives, we train the whole corpus for 1 epoch and apply the Adam Optimizer [89] with a linearly decaying schedule with warmup at 0.1. We empirically compare both objectives in experiments, and found that the linear classifier in (3.2) improves pre-training more significantly.

Self-Training Methods. Though manually labeling entities is expensive, it is easy to collect large amounts of unlabeled data in the target domain. Hence, it becomes desired to improve the model performance by effectively leveraging unlabeled data $\mathcal{D}^{\mathtt{U}}$ with limited labeled data $\mathcal{D}^{\mathtt{L}}$. We resort to the recent self-training scheme [201] for semi-supervised learning. The algorithm operates as follows:

1. Learn teacher model $\boldsymbol{\theta}^{\mathtt{tea}}$ via cross-entropy using (3.1) with labeled tokens $\mathcal{D}^{\mathtt{L}}$.

2. Generate soft labels using a teacher model on unlabeled tokens:

$$\tilde{\boldsymbol{y}}_i = f_{\boldsymbol{\theta}^{\mathtt{tea}}}(\tilde{\boldsymbol{x}}_i), \forall \tilde{\boldsymbol{x}}_i \in \mathcal{D}^{\mathtt{U}} \tag{3.5}$$

3. Learn a student model $\boldsymbol{\theta}^{\mathtt{stu}}$ via cross-entropy using (3.1) on labeled and unlabeled

tokens:

$$\mathcal{L}_{\text{ST}} = \frac{1}{|\mathcal{D}^{\text{L}}|} \sum_{\boldsymbol{x}_i \in \mathcal{D}^{\text{L}}} \mathcal{L}(f_{\boldsymbol{\theta}^{\text{stu}}}(\boldsymbol{x}_i), \boldsymbol{y}_i) + \frac{\lambda_{\text{U}}}{|\mathcal{D}^{\text{U}}|} \sum_{\tilde{\boldsymbol{x}}_i \in \mathcal{D}^{\text{U}}} \mathcal{L}(f_{\boldsymbol{\theta}^{\text{stu}}}(\tilde{\boldsymbol{x}}_i), \tilde{\boldsymbol{y}}_i) \tag{3.6}$$

where λ_{U} is the weighting hyper-parameter.

A visual illustration for self-training procedure shown in Figure 3.2(d). It is optional to iterate from Step 1 to Step 3 multiple times, by initializing $\boldsymbol{\theta}^{\text{tea}}$ in Step 1 with newly learned $\boldsymbol{\theta}^{\text{stu}}$ in Step 3. We only perform self-training once for simplicity, which has already shown excellent performance.

3.4 EVALUATION OF OUR FEW-SHOT NER METHODS

Throughout our experiments, the pre-trained base RoBERTa model is employed as the backbone network. We investigate the following 6 schemes for the comparative study: (*i*) **LC** is the *linear classifier* fine-tuning method in Section 2, *i.e.,* adding a linear classifier on the backbone, and directly fine-tuning on entire model on the target dataset; (*ii*) **P** indicates the *prototype-based method* in Section 3.1; (*iii*) **NSP** refers to the *noisy supervised pre-training* in Section 3.2; Depending on the pre-training objective, we have **LC**+**NSP** and **P**+**NSP**. (*iv*) **ST** is the *self-training* approach in Section 3.3, it is combined with *linear classifier* fine-tuning, denoted as **LC**+**ST**; (*v*) **LC**+**NSP**+**ST**. We evaluate our methods on 10 public benchmark datasets, covering a wide range of domains.

Main Results. To gain thorough insights and benchmark few-shot NER, we first perform an extensive comparative study on 6 methods across 10 datasets. The results are shown in Table 3.1. We can draw the following major conclusions: (*i*) By comparing columns ① and ② (or comparing ③ and ④), it clearly shows that noisy supervised pre-training provides better results in most datasets, especially in the 5-shot setting, which demonstrates that **NSP** endows the model an ability to extract better NER-related features. (*ii*) The comparison between columns ① and ③ provides a head-to-head comparison between linear classifier and prototype-based methods: while the prototype-based method demonstrates better performance than **LC** on CoNLL, WikiGold, WNUT17 and Multiwoz in the 5-shot learning setting, it falls behind **LC** on other datasets and in average statistics. It shows that the prototype-based method only yields better results when there is very limited labeled

Datasets	Settings	① LC	② LC + NSP	③ P	④ P + NSP	⑤ LC + ST	⑥ LC + NSP + ST
CoNLL	5-shot	0.535	0.614	0.584	0.609	0.567	**0.654**
	10%	0.855	0.891	0.878	0.888	0.878	**0.895**
	100%	0.919	**0.920**	0.911	0.915	-	-
Onto	5-shot	0.577	0.688	0.533	0.570	0.605	**0.711**
	10%	0.861	**0.869**	0.854	0.846	0.867	0.867
	100%	0.892	**0.899**	0.886	0.883	-	-
WikiGold	5-shot	0.470	0.640	0.511	0.604	0.481	**0.684**
	10%	0.665	0.747	0.692	0.701	0.695	**0.759**
	100%	0.807	**0.839**	0.801	0.827	-	-
WNUT17	5-shot	0.257	0.342	0.295	0.359	0.300	**0.376**
	10%	0.483	0.492	0.485	0.478	0.490	**0.505**
	100%	0.489	0.520	0.552	**0.560**	-	-
MIT Movie	5-shot	0.513	0.531	0.380	0.438	0.541	**0.559**
	10%	0.651	0.657	0.563	0.583	0.659	**0.666**
	100%	**0.693**	0.692	0.632	0.641	-	-
MIT Restaurant	5-shot	0.487	0.491	0.441	0.484	0.503	**0.513**
	10%	0.745	0.734	0.713	0.721	**0.750**	0.741
	100%	0.790	**0.793**	0.787	0.791	-	-
SNIPS	5-shot	0.792	0.824	0.750	0.773	0.796	**0.830**
	10%	0.945	**0.950**	0.879	0.896	0.946	0.942
	100%	0.970	**0.972**	0.923	0.956	-	-
ATIS	5-shot	**0.908**	**0.908**	0.842	0.896	0.904	0.905
	10%	0.883	0.898	0.785	0.896	0.898	**0.903**
	100%	0.953	**0.956**	0.929	0.943	-	-
Multiwoz	5-shot	0.123	0.198	0.219	**0.451**	0.200	0.225
	10%	0.826	0.830	0.787	0.805	0.835	**0.841**
	100%	0.880	**0.885**	0.837	0.845	-	-
I2B2	5-shot	0.360	0.385	0.320	0.366	0.365	**0.393**
	10%	0.855	0.869	0.703	0.762	0.865	**0.871**
	100%	0.932	**0.935**	0.895	0.906	-	-
Average	5-shot	0.502	0.562	0.488	0.555	0.526	**0.585**
	10%	0.777	0.794	0.734	0.758	0.788	**0.799**
	100%	0.833	**0.841**	0.815	0.827	-	-

Table 3.1: F1-score on benchmark datasets with various sizes of training data.

data: the size of both entity types and examples are small. (*iii*) When comparing columns ⑤ with ① (or comparing columns ⑥ and ②), we observe that using self-training consistently works better than directly fine-tuning with labeled data only, suggesting that **ST** is a useful technique to leverage in-domain unlabeled data if allowed. (*iv*) Column ⑥ shows the highest F1-score in most cases, demonstrating the three proposed schemes in this paper are complementary to each other, and can be combined to yield best results in practice.

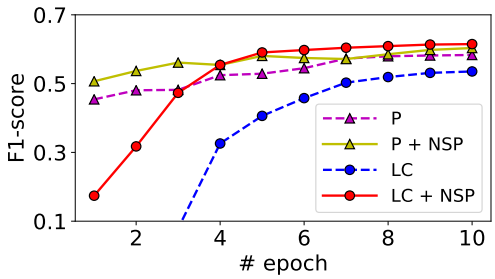


Figure 3.3: Testing F1-score curves on 5-shot NER on CONLL-2003 dataset.

Testing Curve. In Figure 3.3, we show the learning curve of average F1-score on 5-shot CONLL-2003 testing dataset over 10 repeated experiments. The checkpoint via NSP provides a better initialization than Roberta, as NSP exhibits improvement over their counterpart methods at the beginning of learning, and eventually leads to higher F1-score.

Datasets	Methods	*Number of support examples per entity type*					
		10	*20*	*50*	*100*	*200*	*500*
ATIS	Neigh.Tag.†	0.067±0.008	0.088±0.007	0.111±0.007	0.143±0.006	0.221±0.006	0.339±0.006
	Example†	0.174±0.011	0.198±0.012	0.222±0.011	0.268±0.027	0.345±0.022	0.401±0.010
	Prototype	0.381±0.021	0.391±0.022	0.376±0.008	0.379±0.005	0.377±0.006	0.376±0.003
	Prototype + CP	0.684±0.013	0.712±0.014	0.716±0.013	0.705±0.010	0.705±0.006	0.708±0.002
	Multi-Prototype	0.396±0.015	0.415±0.016	0.419±0.012	0.420±0.008	0.422±0.006	0.424±0.005
	Multi-Prototype + CP	**0.712**±0.014	**0.748**±0.011	**0.760**±0.008	**0.742**±0.005	**0.743**±0.003	**0.746**±0.002
MIT.Restaurant	Neigh.Tag.†	0.042±0.018	0.038±0.008	0.037±0.007	0.046±0.008	0.055±0.011	0.081±0.006
	Example.†	0.276±0.018	0.295±0.010	0.312±0.007	0.337±0.005	0.345±0.004	0.346±0.000
	Prototype	0.330±0.013	0.332±0.013	0.332±0.010	0.329±0.003	0.329±0.004	0.331±0.003
	Prototype + CP	0.455±0.016	0.455±0.012	0.455±0.013	0.438±0.013	0.437±0.008	0.438±0.006
	Multi-Prototype	0.345±0.012	0.360±0.015	0.371±0.012	0.376±0.009	0.385±0.005	0.386±0.004
	Multi-Prototype + CP	**0.461**±0.019	**0.482**±0.011	**0.496**±0.008	**0.496**±0.011	**0.500**±0.005	**0.501**±0.003
MIT Movie	Neigh.Tag.†	0.031±0.020	0.045±0.019	0.041±0.011	0.053±0.009	0.054±0.007	0.086±0.008
	Example.†	**0.401**±0.011	**0.395**±0.007	**0.402**±0.007	**0.400**±0.004	**0.400**±0.005	**0.395**±0.007
	Prototype	0.175±0.007	0.168±0.006	0.170±0.004	0.174±0.003	0.173±0.002	0.173±0.002
	Prototype + CP	0.303±0.011	0.293±0.007	0.285±0.006	0.284±0.002	0.282±0.002	0.280±0.002
	Multi-Prototype	0.197±0.007	0.207±0.005	0.219±0.004	0.227±0.002	0.229±0.003	0.230±0.002
	Multi-Prototype + CP	0.364±0.020	0.368±0.011	0.380±0.006	0.382±0.003	0.354±0.003	0.383±0.002

Table 3.2: F1-score on training-free settings, *i.e.,* predicting novel entity types using nearest neighbor methods. The best results are in **bold**. † indicates results from [232, 198].

Training-Free Setting. Some real-world applications require immediate inference on unseen entity types. For example, novel entity types with a few examples are frequently given in an online fashion, but updating model weights $\boldsymbol{\theta}$ frequently is prohibitive. One may store some token examples as supports and utilize them for nearest neighbor classification. The setting is referred to as *training-free* in [198, 232], as the models identify new entities in a completely unseen target domain using only a few supporting examples in this new domain, without updating network parameters $\boldsymbol{\theta}$ in that target domain. Our prototype-based method

is able to perform such immediate inference. Two recent studies on training-free NER are: (*i*) *Neighbor-tagging* [198] which copies token-level labels from weighted nearest neighbors; (*ii*) *Example-based NER* [232] which is the SoTA on training-free NER, identifying the starting and ending tokens of unseen entity types.

We observed that our basic prototype-based method, under the training-free setting, does not gain from more given examples. We hypothesize that this is because tokens belonging to the same entity type are not necessarily close to each other, and are often separated in the representation space. Though it is hard to find one single centroid for all tokens in the same type, we assume that there exist local clusters of tokens belonging to the same type. To resolve such issue, we follow [35] and extend our method to a version called *Multi-Prototype*, by creating $K/5$ prototypes for each type given K examples per type. (*e.g.,* 2 prototypes per class are used for the 10-shot setting). The prediction score for a testing token belonging to a type is computed via averaging the prediction probabilities from all prototypes of the same type.

We compare with previous methods in Table 3.2 and observe that multi-prototype methods not only benefit from more support examples, but also surpass neighbor tagging methods and example-based NER by a large margin on two out of three datasets. For the MIT Movie dataset, one entity type can span a large chunk with multiple consecutive words in a sentence, which favors the span-based method like [232]. For example, the underlined part in the sentence "*what movie does the quote i dont think we are in kansas anymore come from*" is annotated as entity type `Quote`. The proposed methods in this paper can be combined with the span-based approach to specifically tackle this problem, and we leave it as future work. Further, if slightly fine-tuning is allowed, we see that the prototype-based method achieves 0.438 with 5-shot learning in Table 3.1, better than 0.395 achieved by example-based NER given 500 examples.

3.5 HIERARCHICAL FINE-GRAINED ENTITY TYPING

Despite the effectiveness of current prompt-based tuning pipeline, there are two notable limitations: (1) Current automatic verbalizer construction methods extract type representative words to be mapped to the label set based on general knowledge (*e.g.,* knowledge bases), but they are only effective when the type label names have concrete and unambiguous meanings (*e.g.,* "Organization/Company"). When the label names are too abstract or have composite meanings (*e.g.,* "Location/GPE" refers to geopolitical locations including countries and cities), however, suitable type indicative words cannot be easily generated only via external knowledge. Meanwhile, it is challenging to distinguish semantically similar types

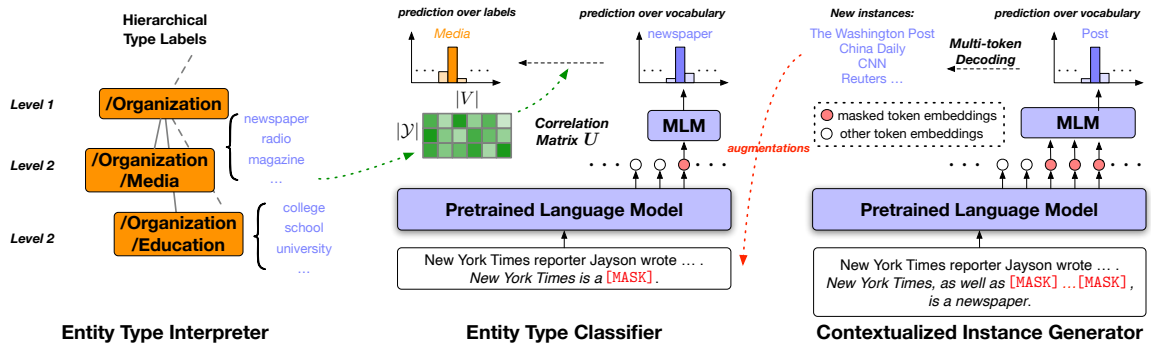


Figure 3.4: Overall framework of ALIGNIE . (Left): With a given type label hierarchy, an entity type interpretation module relates all the words in the vocabulary with the label hierarchy by a correlation matrix, so that the top-related words of "/Organization/Media" are "newspaper", "radio" and "magazine". (Middle): With an entity mention and its context from the training set, an entity typing classifier uses a template and an MLM head to predict word probability at the `[MASK]` position, and then maps the word distribution to type probability using the correlation matrix. The word "newspaper" has a high relevance with the label "/Organization/Media". (Right): After the entity typing classifier predicts a type for an entity mention in the few-shot samples, a type-based contextualized instance generator uses that entity mention and the predicted type to construct a template for new instance generation. The newly generated instances for "New York Times" are: "The Washington Post", "China Daily", *etc.*They are added back to the training set to improve the generalization of the entity typing classifier.

(*e.g.*, "Organization/Sports Team" and "Organization/Sports League") well for robust fine-grained type classification. (2) The current prompt-based fine-tuning approaches have not fully utilized the power of PLMs: PLMs are commonly used as the *representation models* that predict entity types based on entity instance representations. On the other hand, the *generation power* of PLMs acquired through extensive general-domain pretraining can be exploited to generate new entity instances that do not exist in the few-shot training samples so that the entity typing model can be trained with more instances for better generalization.

To address the above limitations, I propose two modules for prompt-based few-shot entity typing: (1) An *entity type label interpretation module* automatically learns to relate the hierarchical entity type labels to tokens in the vocabulary by jointly leveraging the given few-shot instances and the label hierarchy; (2) A *type-based contextualized instance generator* generates new instances based on few-shot instances and their types, providing the type classifier with more training samples. Our overall framework is illustrated in Fig. 3.4.

Hierarchical Entity Label Interpretation. I will introduce how to jointly leverage few-shot samples and the entity label hierarchy to learn a label interpretation module.

We consider a correlation matrix $\mathbf{U} \in \mathbb{R}^{|\mathcal{Y}| \times |V|}$ that characterizes the correlation between each word $w \in V$ and each type label $y \in \mathcal{Y}$. Specifically, each element $u_{y,w}$ (the y-th row and w-th column) in $\mathbf{U}$ represents a learnable correlation score between w and y. The correlation matrix maps the token predictions of PLMs to the final entity label predictions $p(y|\boldsymbol{h})$:

$$p(y|\boldsymbol{h}) = \sum_{w \in V} p(y|w) p(w|\boldsymbol{h}), \tag{3.7}$$

where

$$p(y|w) = \frac{\exp(u_{y,w})}{\sum_{y'} \exp(u_{y',w})}.$$

When Eq. (3.7) is trained via cross entropy loss on few-shot samples, the correlation matrix $\mathbf{U}$ will be updated to automatically reflect the corpus-specific word-to-type correlation. For example, consider an input text with "Ukraine" as a "GPE"-type entity, the output of the MLM head at the `[MASK]` position is very likely to give high probability to words like "country" or "nation", therefore the corresponding elements $u_{y,w}$ with y being "GPE" and w being "country" or "nation" will increase during training.

To further grant $\mathbf{U}$ a good initialization, we assign a higher initial value to those words that are one of the label names of an entity type. Specifically, the elements in $\mathbf{U}$ are initialized to be:

$$u_{y,w} = \begin{cases} \frac{1-\alpha}{|\boldsymbol{y}|} + \frac{\alpha}{|V|-|\boldsymbol{y}|} & w \in \boldsymbol{y} \\ \frac{\alpha}{|V|-|\boldsymbol{y}|} & w \notin \boldsymbol{y} \end{cases}, \tag{3.8}$$

where $\boldsymbol{y}$ denotes the label name set of label y; α is a hyperparameter controlling the initial bias to label name match (*e.g.,* when α is close to 0, words that do not match the label names will have a near-zero initialization).

Apart from the information from few-shot examples, the labels of an FET dataset typically forms a hierarchy that specifies the *inclusive* relationship between parent-child labels and the *exclusive* relationship between sibling labels. Therefore, we can further regularize the correlation matrix $\mathbf{U}$ with the hierarchical information. Following [129] that learns a spherical tree embedding given a hierarchical structure, we learn the correlation matrix by considering the relationship of nodes on the hierarchy.

To model the semantic inclusiveness between parent-child label pairs, we impose an inclusive loss to relate the parent label with its child labels by minimizing their cosine distance:

$$\mathcal{L}_{\text{inc}} = \sum_{y \in \mathcal{Y}} \left(1 - \cos\left(\boldsymbol{u}_y, \boldsymbol{u}_{\text{Parent}(y)}\right)\right), \tag{3.9}$$

where $\boldsymbol{u}_y$ refers to the y-th row in $\mathbf{U}$, and Parent(y) means the parent node of y in the label hierarchy. Eq. (3.9) encourages each label to have a similar correlation score distribution to all its children labels (*e.g.,* the "Organization" label is the broader type of its children labels "Company", "Sports Team", and "Sports League"; "Organization" will be thus regularized to share all the relevant words to any of its children labels).

Meanwhile, we also aim to distinguish sibling type categories like "Sports Team" and "Sports League" for better distinctive power at finer-grained levels. Therefore, we apply an exclusive loss that minimizes the cosine similarity between each pair of sibling labels sharing the same parent node in the hierarchy:

$$\mathcal{L}_{\text{exc}} = \sum_{\substack{y_i, y_j \in \mathcal{Y} \\ y_i \neq y_j}} \cos(\boldsymbol{u}_{y_i}, \boldsymbol{u}_{y_j}),\ \text{Parent}(y_i) = \text{Parent}(y_j). \tag{3.10}$$

Finally, the overall learning objective for the label interpretation module is:

$$\mathcal{L} = \mathcal{L}_{\text{ce}} + \lambda \mathcal{L}_{\text{exc}} + \lambda \mathcal{L}_{\text{inc}}, \tag{3.11}$$

where $\mathcal{L}_{\text{ce}}$ is the cross entropy loss; λ is the weight for regularization.

Contextualized Instance Generator. Previous prompt-based methods mainly utilize the *representation power* of PLMs: By learning contextualized representations of the input texts, PLMs are able to accurately predict the `[MASK]` tokens which are then mapped to the final entity typing predictions. In fact, in a token-level task like FET, not only can PLMs infer the type of an entity mention, but it also may generate new instances of specific entity types based on pretraining knowledge. In text classification studies [128], topic-related words are generated through pre-trained MLM for self-training. Similarly, we explore the *generation power* of PLMs to construct new instances using existing instances and their types as examples. For example, given an entity "Buffalo" and a sentence "At both Delaware State and **Buffalo**, ..., as both schools transitioned from lower levels of NCAA hierarchy", the MLM head in the typing classifier predicts the entity "Buffalo" as a *university*. Based on the predicted type, our goal is to generate new instances of the same type, such as other universities like "Duke" and "Dartmouth". These newly found instances can then be added back to the training set for better generalization of the typing classifier.

To fulfill the goal of generating same-type instances from an example entity $\boldsymbol{m}$, we need to first use the typing classifier to predict the fine-grained type of $\boldsymbol{m}$, denoted as t, and then

create a generative template $T_g(\boldsymbol{m}, t)$ to generate new instances:

$$T_g(\boldsymbol{m}, t) = \boldsymbol{m}, \text{ as well as } \texttt{[MASK]} \text{ , is a } t. \tag{3.12}$$

Using the above template, the MLM head will try to predict the instances with the same type t as entity mention $\boldsymbol{m}$.

The above proposed template can already generate single-token instances effectively, but will obviously miss a large amount of multi-token instances. For example, when we want to generate new instances for **New York Times**, the above method will only generate single-token newspaper names like "Reuters", "CNN", *etc.*Therefore, to generate multiple-token instances, we inject a sequence of `[MASK]` tokens into the template $T_g(\boldsymbol{m}, t)$.

$$T_g(\boldsymbol{m}, t) = \boldsymbol{m}, \text{ as well as } \texttt{[MASK]} \ldots \texttt{[MASK]} \text{ , is a } t. \tag{3.13}$$

Such a multiple-token instance generation process is necessary even for single-word entities, since they can be potentially sliced into multiple subwords by PLM tokenization (*e.g.,* "Chanel" will be sliced into "Chan" and "##el" in BERT tokenization), and using multiple `[MASK]` positions for prediction will give chance to the generation of less frequent words that consist of multiple subwords. Multi-token decoding is commonly carried out in autoregressive language models [151] as sequential left-to-right steps. However, it becomes less straightforward for autoencoder models like BERT that uses bidirectional contexts for prediction: While BERT can predict multiple `[MASK]` tokens in a sequence simultaneously by greedily picking the most likely token at each individual `[MASK]` position, these predictions are made independently from each other and the final results may not make sense as a whole. Therefore, we propose to fill in one blank at each step by selecting the word with the highest score at that position, and recursively predict the other blanks conditioned on the already filled blanks. Later we will score the multiple combinations of words and select the top-M combinations as the new instances. For example, using $T_g(\boldsymbol{m}, t)$ and decoding the masked tokens from left to right, we can generate a new instance that has the same type with **New York Times** as shown in Fig. 3.5.

New York Times, as well as the$_1$ [MASK][MASK] is a newspaper.
New York Times, as well as the$_1$ Washington$_2$ [MASK] is a newspaper.
New York Times, as well as the$_1$ Washington$_2$ Post$_3$ is a newspaper.

Figure 3.5: An example of the decoding steps.

Since there is no ground truth for the length of the new instances, we iterate from 1 to l

`[MASK]` tokens, where l is the length of the example entity. We then rank all generated new instances $\widetilde{\boldsymbol{m}}$ with pseudo log likelihood [81] to select top new instances:

$$\text{Score}(\widetilde{\boldsymbol{m}}) = \sum_{i=1}^{|\widetilde{\boldsymbol{m}}|} \log(s_i) \tag{3.14}$$

where s_i is the conditional probability of predicting the i-th sub-word based on previous $i-1$ generated sub-words.

Apart from probing the knowledge in PLMs to generate new instances, sometimes entity instances of the same type may appear in parallel. For example, the context "..., while Dan Oster, Anjelah Johnson, and Daheli Hall were hired as feature players" include multiple *players/actors* such as "Dan Oster", "Anjelah Johnson" and "Daheli Hall". In the few-shot learning setting where a limited number of training examples at instance-level are provided, it is likely that the above sentence, if appearing in the training set, only contains one annotated entity, and other entities in the same sentence will not appear in the training set. To find potential parallel entity instances in the sentence, we extend the template in Eq. (3.13) by adding the original context $\boldsymbol{x}$ to form our final instance generation template:

$$T_g(\boldsymbol{x}, \boldsymbol{m}, t) = \boldsymbol{x}.\ \boldsymbol{m}, \text{ as well as } \texttt{[MASK]}\ \ldots \texttt{[MASK]}\ , \text{ is a } t. \tag{3.15}$$

Using the above template, both in-context parallel instances and out-of-context new instances based on pre-training knowledge can be generated through the instance generator.

Adding New Instances for Training. For each given entity mention $\boldsymbol{m}$, we can generate M new instances after the training finishes half of the total epochs when the type words output by the MLM head become stable. The type-based new instances are then added back to augment the training set by applying the same classification template. Since there could be noise in generating new instances, we do not impose a hard label on a new instance $\widetilde{\boldsymbol{m}}$ to train the model. Instead, to improve robustness to label noise [118], we use a soft distribution over $\mathcal{Y}$ as its pseudo label $\widetilde{\boldsymbol{y}}$ by smoothing the label y_i of the original entity $\boldsymbol{m}$:

$$\widetilde{\boldsymbol{y}} = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{Y}|} & y = y_i \\ \frac{\epsilon}{|\mathcal{Y}|} & y \neq y_i \end{cases}, \tag{3.16}$$

where ϵ is the smoothing parameter.

We use the KL divergence loss to train the model on newly generated instances:

$$\mathcal{L}_{\text{new}} = \sum_{i=1}^{N}\sum_{k=1}^{M} \text{KL}(\widetilde{\boldsymbol{y}}_{i,k}||p(y|\widetilde{\boldsymbol{m}}_{i,k})), \tag{3.17}$$

where $\widetilde{\boldsymbol{m}}_{i,k}$ is the k-th instance generated from the i-th entity mention, and $\widetilde{\boldsymbol{y}}_{i,k}$ is its corresponding label.

Overall Training Objective. The overall training objective is the combination of training the label interpretation module and using new generated instances as the augmentations:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{ce}} + \lambda\mathcal{L}_{\text{exc}} + \lambda\mathcal{L}_{\text{inc}} + \beta\lambda_n\mathcal{L}_{\text{new}} \tag{3.18}$$

During the first half of training epochs, β is set to 0 to avoid generating false types for existing entities. After half of the training epochs finish, the model becomes more stable on predicting types, and we gradually increase β as the epoch grows: $\beta = \frac{2t-T}{T}$, where T is the total number of epochs and t is the current epoch number.

3.6 EVALUATION OF HIERARCHICAL ENTITY TYPING

This section demonstrates the effectiveness of ALIGNIE . Three benchmark datasets on FET are used and I report the main quantitative results and analyze some case studies.

Method	OntoNotes			BBN			Few-NERD		
	(Acc.)	(Micro-F1)	(Macro-F1)	(Acc.)	(Micro-F1)	(Macro-F1)	(Acc.)	(Micro-F1)	(Macro-F1)
5-Shot Setting									
Fine-tuning	28.60	50.70	51.60	51.03	60.03	58.22	36.09	48.56	48.56
Prompt-based MLM	32.62	60.97	61.82	67.00	75.23	73.55	44.69	59.24	59.24
PLET	48.57	70.63	75.43	71.23	79.22	78.93	56.94	68.81	68.81
ALIGNIE (- hierarchical reg.)	52.74	**77.55**	79.72	72.15	80.35	80.40	59.01	70.91	70.91
ALIGNIE (- new instances)	51.10	72.91	76.88	73.50	81.62	81.31	57.41	69.47	69.47
ALIGNIE	**53.37**	77.21	**80.68**	**75.44**	**82.20**	**82.30**	**59.72**	**71.90**	**71.90**
Fully Supervised Setting									
Fine-tuning	56.70	75.21	78.86	78.06	82.39	82.60	79.75	85.74	85.74
Prompt-based MLM	55.18	74.57	77.47	77.10	81.77	82.05	77.38	85.22	85.22

Table 3.3: Results on three entity typing benchmark datasets. For 5-shot setting, we report the average performance over 5 different sets of randomly-sampled few-shot examples.

Quantitative Results. We apply the widely-used metrics from [110] consisting of strict accuracy (Acc.), loose micro-F1 score (micro-F1), and loose macro-F1 score (macro-F1). The

loose F1 scores tolerate partial correctness for type labels within the same branch but of different granularities.

Table 3.3 presents the performance of all methods on three benchmark datasets. Overall, prompt-based results have higher performance than vanilla fine-tuning in few-shot settings, showing the prompts are better at inducing factual knowledge from PLMs. In fully supervised settings, however, fine-tuning performs a little better than prompt-based MLM, which is also suggested by previous studies [58], because vanilla fine-tuning has an extra linear layer to learn more features from the full training set. Specifically, ALIGNIE achieves the best among all prompt-based methods on all three datasets, and is mostly better than the two ablations, demonstrating that the two proposed modules can effectively find related words and generate new instances for each label. We also notice that ALIGNIE can perform on par with fully supervised setting on **OntoNotes** and **BBN**, but cannot on **Few-NERD**. This is because the training set of **OntoNotes** and **BBN** are automatically inferred from external knowledge bases, and can contain much noise, while the training set of **Few-NERD** is totally labeled by human.

We further showcase some new instances generated based on few-shot examples in **Few-NERD** in Table 3.4. Four single-token instances and seven multiple-token instances are shown respectively. The generated results are of reasonable correctness and rich variety, thus ensuring the quality of the newly added training data. We can also found that the generator is able to capture in-context instances like "Zeus", "Hades" in the first example and "3DS" in the last example, which are also important to improve the context-based entity typing model.

Hyperparameter Study. We study the effect of several hyperparameters in ALIGNIE . We conduct all the hyperparameter study and case studies on **Few-NERD** dataset. We first separately vary the initialization bias α in Eq. (3.8) and the label smoothing parameter ϵ in Eq. (3.16) in range $[0.0, 0.5]$, while keeping the other's value as default (0.1 for both default values). We report the accuracy on **Few-NERD** in Fig. 3.6a. Overall, the performance is insensitive to α since the values in correlation matrix U are trainable. The accuracy also does not change much when ϵ is reasonably close to 0, and can degrade a little when too much weight is given away from the original label. We then vary the number of new instances M generated added to training, and run experiments over $M = [1, 3, 5, 8, 10]$. We present the results in Fig. 3.6b and found that the model performance increases as more new instances are generated. However, the increasing trend slows down at $M = 5$. Since generating more new instances cost extra time, it is already sufficient for us to generate 5 instances per type during training.

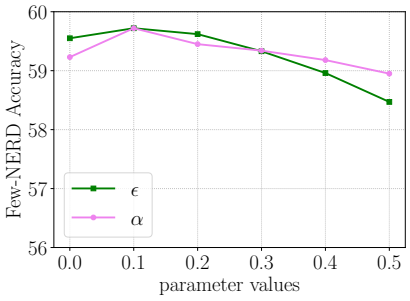
Generation from **single-token** entities		
Context & **entity mention**	MLM predicted type	Generated new instances
From Coeus and Phoebe came **Leto** and Asteria, who married Perses, producing Hekate, and from Cronus and Rhea came Hestia, Demeter, Hera, Poseidon, Hades, and Zeus.	god	Zeus, Hermes, Hades, Apollo, Athena, Hera, Pluto, Prometheus, ...
Orsonwelles malus is a species of spider endemic to **Kauai** in the Hawaiian Islands.	island	Hawaii, Fiji, Samoa, Guam, Taiwan, Honolulu, Japan, ...
At both Delaware State and **Buffalo**, Townsend was responsible for leading the athletic department to achieve full NCAA Division I status, as both schools transitioned from lower levels of NCAA hierarchy.	university	Delaware, Wilmington, Syracuse, Albany, Rutgers, Dartmouth, Duke, ...
A well-known philanthropist, Shaw donated billions of Hong Kong **dollars** to educational institutions in Hong Kong and mainland China.	currency	yuan, euros, sterling, yen, bitcoin, pounds, ...
Generation from **multi-token** entities		
Context & **entity mention**	MLM predicted type	Generated new instances
The album also included the song "Vivir Lo Nuestro," a duet with **Marc Anthony**.	singer	Beyonce, Jennifer Lopez, Rihanna, Taylor Swift, Lady Gaga, Michael Jackson, ...
The film was released on August 9, 1925, by **Universal Pictures**.	company	Warner Brothers, Paramount Pictures , Columbia Pictures, Lucasfilm, Hollywood Pictures, ...
Everland hosted 7.5 million guests in 2006, ranking it fourth in Asia behind the two **Tokyo Disney Resort** parks and Universal Studios Japan, while Lotte World attracted 5.5 million guests to land in fifth place.	park	Lotte World, Universal Studios Japan, Shanghai Disney World , Orlando Universal Studios, ...
Their daughter, Caroline Montgomery Marriott, was killed by the **Spanish Flu** epidemic in 1918.	disease	Yellow Fever, bird flu, typhus, small pox, polio, ...
The disappearance of the Norse settlements probably resulted from a combination of the Little Ice Age 's cooling temperatures, abandonment after the **Black Plague** and political turmoils.	disaster	climate change, Ice Age, global warming, Norman Conquest, cannibalism, natural disasters, ...
The site of Drake's landing as officially recognised by the **U.S. Department of the Interior** and other agencies is Drake's Cove.	government agency	the Department of Homeland Security, the Bureau of Land Management, the Federal Bureau of Investigation, the United States Forest Service, the National Institutes of Health, ...
Pikmin also make a cameo during the process of transferring downloadable content from a **Nintendo DSi** to a 3DS, with various types of Pikmin carrying the data over.	handheld	3DS, 2DS, Wii U, Nintendo Switch, the PSP, PlayStation Vita, ...

Table 3.4: Top generated instances based on predicted types of example entities.

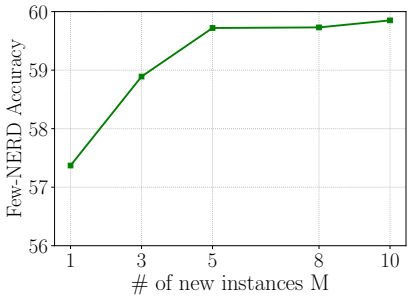
Organization	Organization /Company	Organization /Government	Organization /Sports League	Organization /Sports Team	Organization /Education	Organization /Media_Newspaper	Organization /ShowOrganization
association	corporation	bureau	match	teams	school	channel	orchestra
club	companies	agency	matches	players	university	television	choir
NGO	firm	bank	competition	squad	institution	newspaper	band
group	retailer	court	tournament	TEAM	college	magazine	bands
formation	airline	army	championship	side	Academy	broadcast	ensemble

Table 3.5: Top 5 related words of types on the "Organization" branch.

Effects of Templates. Previous studies [52, 224] point out that the quality of prompt templates play an important role in the performance of prompt-based methods. To study the impact of different template choices, we replace our template with two more templates and show their performance in Table 3.6. The results indicate that the choice of templates can impact the final performance, and the result is better when the typing template T_c matches more with the generating template T_g.



(a) Accuracy on Few-NERD dataset with different smoothing parameter ϵ and initialization bias α applied.



(b) Accuracy on Few-NERD dataset with different numbers of instance generated per type.

Figure 3.6: Hyperparameter study and testing curve analysis.

Typing template $T_c(\boldsymbol{x}, \boldsymbol{m})$	Instance generating template $T_g(\boldsymbol{x}, \boldsymbol{m}, t)$	Acc.	Micro-F1	Macro-F1
$\boldsymbol{x}$. $\boldsymbol{m}$ is a [MASK].	$\boldsymbol{x}$. $\boldsymbol{m}$, as well as [MASK], is a t.	59.72	71.90	71.90
$\boldsymbol{x}$. In this sentence, $\boldsymbol{m}$ is a [MASK].	$\boldsymbol{x}$. $\boldsymbol{m}$, as well as [MASK], is a t.	58.57	71.69	71.69
$\boldsymbol{x}$. $\boldsymbol{m}$ is a type of [MASK].	$\boldsymbol{x}$. $\boldsymbol{m}$, as well as [MASK], is a type of t.	58.86	72.05	72.05

Table 3.6: Performance with different templates on the Few-NERD dataset.

3.7 SUMMARY

In this chapter I introduce coarse and fine-grained entity recognition tasks, along with two frameworks for few-shot learning setting. I first introduce a systematic study for few-shot NER with three distinct directions, and then focus on hierarchical settings where type labels are more fine-grained. Pre-trained models are leveraged in both studies for representations of entity instances and type labels. Experiments also demonstrate that our proposed frameworks are very effective, and outperform previous baselines by large margins on benchmark datasets.

Recently, a bunch of instruction-tuning based methods [229, 83] are designed to align language models on specific tasks with hundreds/thousands of human-written question answer pairs. However, these methods still require human annotations on at least a few thousand training pieces, and this may not be always available on domain-specific data which require expertise. My method provides an approach to reduce the annotation burden, by leveraging the domain corpus itself for enhancing model performance.

CHAPTER 4: KNOWLEDGE-GUIDED COMMONSENSE REASONING

Apart from extracting static knowledge such as entities, relations and document types from a corpus, I also work on more complex user-oriented tasks such as commonsense reasoning that relies on dynamically utilizing the static knowledge. For example, a question asking about "Would Albert Einstein be more likely to know about astrology than physics?" requires entity information about the physicist Albert from science-related documents. My research [75] aims to outputting a generalized model that can answer various styles of reasoning questions given a few examples of in-domain questions and answers with explanations.

In the rapidly evolving domain of NLP, Large Language Models (LLMs) have shown their emergent abilities to reason and learn from limited examples. While previous methodologies have primarily centered around enhancing LLM performances with different prompting strategies, my research introduces a novel approach, named as **LMSI**, aiming to bridge the gap between the inherent capabilities of LLMs and the human brain's metacognitive processes, offering insights into how LLMs can self-enhance their reasoning abilities without the need for supervised data. The subsequent sections provide a comprehensive analysis of the proposed method, its empirical validation, and its potential implications for the broader NLP community.

4.1 OVERVIEW

Previous methods [195, 230, 91] for few-shot reasoning tasks usually focus on creating better demonstration formats for language models to imitate on new test-time qustions, leveraging their pre-trained generation power. For example, [195] proposes to prompt the language model with few-shot examples of questions, answers, and detailed reasoning text sequences, so that the final answer can be computed based on intermediate reasoning paths. Orthogonal to their direction, my proposed method **LMSI** [75] focuses on improving upon the pre-trained model itself for general reasoning tasks via self-improving. Specifically, given that diverse decoding paths [190] show a performance gap on the final answer accuracy over greedy decoding, my method leverages enormous unsupervised training questions to generate multiple reasoning paths with confidence-based sampling. Then the sampled reasoning paths are transferred into mixed formats/styles for self-training on the language model. The model after such unsupervised training can improve its generic reasoning performance on both in-domain and out-of-domain tasks, including commonsense reasoning, arithmetic reasoning, and sentence entailment.

Scaling has enabled Large Language Models (LLMs) to achieve state-of-the-art performance on a range of Natural Language Processing (NLP) tasks [186, 185, 152]. More importantly, new capabilities have emerged from LLMs as they are scaled to hundreds of billions of parameters [194]: in-context few-shot learning [13] makes it possible for an LLM to perform well on a task it never trained on with only a handful of examples; Chain-of-Thought (CoT) prompting [195, 91] demonstrates strong reasoning ability of LLMs across diverse tasks with or without few-shot examples; self-consistency [190] further improves the performance via self-evaluating multiple reasoning paths.

Despite these incredible capabilities of models trained on large text corpus [13, 24], fundamentally improving the model performances beyond few-shot baselines still requires finetuning on an extensive amount of *high-quality supervised* datasets. FLAN [193, 25] and T0 [162] curated tens of benchmark NLP datasets to boost zero-shot task performances on unseen tasks; InstructGPT [142] crowd-sourced many human answers for diverse sets of text instructions to better align their model to human instructions. While significant efforts were committed on collecting high-quality supervised datasets, human brain, on the contrary, is capable of the metacognition process [44], where we can refine our own reasoning ability without external inputs.

In this chapter, I will introduce how an LLM capable of in-context few-shot learning and chain-of-thought reasoning, is able to *self-improve* its reasoning ability without supervised data. I show that using only input sequences (without ground truth output sequences) from multiple NLP task datasets, a pre-trained LLM is able to improve performances for both in-domain and out-of-domain tasks. Our method is shown in Figure 4.1: I first sample multiple predictions using few-shot Chain-of-Thought (CoT) [195] as prompts, filter "high-confidence" predictions using majority voting [190], and finally finetune the LLM on these high-confidence predictions. The resulting model shows improved reasoning in both greedy and multi-path evaluations. We call the model fine-tuned in this way as **Language Model Self-Improved (LMSI)**. Note that LMSI depends on in-context few-shot learning and chain-of-thought reasoning abilities which small language models do not necessarily have. I empirically verify **LMSI** using a pre-trained PaLM 540B LLM, where our method not only improves training task performances (74.4%→82.1% on GSM8K, 78.2%→83.0% on DROP, 90.0%→94.4% on OpenBookQA, and 63.4%→67.9% on ANLI-A3), but also enhances out-of-domain (OOD) test tasks (AQUA, StrategyQA, MNLI), achieving state-of-the-art performances in many tasks without relying on supervised ground truth answers. Lastly, I conduct preliminary studies on self-generating additional input questions and few-shot CoT prompts, which could further reduce the amount of human effort required for model self-improving, and ablation studies on important hyperparameters of our approach. We hope our simple approach and

strong empirical results could encourage more future work by the community to investigate optimal performances of pretrained LLMs without additional human supervision.

4.2 RELATED WORK

Learning from Explanations. Augmenting a machine learning model with explanations has been studied in existing literature extensively. For example, in the supervised learning setting, a model can be fine-tuned using human-annotated rationales [217, 109, 139, 16, 29, 25]. A few works have also looked at how explanations can help the models in various settings, *e.g.,* in-context learning [95] and in distillation [150]. [103] treat explanations as process supervision to train a reward model. In this chapter, I focus more on the *unsupervised learning* setting, where no rationale-augmented training dataset is available, since human-annotated rationales can be expensive.

Few-Shot Explanations Improves Reasoning in LLMs. Recently, a lot of progress has been made towards improving LLMs' reasoning abilities via prompting or in-context learning. [195] propose Chain-of-Thought prompting, which prompts the language model to generate a series of natural-language-based intermediate steps, and show it can help language models better solve complex and multi-step reasoning tasks. [190] improve Chain-of-Thought prompting by sampling multiple diverse reasoning paths and finding the most consistent answers via majority voting. [91] propose to prompt the language model with "Let's think step by step" to generate reasoning in a zero-shot fashion. [230] decompose the questions into multiple sub-questions, and ask the language model to solve each sub-question sequentially.

Refining Explanations. More recent work proposes to further refine the generated reasoning paths as some of them could be unreliable. For example, [209] calibrate model predictions based on the reliability of the explanations, [87] show that inducing a tree of explanations and inferring the satisfiability of each explanation can further help judge the correctness of explanations. [101] show that sampling a diverse set of prompts from the training data, and a voting verifier can be used to improve model's reasoning performance. [200] and [226] propose to polish the problem progressively before the model reaching a stable answer. Our work is orthogonal to these lines of work, as we utilize refined explanations for model self-improvement, and could readily incorporate these other refinement techniques for generating higher-quality self-training data. Our work is closely related to [218] where we both propose to fine-tune a model on self-generated CoT data, but our method does not require ground truth labels and shows stronger empirical results with multi-task generalization.

Self-Training Models. One related line of work is self-training (see a survey from [4]). The key idea is to assign pseudo labels from a learned classifier to unlabeled data, and use these pseudo-labeled examples to further improve the original model training, e.g., [159, 201, 67]. Different from such prior work, our proposed self-improvement framework uses CoT prompting plus self-consistency to obtain high-confidence solutions on a large set of unlabeled data to augment the fine-tuning process.

Distillation and Dark Knowledge. Our method also tangentially relates to rich literature on distillation [7, 69]. A key detail is to learn from soft targets instead of hard predicted labels, as softmax outputs with a high temperature reveal more detailed relative class likelihoods, colloquially known as *dark knowledge* [69, 93]. Recent studies [218, 171, 45] show that *dark knowledge* within LLMs can be retrieved with more computation at inference time, such as adding informative instructions into the input sequence and output CoT generation [195, 91]. Recent works [121, 39, 70] demonstrated that distillation on explanations generated from large models can increase the reasoning abilities of smaller models with ground truth filtering.

4.3 SELF-IMPROVING MACHINE REASONING WITH LANGUAGE MODELS

The overview of our method is illustrated in Fig. 4.1: We are given a pre-trained Large Language Model (LLM) M and a question-only training dataset $\mathcal{D}^{\texttt{train}} = \{x_i\}_{i=1}^{D}$ with few-shot Chain-of-Thought (CoT) examples [195]. We apply multiple path decoding with a sampling temperature $T > 0$ for generating m reasoning paths and answers $\{r_{i_1}, r_{i_2}, \ldots, r_{i_m}\}$ for each question x_i in $\mathcal{D}^{\texttt{train}}$, and use majority voting (self-consistency) to select the most consistent, highest confidence answer [190]. We then keep all reasoning paths that lead to the most consistent answer, apply mixed formats of prompts and answers for augmentation, and fine-tune the model on these self-generated reasoning-answer data. We consider our approach as making the model self-improve. In the following sections, we detail important designs within our method, along with additional approaches for the model to self-improve without supervised data.

Generating and Filtering Multiple Reasoning Paths. Self-consistency [190] brings large improvements on reasoning tasks (e.g., 56.5% $\rightarrow$ 74.4% on GSM8K test set), and the gap between greedy decoding and diverse decoding shows there is a potential for further improving the reasoning ability of M, using the self-selected high-confidence reasoning paths as training data.

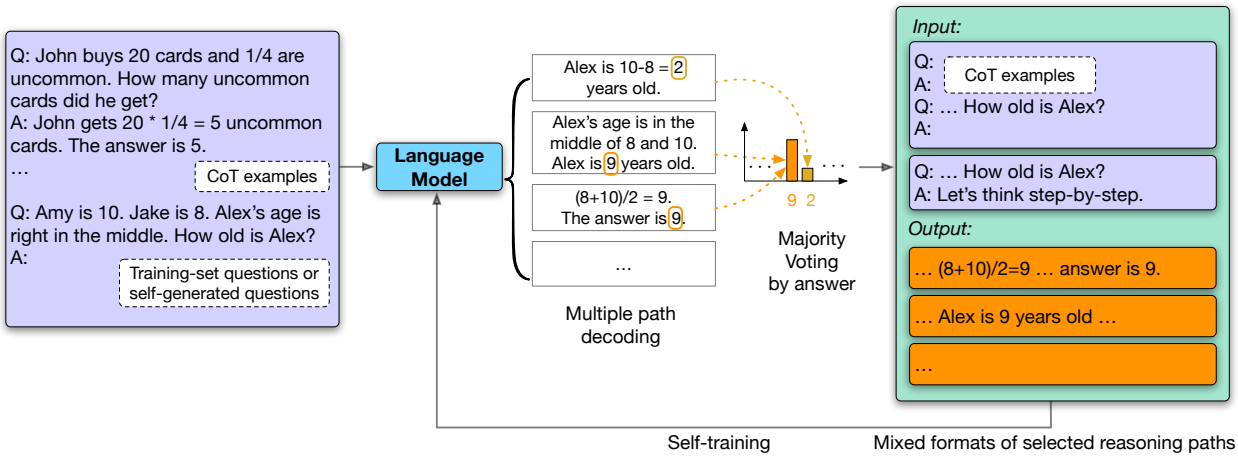


Figure 4.1: Overview of our method. With Chain-of-Thought (CoT) examples as demonstration [195], the language model generates multiple CoT reasoning paths and answers (temperature $T > 0$) for each question. The most consistent answer is selected by majority voting [190]. The "high-confidence" CoT reasoning paths that lead to the majority answer are augmented by mixed formats as the final training samples to be fed back to the model for fine-tuning.

Question: Stefan goes to a restaurant with his family. They order an appetizer that costs \$10 and 4 entrees that are \$20 each. If they tip 20% of the total, what is the total amount of money that they spend?
Multiple Path Decoding: **Output 1**: The appetizer costs \$10. The entrees cost \$20 each so in total 4 * \$20 = \$80. This means the total cost is \$10 + \$80 = \$90. They tip 20% of it, so the total amount they spend is \$90 * 1.2 = \$108. The answer is 108. **Output 2**: The appetizer costs \$10 and the entrees are \$20 each. There are 4 entrees so the sum is \$20 * 4 = \$80. The waiter gets 20% of the total. 20% of \$80 is \$80 * .2 = \$16. The answer is \$80 + \$16 = \$96. *(Incorrect reasoning path)* **Output 3**: The appetizer costs \$10. The entrees cost 4 * \$20 = \$80. The tip is 20% of the total, so it is 20% of the \$90 they have spent. The tip is 0.2 * 90 = \$18. The total they spent is \$90 + \$18 = \$108. The answer is 108.

Table 4.1: Examples of 3 self-generated CoT reasoning paths given a question. Output 1 and 3 are the most consistent reasoning paths based on majority voting and kept as self-training data.

For each training question x_i, we sample m reasoning paths, denoted as $\{r_{i_1}, \ldots, r_{i_m}\}$ (see Table 4.1 for examples). Since M is prompted with the CoT examples from [195], we apply the same output parsing with "The answer is" to generate their predicted answers $\{y_{i_1}, y_{i_2}, \ldots, y_{i_m}\}$. The most consistent answer, which is not necessarily a correct answer, is selected by majority voting, denoted as $\tilde{y}_i = \arg\max_{y_{i_j}} \sum_{k=1}^{m} \mathbb{I}(y_{i_j} = y_{i_k})$. For all the training

questions, we filter the CoT reasoning paths that reach $\tilde{y}$ as the final answer to be put into the self-training data, denoted as $\mathcal{D}^{\text{sc}} = \{x_i, \tilde{\boldsymbol{r}}_i\}$, where $\tilde{\boldsymbol{r}}_i = \{r_{i_j} | 1 \leq j \leq m, y_{i_j} = \tilde{y}_i\}$.

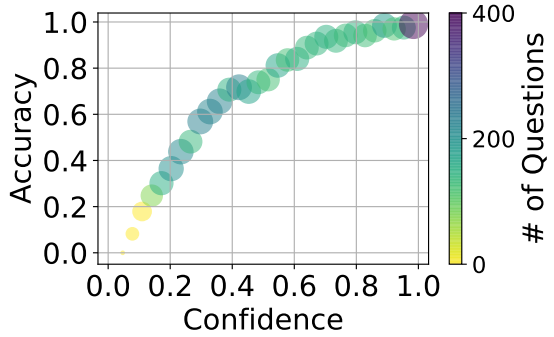


Figure 4.2: The relation of accuracy and confidence of the majority-voted answer after multiple path decoding on GSM8K training-set questions. Predicted confidence from self-consistency [190] is well calibrated [59].

Since we do not use any ground truth labels to filter out cases where $\tilde{y}_i \neq y_i$, it is important that the self-generated CoT reasoning paths are mostly reliable and incorrect answers do not hurt the self-improvement of the model. We plot the relation between the accuracy and confidence of self-generated CoT paths for each question in GSM8K training set in Fig. 4.2. The confidence is the number of CoT paths leading to $\tilde{y}$ divided by the total path number m. The y-axis shows the accuracy of $\tilde{y}$ under a certain confidence. The circle area and the color darkness shows the number of questions under a certain confidence. We can observe that confident answers are more likely to be correct, which means that when a question has many consistent CoT paths, then the corresponding $\tilde{y}$ is more likely to be correct. On the other hand, when $\tilde{y}$ is wrong, it is likely to be supported by fewer CoT paths, and brings little noise to the training samples.

Training with Mixed Formats. To prevent the language model from overfitting to specific prompts or answer styles, we create four different formats for each reasoning path to be mixed in the self-training data, shown in Table 4.2. In the first format, a few Chain-of-Thought examples (questions followed by reasoning paths leading to the correct final answers) are prepended to the new question, while the language model output is trained to be the same with the filtered CoT reasoning paths. In the second format, we use examples of questions and their direct answers as standard prompting, and the language model output is supposed to also only contain the direct answer. The third and fourth format are similar to the first and second format, except that no example of question-answer pairs are given, so that the model will learn to think on its own in an in-context zero-shot manner. In the third format,

Question: Amy is 10 years old. Jake is 8 years old. Alex's age is right in the middle. How old is Alex? **Selected Chain-of-Thought**: Amy is 10 years old. Jake is 8 years old. Alex's age is in the middle of Amy and Jake, so Alex is (8 + 10) / 2 = 9 years old. The answer is 9.
Mixed-formats of training data: **Format 1**: **Input**: *[CoT prompting examples]* + '\n' + *[Question]* + '\n' + 'A:' **Output**: Amy is 10 years old. Jake is 8 years old. Alex's age is in the middle of Amy and Jake, so Alex is (8 + 10) / 2 = 9 years old. The answer is 9. **Format 2**: **Input**: *[Standard prompting examples]* + '\n' + *[Question]* + '\n' + 'A:' **Output**: The answer is 9. **Format 3**: **Input**: *[Question]* + '\n' + 'A: Let's think step by step.' **Output**: Amy is 10 years old. Jake is 8 years old. Alex's age is in the middle of Amy and Jake, so Alex is (8 + 10) / 2 = 9 years old. The answer is 9. **Format 4**: **Input**: *[Question]* + '\n' + 'A:' **Output**: The answer is 9.

Table 4.2: An example of how a reasoning path is augmented into four formats of training data with different prompts (in input) and answer styles (in output). Specifically, the *CoT prompting examples* used for each tasks are listed in Appendix A.5. The *Standard prompting examples* are the same question-answer pairs with *CoT prompting examples*, except that reasoning is removed.

where we want the model to output CoT reasoning without prepending examples containing CoT reasonings, we append "Let's think step by step." at the end of the input sequence, to guide the language model to generate step-by-step CoT reasoning paths [91]. The mixed formats of training samples are then used to fine-tune the pre-trained language model M.

Generating Questions and Prompts. In some cases where even training questions or human-curated CoT prompts are limited, our method may not generate sufficient training samples for language model self-training. Therefore, we investigate how to self-generate more training questions as well as example prompts to further reduce human effort.

Previous work [210, 126] discuss few-shot data augmentation by generating diverse training samples using LLMs. However, those methods are designed for classification tasks and require ground truth label for each few-shot example. We use a simple yet effective approach to generate diverse questions (without using ground truth answers) from a few example questions. Specifically, we randomly sample and concatenate example questions in a random order as input prompt, and let the language model generate consecutive sequences as new questions. We repeat the process to obtain a large set of new questions, then use self-consistency [190] to only keep the questions that have a highly confident answer. Those questions are then used as self-generated training questions.

Given a set of questions, humans can write CoT examples as reasoning paths leading to

the final answer. In zero-shot setting without manual prompts, we can generate these CoT paths using the model itself. Following [91], we start the answer with "A: Let's think step by step." and let the language model generate the consecutive reasoning paths. We then use those generated reasoning paths as examples for few-shot CoT prompting.

4.4 EVALUATION OF SELF-IMPROVING LANGUAGE MODELS

We conduct a series of experiments to demonstrate the effectiveness of our proposed self-improving method. First, we apply our method on each individual dataset (task) and report the results. We then merge the generated data from all datasets and train one model to study the generalization ability of the model on unseen datasets as in [193]. In addition to the results of using generated CoT reasoning paths, we show studies on generating input questions and few-shot prompts. We end with ablation studies on model sizes and hyperparameters.

Tasks and Datasets. We demonstrate the effectiveness of our method on three types of tasks[2]:

- **Arithmetic reasoning**: We use the math problem set GSM8K [29], and a reading comprehension benchmark DROP [43] which requires numerical reasoning. We follow [230] to partition the DROP dataset into football related and non-football related subsets for training.
- **Commonsense reasoning**: We use the OpenBookQA [130] dataset, and the AI2 Reasoning Challenge (ARC) [28] dataset. Note that for ARC, we only use the Challenge sub-set (ARC-c) in our experiments. Both datasets contain multiple-choice questions.
- **Natural Language Inference**: We use the Adversarial NLI (ANLI) [130] subsets, ANLI-A2 and ANLI-A3, which are the more challenging subsets compared to ANLI-A1. These datasets contain pairs of sentences with relations of entailment, neutral, or contradiction.

Main Results. We list the results of using the PaLM 540B model before and after **LMSI** in Table 4.3. For each model, during test time, we apply three separate prompting methods on all six datasets: standard-prompting, CoT-Prompting, and Self-Consistency. We observe that after **LMSI** , the performance of all three prompting methods increase by a large margin. We observe significant improvement, comparing self-consistency versus **LMSI** with self-consistency: +7.7% on GSM8K, +4.8% on DROP, +4.4% on OpenBookQA, and +4.5%

[2]We evaluate on the test sets of GSM8K, ARC, OpenBookQA, and ANLI, and the dev set of DROP (ground truth labels of the test set are not publicly available).

Prompting Method	w. or w/o **LMSI**	**GSM8K**	**DROP**	**ARC**-c	**OpenBookQA**	**ANLI**-A2	**ANLI**-A3
Standard-Prompting	w/o **LMSI**	17.9	60.0	87.1	84.4	55.8	55.8
	w. **LMSI**	32.2 (+14.3)	71.7 (+11.7)	87.2 (+0.1)	92.0 (+7.6)	64.8 (+9.0)	66.9 (+11.1)
CoT-Prompting	w/o **LMSI**	56.5	70.6	85.2	86.4	58.9	60.6
	w. **LMSI**	73.5 (+17.0)	76.2 (+5.6)	88.3 (+3.1)	93.0 (+6.6)	65.3 (+6.4)	67.3 (+6.7)
Self-Consistency	w/o **LMSI**	74.4	78.2	88.7	90.0	64.5	63.4
	w. **LMSI**	**82.1** (+7.7)	**83.0** (+4.8)	**89.8** (+1.1)	**94.4** (+4.4)	**66.5** (+2.0)	**67.9** (+4.5)

Table 4.3: Accuracy results on six reasoning benchmarks with or without **LMSI** using different prompting method.

on ANLI-A3. This shows that our proposed method is quite effective. Furthermore, the single path CoT-Prompting performance of **LMSI** is close to or even better than the multiple path Self-Consistency performance of the model without **LMSI** , showing that **LMSI** truly helps the language model learn from the multiple consistent reasoning paths. We also apply **LMSI** on a recently proposed public language model, UL2 (20B) [178], and show the results in Table 4.8. Compared to the 540B model (decoder-only), UL2 has a smaller scale, and a different architecture (encoder-decoder). We observe that for most datasets, **LMSI** still outperforms the original UL2 results, but the improvement is not as large as that on the 540B model.

	Self-training data	**AQUA**	**SVAMP**	**StrategyQA**	**ANLI**-A1	**RTE**	**MNLI**-M/MM
w/o **LMSI**	-	35.8	79.0	75.3	68.8	79.1	72.0/74.0
w. **LMSI**	GSM8K + DROP + ...	39.0 (+3.2)	82.8 (+3.8)	77.8 (+2.5)	79.2 (+10.4)	80.1 (+1.0)	81.8/82.2 (+9.8/+8.2)

Table 4.4: Comparison of CoT-prompting accuracy results on six Out-Of-Domain benchmarks with or without training on six In-Domain (GSM8K, DROP, ARC-c, OpenBookQA, ANLI-A2, ANLI-A3) training-set questions.

Multi-Task Self-Training for Unseen Tasks. To demonstrate the generalization ability of **LMSI** , we conduct experiments of self-training on a mixture of the training-set questions from the above six datasets (denoted as In-Domain tasks), then use the same model checkpoint for the evaluation on six Out-Of-Domain (OOD) tasks, as shown in Table 4.4. Of all the OOD tasks: (1) **AQUA** [109] and **SVAMP** [143] are arithmetic reasoning tasks; (2) **StrategyQA** [54] is a commonsense reasoning task; (3) **ANLI-A1** [130], **RTE** [31] and **MNLI-M/MM** [197] are natural language inference tasks.[3] Among these tasks, **AQUA**, **StrategyQA**, and **RTE** are significantly different from any In-Domain task, and have their own few-shot prompts. From Table 4.4, we observe that **LMSI** achieves higher accuracy

[3]We evaluate on the test set of SVAMP and ANLI, the dev set of MNLI and RTE (ground truth labels of the test sets are not publicly available). For StrategyQA we use the question-only set from [10].

results on all OOD tasks, showing that the overall reasoning ability of the language model is improved.

	Results on **GSM8K**	
	Std. Prompting	CoT Prompting
w/o **LMSI**	17.9	56.5
LMSI w/o CoT formats	23.6 (+5.7)	61.6 (+5.1)
LMSI w/ CoT formats	32.2 (+14.3)	73.5 (+17.0)

Table 4.5: Ablation study: w/ or w/o CoT reasoning paths as training format on GSM8K dataset.

Importance of Training with Chain-of-Thought Formats. We demonstrate the importance of training language models with Chain-of-Thoughts compared to training with only direct answers. In Table 4.5, we list the results of **LMSI** with all four formats, and the results of **LMSI** with only direct answer formats. The results clearly show that without the CoT formats, the language model can still self-improve, but the performance gain drops by a large amount compared to using all four formats.

	Questions used for Self-Training	**GSM8K**	
		CoT	SC
w/o **LMSI**	-	56.5	74.4
w. **LMSI**	Generated	66.2 (+9.7)	78.1 (+3.7)
w. **LMSI**	Training-set	73.5 (+17.0)	82.1 (+7.7)

Table 4.6: Accuracy on GSM8K test set after self-training on different question sets. Results are shown for both CoT-Prompting (CoT) and Self-Consistency (SC).

Pushing the Limit of Self-Improvements. We further explore the few-shot setting where there are only limited training questions in the target domain. On GSM8K, we sample 10 real questions as few-shot samples, and use the language model to generate more training questions. We then self-train the language model with these generated questions and list the results in Table 4.6. The results show that using self-generated questions still improves the reasoning ability of language models, but using the real training-set questions leads to better results.

We explore the situation where no in-domain CoT examples are provided for a task. We apply the Step-by-Step method [91] to generate CoT examples using the language model,

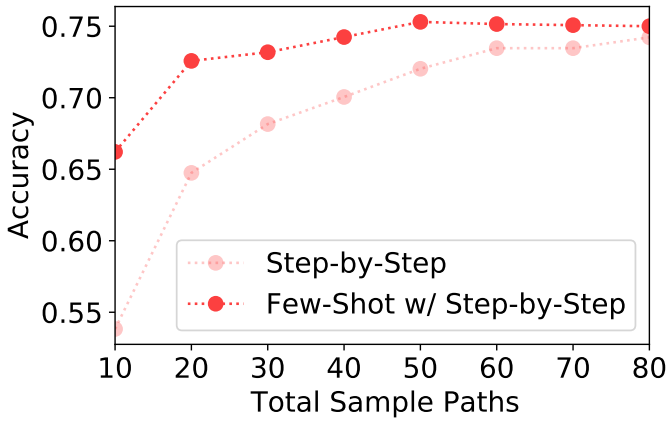


Figure 4.3: Accuracy results on GSM8K test set using PaLM 540B model with multi-path sampling and self-consistency [190]. "Step-by-Step" is the baseline performance of [91] plus self-consistency [190], while our "Few-Shot w/ Step-by-Step" uses exemplers self-generated from Step-by-Step (greedy decoding) for few-shot prompting the LLM.

and show the results in Figure 4.3. We observe that few-shot prompting with self-generated Step-by-Step CoT examples substantially outperforms the Step-by-Step [91] baseline (66.2% vs 53.8% at 10 paths, 74.2% vs 70.1% at 40 paths), and nearly matches the performance of human-written few-shot CoT [193] (74.4% at 40 paths [190]). The strong performance of "Few-Shot w/ Step-by-Step" despite the limited accuracy of prompt examples (43.0% for greedy Step-by-Step) likely comes from leveraging more diverse CoT prompts for multi-path decoding [101], where at 40 paths it uses 20 generate prompt-templates, each with 4-shot CoT examples, i.e. a total of 80 generated CoT examples compared to 8 human-written examples use in [195]. Since we did not use training questions or few-shot CoT examples, 74.2% also marks the new state-of-the-art zero-shot performance on GSM8K.

	Results on **GSM8K**		
	8 billion	62 billion	540 billion
w/o **LMSI**	5.0	29.7	56.5
Distilled from **LMSI**	33.4 (+28.4)	57.4 (+27.7)	-

Table 4.7: Distillation from PaLM 540B model to small models. We see that distilled smaller models outperform models that are one-tier larger.

Distillation to Smaller Models. We also explore whether the knowledge can be distilled to smaller models, such as in distillation [69] and in [218]. We use the same set of training

samples generated by the PaLM 540B model, but fine-tune on models with smaller sizes (PaLM 8B and PaLM 62B respectively), and show the results of CoT-prompting in Table 4.7. It is interesting to point out that after distillation from **LMSI** , the 62 billion model can outperform the pre-trained 540 billion model, and the 8 billion model can outperform the pre-trained 62 billion model. This implies that for downstream applications with limited computing resources, the reasoning knowledge from large models can be used to largely enhance small models to achieve competitive performance.

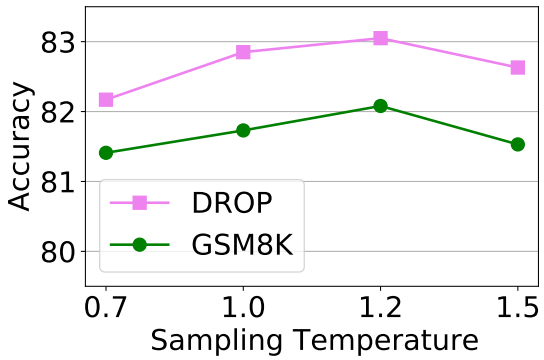


Figure 4.4: Accuracy results of **LMSI** on GSM8K and DROP test set when different sampling temperatures are applied for Self-Consistency.

Hyperparameter Study. We study the effect of varying the temperature T for multiple path decoding after **LMSI** is applied. Specifically, we vary T between $[0.7, 1.0, 1.2, 1.5]$ and show the results on GSM8K and DROP dataset respectively in Fig. 4.4. As shown in the figure, $T = 1.2$ benefits both datasets the most, and is used in the Self-Consistency method for **LMSI** on all datasets. We notice that the optimal T after model self-improvement is larger than the optimal $T = 0.7$ [190] before self-improvement. We believe the reason is that after training the model, the entropy of the output distribution is reduced.

We study whether the number of sampled reasoning paths m for Self-Consistency largely affects the accuracy after **LMSI** is applied. We show the accuracy on GSM8K test set for models both with or without **LMSI** in Fig. 4.5. For both cases, setting $m = 15$ already achieves a reasonably good accuracy, and using a larger m only brings marginal improvements. We also notice that after Self-Improvement, using 5 paths for Self-Consistency can already surpass the performance of using 32 paths for model without Self-Improvement. Thus, with a well-improved model, huge computing resources can be saved when applied to real applications.

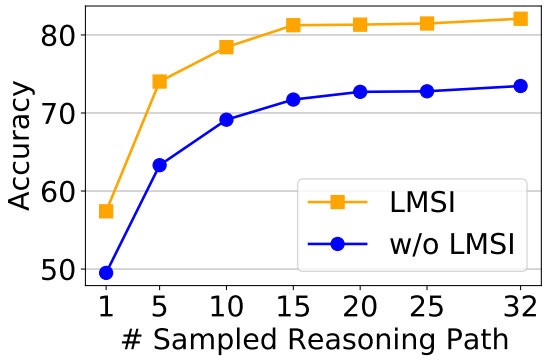


Figure 4.5: Accuracy results with or without **LMSI** on GSM8K test set using different numbers of sampled reasoning path for Self-Consistency.

4.5 DISCUSSIONS

Our approach mainly relies on the effectiveness of demonstration-based in-context few-shot learning which works most effectively on large language models, according to [194]. For example, [218] showed that a 6B model, GPT-J, achieves only 3.1% accuracy on GSM8K with few-shot CoT prompting, while GPT-3 (175 B) achieves 46.9%, according to [195]. Based on these existing studies, we believe that **LMSI** is more applicable to large-scale language models, since large models generate much more reliable reasoning paths for training. In addition, we have already shown that distillation from large models to small models are very promising. Therefore, smaller models can also be improved when large model APIs are accessible. We are fortunate to have enough resources for this work. Though the computation requirements for training large-scale language models are still prohibitively high for most researchers to conduct empirical studies along this line, we believe that our findings are conceptually useful for the NLP community by providing new insights for the properties of large language models.

	Prompting Method	**GSM8K**	**DROP**	**ARC-c**	**OpenBookQA**	**ANLI-A2**	**ANLI-A3**
w/o **LMSI**	CoT-Prompting	5.4/7.1	11.1/16.8	49.9	53.6	35.9	33.8
	Self-Consistency	6.4/9.9	16.8/26.5	54.7	54.0	37.4	36.8
LMSI	CoT-Prompting	6.1/8.6	11.4/17.1	50.9	53.8	35.4	34.4
	Self-Consistency	7.9/10.2	18.1/28.1	54.9	55.2	38.1	37.4

Table 4.8: Accuracy results on six reasoning benchmarks with **LMSI** on UL2. On GSM8K and DROP, we also include accuracy scores after an equation-correction postprocessing step.

We also apply **LMSI** on a recently proposed public language model, UL2 [178], using the

pre-trained model at step 2,650,000[4]. We use a fixed set of hyperparameters for fine-tuning on each dataset. Specifically, we generate $m = 40$ reasoning paths for each question in a training set for majority voting. We fine-tune the model for 10k steps with a learning rate of 5e−5 and a batch size of 32. For multiple path decoding, we use a sampling temperature of $T = 0.5$ with the pre-trained UL2 model following [178], and set $T = 0.7$ for the language model after **LMSI** . We set the maximum number of decode steps to 256 for all experiments.

The results are shown in Table 4.8. For arithmetic reasoning datasets, we follow [178] to provide both exact matching accuracy scores as well as accuracy scores after an equation-correction postprocessing step. We observe that for most datasets, **LMSI** still improves the reasoning accuracy (+1.6% on DROP, +1.2% on OpenBookQA, and +0.7% on ANLI-A2), but the improvement on UL2 is not as large as that on PaLM 540B. We think the reason is that, since **LMSI** exploits the implicit rationale of language models, and the capacity of a language model is determined by its size, larger models can capture more high-order semantics and are more likely to benefit from **LMSI** . For example, on the adversarial entailment tasks of ANLI (which is a three-class classification problem with labels "yes", "no", or "it is not possible to tell"), the UL2 model w/o **LMSI** only achieves an accuracy of marginally above 1/3, implying that the model is slightly better than doing random guess on this challenging task without any training. Our proposed **LMSI** can still improve the performance under this hard case by training on its implicit knowledge from self-generated paths.

4.6 SUMMARY

In this chapter I introduced an approach to improve knowledge-guided commonsense reasoning, by leveraging and self-improving large language models, without using ground truth supervision. By leveraging the training corpus, the model can refine its own generated responses, and can achieve competitive in-domain multi-task performances as well as out-of-domain generalization. Two other approaches are also studied for low-resource settings, where there are few training questions or no human given seed examples. For future work, one potential direction to improve **LMSI** is to integrate training data with self-generated data to further enlarge the diversity of training questions and responses.

Recently, instruction-tuning based methods have been used for improving LLM performance via human annotations on hundreds/thousands of annotations. To reduce the burden on human annotations, approaches that distill knowledge from LLMs to smaller language models have also been proposed, such as using GPT-4 generated responses for instruction-tuning [144].

[4]UL2: `https://github.com/google-research/google-research/tree/master/ul2`

Our proposed **LMSI** mainly focuses on improving the language model itself. Though being more applicable to large-sized language models, it can also be used to improve smaller language models as shown in our experiments.

CHAPTER 5: FUTURE WORK

The previous chapters concluded my completed research, and in this chapter I list my potential future direction to further enhance knowledge-based NLP applications.

5.1 IMPROVING MODEL FACTUALITY AND CALIBRATION

In Chapter 4, I introduce my work on using language models to do commonsense reasoning guided by a few human written prompts. However, Natural Language Generation (NLG) models are known to generate hallucinations [123], which could be factually incorrect statements or logically inconsistent sentences. It remains an open problem to determine whether the generated outputs are faithful or not.

Improving Factuality with Retrieval. To improve the facutality of language model outputs, one direction is to leverage information retrieved from external knowledge bases, such as Wikipedia and other online knowledge sources. There have been studies on using retrieval-based methods in pre-training or inference stage. For example, non-parametric language model [133] is proposed to let language models output responses according to documents from a certain corpus. Context-aware decoding [167] is proposed to make language models decode contrastively according to the provided pieces of context during inference stage, and thus can output less hallucination.

To improve the faculity of language model outputs, we can explore to develop verifier models that leverage related external knowledge to discriminate the truthfulness of model responses, and then decide whether or not to let the model re-generate new responses.

Improving Model Calibration. Another important factor of improving the usefulness of language model outputs for knowledge acquisition is to let them convey the degree of uncertainty about their statements, so users know how much to trust a given statement. This is also important for questions with no known ground truth (using LLMs for economic forecasts or open problems in science). A well-calibrated neural model provides a reliable confidence measure in addition to its prediction, which means the gap between average accuracy and average distance is small.

Three kinds of confidence/uncertainty expression can be provided by language models, as mentioned in [105]: verbalized expression ('61%' or 'medium confidence'), normalized answer logit, and the logit of "True" or "False" for each statement. [180] has found out that

generating multiple top guesses increase model calibration compared to only generating top-1 guess, since models tend to distribute lower confidence to answers that they are likely incorrect. For long-form generated responses, different output sequences might have the same/different meaning, so what matters is the model's uncertainty over meanings instead of uncertainty over tokens. This situation is studied by [94], and semantic variances are estimated by aggregating sequence probabilities.

One direction to improve language model calibration is to explore better confidence expressions. For example, it is found that using different numerical degree of confidence could affect the performance [231]. Moreover, language models tend to be overconfident when they are provided with supporting paragraphs to questions, so how to re-scale the confidence is also worthy of explorations.

Continual Learning and Knowledge Updating within Language Models. One major drawback of current language models is that after pre-training is done, the parameters are kept fixed and language models lack the ability to update themselves. Knowledge can evolve over time: for example, the answer to the question "Who is the president of the US?" might change every four years. Fine-tuning models on new data causes degradation of performance, thus how to edit model parameters to update the knowledge remains an open and challenging question. [124] develops a technique to identify and localize the parameters which are decisive for a piece of information, and only those corresponding weights are updated for the new knowledge. Another line of study [134] leverages an external memory network to store the edited facts.

5.2 IMPROVING MODEL COHERENCE AND CONSISTENCY

In real world practice, there could be many actionable tasks that contain long procedural or sequential subgoals, which require long term planning for humans to carry out solutions. While current models excel in generating coherent and contextually relevant sentences, they sometimes lack the ability for planning ahead as well as maintaining a consistent narrative or argument throughout longer passages. For example, a recent study [14] shows that GPT-4 cannot correctly perform the task of "Tower of Hanoi" with three rods.

Improving Decoding Strategy. One of the primary challenges in enhancing the global planning ability of language models is refining the decoding strategy. Traditional beam search or greedy decoding might not be sufficient for maintaining a consistent narrative over long passages. Most models, including the transformer-based architectures, rely on

autoregressive decoding, where each token is generated based on the preceding tokens. This often leads to locally coherent but globally inconsistent outputs. One potential direction is to utilize non-autoregressive decoding, such as using hierarchical decoding, where the model first generates an outline or skeleton of the content and then fills in the details [176]. This ensures that the overall structure is coherent, and the details align with the broader narrative. One could also explore parallel decoding strategies [206], where tokens are generated in parallel.

Incorporating Symbolic Meanings and World Models. Humans think with symbolic models, which are systematic and structured, but often rely on patterns or hand-engineering for model learning. On the other hand, language models are flexible with natural language, but lack ground reasoning with coherent plans. A recent study [199] integrates world knowledge, observations and questions into language model with a rational reasoning construction framework, by translating natural language expressions to formal languages using language models, and modeling thinking using probabilistic programming that combines a generative world model. Future works can also focus on how to leverage real-world structured data such as relations, graphs and tables in the planning process.

5.3 USER-SPECIFIC LANGUAGE MODELS

As language models have become more and more powerful, it is important to align their usage with human interest and values. One way is to improve their interactions with human beings.

Personalized Language Models. Current language models are mostly trained on public data, and can perform well on general tasks like commonsense reasoning. However, some applications require the model to be tailored to a specific organization or end user. For example, users may want a personalized recommendation for products or movies, as well as using language models to assist article writing in personalized styles. Current language models may not satisfy users' needs since they tend to output very general responses that appear frequently in the pre-training corpus. By training models on individual user data, language models can adapt to the user's style, preferences, and needs. A recent study [160] explores diverse tasks that require personalized language modeling, such as email drafting and tweet rephrasing. However, personalized language models could raise privacy concerns since the training data come from users. Ensuring data security and user anonymity could rely on techniques like differential privacy [1] to prevent the extraction of training data.

Language Models with Psychology. Emotions play a pivotal role in human decision-making and communication. Integrating emotional intelligence into LLMs can lead to more empathetic and contextually appropriate responses. Pre-trained models have been employed to detect depression and classify other mental disorders [80]. However, a deeper understanding requires the model to recognize complex emotional states and respond in a manner that acknowledges and respects the user's feelings. Emotionally-aware LLMs can be particularly useful in sensitive domains like mental health support or conflict resolution.

To further explore the integration of psychology and LLMs, future work can focus on developing models to adapt to the cognitive and emotional states of users, by acting as different supporting roles needed by users. Moreover, language models can be applied in therapy chatbots or mental health monitoring systems, and can generate diagnostic reports that therapists review before discussing with patients.

APPENDIX A: IMPLEMENTATION DETAILS OF PROPOSED METHODS

In this chapter, I list all the implementation details of the experiments conducted for evaluating proposed methods.

A.1 IMPLEMENTATION DETAILS FOR SET-COEXPAN

Dataset	# classes	# queries	entity vocabulary size	# documents
Wiki	8	40	41242	780556
APR	3	15	71707	1014140

Table A.1: Descriptions of datasets.

Datasets. We conduct our experiments on two large corpus also used in previous studies [165]: (1) **Wiki** is a subset of Wikipedia English dump from May 2011. (2) **APR** contains news articles published by two news platforms, Associated Press and Reuters in the year of 2015. We use the same sets of seed queries and ground truth of each semantic classes with previous studies as well. For **Wiki** dataset, there are 8 semantic classes, and 5 queries per class. For **APR** dataset, there are 3 semantic classes, and 5 queries for each class. These queries cover a wide range of semantic classes. Each query contains three seed terms belonging to a certain semantic class. Table A.1 lists the detailed descriptions of these datasets and queries. To extract possible entity mentions, we use the same preprocessing pipeline as SetExpan [165].

Parameter Settings. Since our framework iteratively generates auxiliary sets and co-expands target and auxiliary sets, we set the expansion number t at each iteration to be 5, and will later provide parameter study of t. In the auxiliary sets generation module, the embedding training balances the global context loss and local context loss by a ratio λ of 1.5, and we retrieve 10 related words for each element in the current expanded set. In the cross-seed merging stage, two groups can be merged together if they have enough overlap, thus we set η to be $\sqrt{min(g^i_{e,expanded}, g^j_{e,expanded})}$. In the multiple sets co-expansion module, since the size of generated auxiliary sets are often larger than the currently expanded target set, we need to balance the size of multiple sets to prevent the model from extracting features that only focus on making the largest set to be cohesive. We randomly sample seeds from each auxiliary set to keep it as large as the target set. When extracting skip-gram features for each entity mention, we first extract a fixed window size of 3, and then transform them into dynamic independent units, and there are two parameters in this transformation: the

breaking threshold γ is set to be 1.0, and the generalizing threshold k is set to be 100, which stops the original skip-gram to be broken down to too general contexts. We set $T = 10$ and $Q = 200$ as SetExpan [165]. We use the same hyperparameters for all queries on both datasets.

Compared Methods. We compare our methods to several previous corpus-based set expansion algorithms that only take corpus as input to expand entity seed sets. We also implement a baseline named BERT that only uses the pre-trained BERT model to retrieve the most similar entities. To show the effectiveness of auxiliary sets and flexible transformation of skip-grams in improving set expansion, we also compared Set-CoExpan with several ablations.

- SetExpan [165]: This method uses skip-gram features, distributed representations and coarse-grained types as context features. It uses rank ensemble to combine the result of three pre-ranking lists. We run the algorithm using its default setting.
- SetExpander [122]: This method incorporates multiple context features by training separate embedding space for each type of contexts, and a classifier is trained on a labeled dataset to adjust the weight of different features.
- CaSE [214]: This method combines skip-gram patterns and distributed representations to generate a one-time ranking for candidate entities. The model has three variants using different distributed representations, and we report the result of CaSE-W2V which has the highest performance.
- BERT [38]: We implement a baseline that only uses BERT, a recent contextualized embedding framework. We use the pre-trained model (uncased, base, 768 dimensions) and average over all occurrences in the corpus to get the centroid for any entity. We then use KNN to search for most similar entities.
- Set-CoExpan (no aux.): This ablation excludes the auxiliary sets generation module thus only expand one query of seed set in the co-expansion module.
- Set-CoExpan (no flex.): This ablation expands auxiliary sets simultaneously with the target set. However, it removes the flexible transformation of skip-grams.

A.2 IMPLEMENTATION DETAILS OF COREL

Datasets. Our experiments are conducted on two large real-world datasets: (1) **DBLP** contains around 157 thousand abstracts from publications in the field of computer science.

For preprocessing, we use AutoPhrase [164] to extract meaningful phrases to serve as our vocabulary, we further discard infrequent terms occuring less than 50 times, resulting in 16650 terms. (2) **Yelp** is collected from the recent released *Yelp Dataset Challenge*[5], containing around 1.08 million restaurant reviews. Similarly, we extract meaningful phrases and remove infrequent terms, resulting in 14619 terms[6].

Hyperparameter Study. For our relation classifier, the hyperparameter is set to be: batch size = 16, training epochs = 5, model: Bert-Base (12 layers, 768 hidden size, 12 heads). When training the relation classifier, we make a 90/10 training/validation split on training samples. In our relation transferring process, we set the threshold for relation score in Eqs. (2.11) and (2.12) to 0.7, and the threshold for KL divergence δ to 0.5. For our concept learning module, we set the following hyperparameters for embedding training process: embedding dimension = 100, local context window size = 5, $\lambda_d = 1.5$, and $\lambda_p = 1.0$. The threshold for Cluster Consistency is 0.5. We use the same hyperparameters for both datasets.

Compared Methods. We compare CoRel to several previous corpus-based taxonomy construction algorithms. To the best of our knowledge, there is no seed-guided topical taxonomy construction methods where each node is represented by a cluster of words, so we also compare CoRel with some unsupervised methods.

- Hi-Expan [166] + Concept Learning: Hi-Expan is a seed-guided instance-based taxonomy construction algorithm, which has the same input as our setting and constructs the taxonomy structure by set expansion and word analogy. Since its output node is represented by single word, we apply our concept learning module to enrich each node with a cluster of words.
- TaxoGen [220]: An unsupervised topical taxonomy construction method. It uses spherical clustering and local-corpus embedding to discover fine-grained topics represented by clusters of words.
- HLDA [11]: A non-parametric hierarchical topic model. It models the generation of documents in a corpus as sampling words from the paths when moving from the root node to a leaf node. Thus we can take each node as a topic.
- HPAM [132]: A state-of-the-art hierarchical topic model which requires a pre-defined number of topics and outputs topics at different levels.

[5]https://www.yelp.com/dataset/challenge

[6]Our code and data are available at `https://github.com/teapot123/CoRel`

A.3 IMPLEMENTATION DETAILS OF FEW-SHOT NER

Experiment Settings. Throughout our experiments, the pre-trained base RoBERTa model is employed as the backbone network. We investigate the following 6 schemes for the comparative study: (*i*) **LC** is the *linear classifier* fine-tuning method in Section 2, *i.e.,* adding a linear classifier on the backbone, and directly fine-tuning on entire model on the target dataset; (*ii*) **P** indicates the *prototype-based method* in Section 3.1; (*iii*) **NSP** refers to the *noisy supervised pre-training* in Section 3.2; Depending on the pre-training objective, we have **LC**+**NSP** and **P**+**NSP**. (*iv*) **ST** is the *self-training* approach in Section 3.3, it is combined with *linear classifier* fine-tuning, denoted as **LC**+**ST**; (*v*) **LC**+**NSP**+**ST**.

Datasets	CoNLL	Onto	WikiGold	WNUT	Movie	Restaurant	SNIPS	ATIS	Multiwoz	I2B2
Domain	News	General	General	Social Media	Review	Review	Dialogue	Dialogue	Dialogue	Medical
#Train	14.0k	60.0k	1.0k	3.4k	7.8k	7.7k	13.6k	5.0k	20.3k	56.2k
#Test	3.5k	8.3k	339	1.3k	2.0k	1.5k	697	893	2.8k	51.7k
#Entity Types	4	18	4	6	12	8	53	79	14	23

Table A.2: Statistics on the 10 public datasets studied in our NER benchmark.

Datasets. We evaluate our methods on 10 public benchmark datasets, covering a wide range of domains: OntoNotes 5.0 [153], WikiGold[7] [8] on general domain, CoNLL 2003 [161] on news domain, WNUT 2017 [36] on social domain, MIT Moive [112] and MIT Restaurant[8] [111] on review domain, SNIPS[9] [30], ATIS[10] [63] and Multiwoz[11] [15] on dialogue domain, and I2B2[12] [175] on medical domain. The detailed statistics of these datasets are summarized in Table A.2.

For each dataset, we conduct three sets of experiments using various proportions of the training data: 5-shot, 10% and 100%. For 5-shot setting, we sample 5 sentences for each entity type in the training set and repeat each experiment for 10 times. For 10% setting, we down-sample 10 percent of the training set, and for 100% setting, we use the full training set as labeled data. We only study the self-training method in 5-shot and 10% settings, by using the rest of the training set as unlabeled in-domain corpus.

Hyperparameters. We have described details for noisy supervised pre-training. For training on target datasets, we set a fixed set of hyperparameters across all the datasets:

[7] https://github.com/juand-r/entity-recognition-datasets
[8] https://groups.csail.mit.edu/sls/downloads/
[9] https://github.com/sonos/nlu-benchmark/tree/master/2017-06-custom-intent-engines
[10] https://github.com/yvchen/JointSLU
[11] https://github.com/budzianowski/multiwoz
[12] https://portal.dbmi.hms.harvard.edu/projects/n2c2-2014/

Entity type name	# Entities	Entity type name	# Entities
person	10976096	person/author	8603619
person/athlete	5990955	person/actor	5646333
person/fictional character	4820634	person/musician	4414457
person/ethnicity	1603194	person/politician	4196151
person/artist	3971716	person/director	870744
person/monarch	847943	person/soldier	297018
person/coach	284695	person/religious leader	272999
person/engineer	202533	person/architect	192587
person/doctor	101930	person/terrorist	2759
location	11234535	location/city	13478825
location/country	10022782	location/province	2555375
location/island	741533	location/body of water	1372583
location/county	930301	location/road	740874
location/astral body	410792	location/mountain	409878
location/cemetery	155498	location/park	78388
location/railway	61438	location/bridge	39528
location/glacier	17158		
organization	5280100	organization/company	8070793
organization/sports team	3236586	organization/educational institution	2124661
organization/government	1146508	organization/military	1118635
organization/political party	1006768	organization/sports league	854429
organization/news agency	378262	organization/government agency	314572
organization/airline	170127	organization/terrorist organization	40272
organization/fraternity sorority	35299		
art	3420964	art/music	4480181
art/written work	3284486	art/film	2583704
art/play	214837	art/newspaper	17488
building	2038445	building/sports facility	289182
building/airport	235172	building/theater	134604
building/hospital	89793	building/restaurant	50499
building/hotel	41571	building/library	24556
building/power station	18211	building/dam	10634
computer/algorithm	1808698	computer/programming language	110646
event	2877275	event/military conflict	1199857
event/attack	453078	event/election	358101
event/sports event	268484	event/natural disaster	149982
event/protest	93586	event/terrorist attack	2244
livingthing	2736344	livingthing/animal	1117967
product	969818	product/ship	776310
product/game	770000	product/instrument	622648
product/train	558943	product/software	550727
product/car	347887	product/airplane	321361
product/weapon	320842	product/spacecraft	56071
product/computer	50812	product/mobile phone	14585
product/engine device	31956	product/camera	10198
education/educational degree	507088	education/department	126584
medicine/symptom	440572	medicine/medical treatment	360235
medicine/drug	158258		
finance/currency	140008	finance/stock exchange	14861
broadcast program	1944347	broadcast/tv channel	91262
time	31543479	title	5752995
language	1565042	broadcast network	997300
food	903886	disease	743015
body part	741448	religion	666482
god	578091	chemistry	542973
award	488515	internet website	259798
law	230483	transit	132448
biology	123600	metropolitan transit line	92136

Table A.3: Entity type names and corresponding numbers on Wikipedia data used in supervised pre-training. For better visualization, we group entity label names belonging to the same root into the same blocks.

For the linear classifier, we set batch size = 16 for 100% and 10% settings, batch size = 4 for 5-shot setting. For each episode in the prototype-based method, we set the number of

Dataset	Entity type name	# Entities	Entity type name	# Entities
CONLL-2003	location	7140	person	6600
	organization	6321	misc.	3438
Onto	person	15429	countries, cities, states	15405
	organization	12820	date	10922
	cardinal	7367	political groups	6870
	money	2434	percent	1763
	ordinal number	1640	time	1233
	work of art	974	buildings,highways,bridges	860
	event	748	quantity	657
	product	606	language	304
	law	282	other locations	12
Wikigold	location	628	organization	559
	person	538	misc.	365
WNUT17	person	660	location	548
	group	264	corporation	221
	product	142	creative work	140
MIT Movie	plot	6468	actor	5010
	genre	3384	year	2702
	character name	1025	director	1787
	opinion	810	origin	779
	relationship	580	award	309
	quotation	126	soundtrack	50
MIT Restaurant	location	3817	cuisine	2839
	amenity	2541	restaurant name	1901
	dish	1475	rating	1070
	hours	990	price	730
SNIPS	(AddToPlaylist) playlist	1869	(AddToPlaylist) playlist owner	1107
	(AddToPlaylist) music item	887	(AddToPlaylist) artist	738
	(AddToPlaylist) entity name	590	(BookRestaurant) restaurant type	1359
	(BookRestaurant) party size number	1022	(BookRestaurant) time range	674
	(BookRestaurant) state	519	(BookRestaurant) city	513
	(BookRestaurant) restaurant name	339	(BookRestaurant) country	356
	(BookRestaurant) spatial relation	324	(BookRestaurant) party size description	316
	(BookRestaurant) served dish	269	(BookRestaurant) cuisine	210
	(BookRestaurant) sort	203	(BookRestaurant) facility	159
	(BookRestaurant) poi	143	(GetWeather) time range	1047
	(GetWeather) city	851	(GetWeather) country	498
	(GetWeather) state	491	(GetWeather) condition temperature	476
	(GetWeather) condition description	454	(GetWeather) geographic poi	290
	(GetWeather) current location	271	(GetWeather) spatial relation	209
	(PlayMusic) artist	1169	(PlayMusic) music item	791
	(PlayMusic) service	756	(PlayMusic) year	630
	(PlayMusic) sort	346	(PlayMusic) track	211
	(PlayMusic) album	176	(PlayMusic) playlist	149
	(PlayMusic) genre	144	(RateBook) rating value	1924
	(RateBook) rating unit	1103	(RateBook) best rating	1033
	(RateBook) object name	979	(RateBook) object select	952
	(RateBook) object type	919	(RateBook) object part of series type	307
	(SearchCreativeWork) object name	1951	(SearchCreativeWork) object type	1462
	(SearchScreeningEvent) movie name	808	(SearchScreeningEvent) object type	692
	(SearchScreeningEvent) movie type	674	(SearchScreeningEvent) spatial relation	665
	(SearchScreeningEvent) location name	586	(SearchScreeningEvent) object location type	458
	(SearchScreeningEvent) time range	265		

Table A.4: Entity type names and corresponding numbers on benchmark NER datasets CONLL-2003, Onto, WikiGold, WNUT17, MIT Movie, MIT Restaurant, and SNIPS.

sentences per entity type in support and query set (K, K') to be $(5, 15)$ for 100% and 10% settings, and $(2, 3)$ for 5-shot setting. For both training objectives, we set learning rate =

Dataset	Entity type name	# Entities	Entity type name	# Entities
ATIS	(fromloc) city name	4326	(fromloc) airport name	89
	(fromloc) state code	46	(fromloc) state name	39
	(fromloc) airport code	15	(toloc) city name	4343
	(toloc) state code	86	(toloc) state name	77
	(toloc) airport name	39	(toloc) airport code	20
	(toloc) country name	3	(depart date) day name	889
	(depart date) day number	395	(depart date) month name	379
	(depart date) today relative	84	(depart date) date relative	82
	(depart date) year	25	(depart time) time	692
	(depart time) period of day	593	(depart time) time relative	323
	(depart time) period mod	44	(depart time) start time	25
	(depart time) end time	25	(stoploc) city name	239
	(arrive time) time relative	187	(arrive time) time	172
	(arrive time) period of day	64	(arrive time) start time	21
	(arrive time) end time	20	(arrive time) period mod	4
	(arrive date) day name	88	(arrive date) month name	47
	(arrive date) day number	47	(arrive date) date relative	11
	(arrive date) today relative	2	(return date) date relative	10
	(return date) month name	4	(return date) day number	4
	(return date) today relative	1	(return date) day name	1
	(stoploc) state code	5	(stoploc) airport name	1
	(return time) period of day	3	(return time) period mod	2
	airline name	701	round trip	348
	cost relative	344	flight mod	329
	city name	227	class type	217
	flight stop	168	airline code	136
	flight number	84	fare basis code	76
	flight time	71	meal description	57
	fare amount	53	transport type	48
	connect	40	flight days	39
	airport name	38	economy	36
	airline name	32	aircraft code	31
	mod	30	airport code	29
	restriction code	23	meal	17
	state code	8	meal code	6
	day name	5	period of day	5
	days code	3	time	2
	today relative	2	state name	2
	month name	2	day number	2
	time relative	1		
Multiwoz	restaurant food	4041	restaurant name	3054
	hotel name	2863	restaurant booktime	2361
	attraction name	1972	train leaveat	1617
	train arriveby	1518	taxi departure	995
	taxi destination	970	taxi leaveat	800
	taxi arriveby	439	hospital department	107
	bus destination	3	bus departure	2
I2B2	date	5195	doctor	1989
	patient	931	hospital	925
	medical record	408	city	258
	phone	233	state	221
	username	219	street	208
	id num	174	profession	150
	zip	139	organization	85
	country	53	age	8
	device	7	tax	5
	other locations	4	email	3
	url	2	bioid	1
	health plan	1		

Table A.5: Entity type names and corresponding numbers on benchmark NER datasets ATIS, Multiwoz, and I2B2.

$5e^{-5}$ for 100% and 10% settings, and learning rate = $1e^{-4}$ for 5-shot setting. For all training data sizes, we set training epoch = 10, and Adam optimizer [89] is used with the same linear

decaying schedule as the pre-training stage. For self-training, we set $\lambda_{\mathtt{U}} = 0.5$.

Evaluation. We follow the standard protocols for NER tasks to evaluate the performance on the test set [161]. Since RoBERTa tokenizes each word into subwords, we generate word-level predictions based on the first word piece of a word. Word-level predictions are then turned into entity-level predictions for evaluation when calculating the f1-score. Two tagging schemas are typically considered to encode chunks of tokens into entities: BIO schema marks the beginning token of an entity as `B-X` and the consecutive tokens as `I-X`, and other tokens are marked as `O`. IO schema uses `I-X` to mark all tokens inside an entity, thus is defective as there is no boundary tag between same type of entities. In our study, we use BIO schema by default, but also report the performance evaluated by IO schema for fair comparison with some previous studies.

Entity Types of Training and Testing Datasets. We show the entity types and their corresponding frequencies in pre-training dataset in Table A.3 and downstream benchmark datasets in Table A.4 and Table A.5. We see that the entity types for pre-training and fine-tuning are semantically related, but different in granularity. For example, the location category in pre-training dataset contains fine-grained entity types like country, city, road, and bridge, while the Onto dataset for fine-tuning only gives a coarse-grained partition by geopolitical locations (countries, cities) and non geopolitical ones (highways, bridges). Further, for each categories of entity types, the pre-training dataset has a much higher frequency than fine-tuning dataset, allowing the model to learn heterogeneous contextual knowledge before deploying to a specific domain.

A.4 IMPLEMENTATION DETAILS OF ALIGNIE

Datasets. We use three fine-grained entity typing benchmark datasets.

- **OntoNotes**. The OntoNotes dataset is derived from the OntoNotes corpus [153] and 12,017 manual annotations were done by [56] with 3 hierarchical layers of 89 fine-grained types. We follow the dataset split by [168] that retains 8,963 non-pronominal annotations, where the training set is automatically extracted from Freebase API by [156]. Since our paper focus on few-shot learning, we apply extra preprocessing by filtering out classes with less than 5 annotations in training, validation, and test set, resulting in a total of 21 classes left.

- **BBN Pronoun Coreference and Entity Type Corpus** (BBN). This dataset uses 2,311 Wall Street Journal articles and is annotated by [196] with 2 hierarchical layer of 46 types. We follow the split by [156] and also filter out classes with less than 5 annotations in training, validation and test set, leading to a total of 25 classes.
- **Few-NERD**. The Few-NERD [41] dataset is a recently proposed large-scale manually annotated dataset with 2 hierarchical layers of 66 types. We follow [40] and uses the supervised setting of the dataset, FEW-NERD (SUP), as well as the official split for few-shot example sampling. All 66 classes have more than 5 annotations in each of the dataset splits.

Experiment Settings. We conduct 5-shot learning on three datasets by sampling 5 instances for training and 5 instances for dev set in each run of experiment. We repeat experiments for each dataset for 5 times with different sampled sets and report the average result. We use the pre-trained RoBERTa-base model as the backbone transformer model (for ALIGNIE and all baselines). For all three datasets: the max sequence length is set to be 128; the batch size is 8; the training epoch number is 30; the hyperparameters α and ϵ are set to 0.1; the instance generating number M is set to 5 per type; the training weights λ and λ_n are set to 1.0. We use a learning rate of $1e-2$ for the correlation matrix U and a gradient multiplication layer of $1e-7$ is applied to the bottom Transformer backbone so that the pre-trained weights stay very stable during the training. We use Adam [89] as the optimizer with linear decaying schedule. The model is run on NVIDIA GeForce GTX 1080 Ti GPU.

Compared Methods. We include the results of vanilla fine-tuning, prompt-based MLM using hard templates in [40], and the current state-of-the-art method **PLET** [40] on few-shot entity typing, which uses multiple hard and soft templates for prompt-based tuning, and automatically constructs verbalizers through external knowledge graph. We conduct ablation studies by removing the following two parts one at a time: (1) not using the hierarchical regularization in Eq. (3.10) and Eq. (3.9) (- hierarchical reg.); (2) not using the type-based contextual instance generator to generate new instances for training in Eq. (3.17) (- new instances). We also compare with vanilla fine-tuning and prompt-based tuning in a fully supervised setting.

Evaluation Metrics. We apply the widely-used metrics from [110] consisting of strict accuracy (Acc.), loose micro-F1 score (micro-F1), and loose macro-F1 score (macro-F1). The loose F1 scores tolerate partial correctness for type labels within the same branch but of different granularities.

A.5 IMPLEMENTATION DETAILS OF **LMSI**

Models, Training settings and Hyperparameters. We follow previous studies [195, 190] and conduct our experiments on an autoregressive Transformer-based language model with 540 billion parameters. The CoT examples for each dataset are listed below. We generate $m = 32$ reasoning paths for each question in a training set. For DROP and ANLI-A2/A3, we sample 5k examples for reasoning path generation to reduce the training burden; For other datasets, we keep the whole training set. For each dataset, we fine-tune the model for 10k steps with a learning rate of 5e−5 and a batch size of 32. We use a sampling temperature of $T = 0.7$ with the pre-trained model as suggested by [190]. We use $T = 1.2$ for the language model after self-improvement (**LMSI**). We set the maximum number of decoded steps to 256 for all experiments.

Chain-of-Thought Prompts for Each Dataset. We list the Chain-of-Thought Prompts for each dataset for "CoT-Prompting" experiments and self-generated training samples.

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?

A: We start with 15 trees. Later we have 21 trees. The difference must be the number of trees they planted. So, they must have planted 21 - 15 = 6 trees. The answer is 6.

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

A: There are 3 cars in the parking lot already. 2 more arrive. Now there are 3 + 2 = 5 cars. The answer is 5.

Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?

A: Leah had 32 chocolates and Leah's sister had 42. That means there were originally 32 + 42 = 74 chocolates. 35 have been eaten. So in total they still have 74 - 35 = 39 chocolates. The answer is 39.

Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?

A: Jason had 20 lollipops. Since he only has 12 now, he must have given the rest to Denny. The number of lollipops he has given to Denny must have been 20 - 12 = 8 lollipops. The answer is 8.

Q: Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?

A: He has 5 toys. He got 2 from mom, so after that he has 5 + 2 = 7 toys. Then he got 2 more from dad, so in total he has 7 + 2 = 9 toys. The answer is 9.

Q: There were nine computers in the server room. Five more computers were installed each day, from monday to thursday. How many computers are now in the server room?

A: There are 4 days from monday to thursday. 5 computers were added each day. That means in total 4 * 5 = 20 computers were added. There were 9 computers in the beginning, so now there are 9 + 20 = 29 computers. The answer is 29.

Q: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On wednesday, he lost 2 more. How many golf balls did he have at the end of wednesday?

A: Michael initially had 58 balls. He lost 23 on Tuesday, so after that he has 58 - 23 = 35 balls. On Wednesday he lost 2 more so now he has 35 - 2 = 33 balls. The answer is 33.

Q: Olivia has $23. She bought five bagels for $3 each. How much money does she have left?

A: She bought 5 bagels for $3 each. This means she spent 5 * $3 = $15 on the bagels. She had $23 in beginning, so now she has $23 - $15 = $8. The answer is 8.

Table A.6: Few-shot CoT prompts for GSM8K and SVAMP, from [195].

Q: Poison causes harm to which of the following? (a) a Tree (b) a robot (c) a house (d) a car

A: Poison will harm living things, only a tree is a living thing. The answer is (a).

Q: As you look deeper into a Marbel you can see (a) the future (b) minut defects (c) colors (d) the other side

A: Marbel is not transparent, so you can not see the other side. Marbel does not necessarily have multiple colors. You will see minut defects. The answer is (b).

Q: When food is reduced in the stomach (a) the mind needs time to digest (b) take a second to digest what I said (c) nutrients are being deconstructed (d) reader's digest is a body of works

A: The food is being deconstructed in the stomach during digestion. The answer is (c).

Q: The sun is responsible for (a) puppies learning new tricks (b) children growing up and getting old (c) flowers wilting in a vase (d) plants sprouting, blooming and wilting

A: The sun can affect the growing of living things, like plants. The answer is (d).

Table A.7: Few-shot CoT prompts for OpenBookQA, from [189].

Q: Since the 1970s, U.S. governments have negotiated managed-trade agreements, such as the North American Free Trade Agreement in the 1990s, the Dominican Republic-Central America Free Trade Agreement in 2006, and a number of bilateral agreements. In Europe, six countries formed the European Coal and Steel Community in 1951 which became the European Economic Community in 1958. Two core objectives of the EEC were the development of a common market, subsequently renamed the single market, and establishing a customs union between its member states. How many years did the European Coal and Steel Community exist?

A: According to the passage, the European Coal and Steel Community was established in 1951 and became the EEC in 1958. 1958 - 1951 = 7. So the answer is 7.

Q: In the county, the population was spread out with 23.50% under the age of 18, 8.70% from 18 to 24, 29.70% from 25 to 44, 24.70% from 45 to 64, and 13.30% who were 65 years of age or older. How many more percent are under the age of 18 compared to the 18 to 24 group?

A: According to the passage, 23.5% are under the age of 18, and 8.7% are from ages 18 to 24. 23.5% - 8.7% = 14.8%. So the answer is 14.8.

Q: Playing in their second straight Thanksgiving game, the Eagles struggled especially on defense, where they were unable to stop the much-hyped Lions offense. The worst of it all was how unproven rookie Eric Rowe was tasked with covering wide receiver Calvin Johnson, leading to Johnson catching 3 touchdowns. Stafford's five passing touchdowns, including three of them to Johnson was too much for the Eagles to overcome and for the second consecutive time this season, the Eagles gave up 45 points in a game. With the loss, the Eagles drop to 4-7 on the season and 6-1 when playing on Thanksgiving. How many TD passes did Stafford throw other than to Johnson?

A: According to the passage, Stafford threw 5 TD passes, 3 of which were to Johnson. 5 - 3 = 2. So the answer is 2.

Table A.8: Few-shot CoT prompts for DROP (nonfootball), from [230].

Q: The Seahawks played the San Francisco 49ers. In the first quarter, the Hawks RB Julius Jones got a 27-yard TD run, along with DT Craig Terrill returning a fumble 9 yards for a touchdown. In the third quarter, the 49ers almost rallied as RB H. J. Torres made a 12-yard TD pass to Lucas Nelly, along with Mare kicking a 32-yard field goal. In the final quarter, Julius Jones got another 11-yard TD. How many yards do the shortest touchdown run and the longest touchdown pass combine for?

A: All the touchdown runs are: a 27-yard touchdown run, a 9-yard touchdown run, a 11-yard touchdown run. The smallest number among 27, 9, 11 is 9. So the shortest touchdown run was 9 yards. All the touchdown passes are: a 12-yard touchdown pass. So the longest touchdown pass was 12 yards. So the shortest touchdown run and the longest touchdown pass combine for 9 + 12 = 21 yards. So the answer is 21 yards.

Q: The Steelers went home for a duel with the Baltimore Ravens. Pittsburgh would deliver the opening punch in the first quarter with a 1-yard touchdown from running back Rashard Mendenhall. The Ravens would make it even as running back Willis McGahee got a 9-yard TD. The Ravens kicker Billy Cundiff got a 45-yard field goal in the second quarter, concluding the first half with a 10-7 lead. The Steelers brought the game into overtime with a 38-yard field goal by Andrew Foster. The Ravens Billy Cundiff pulled off a winning 33-yard field goal in overtime. How many points did the Ravens have at halftime?

A: The Ravens kicker Billy Cundiff got a 45-yard field goal in the second quarter, concluding the first half with a 10-7 lead. So the Ravens had 10 points at halftime. So the answer is 10 points.

Q: The Vikings flew to Bank of America Stadium to face the Carolina Panthers. After a scoreless first quarter, Carolina got on the board with quarterback Matt Moore finding fullback Brad Hoover on a 1-yard TD pass. After yet another scoreless quarter, Carolina sealed the game as Matt Moore completed a 42-yard touchdown pass to wide receiver Steve Smith. How many scoreless quarters were there?

A: The first and third quarters were the scoreless quarters. So there are 2 scoreless quarters. So the answer is 2.

Table A.9: Few-shot CoT prompts for DROP (football), from [230].

Premise:
"Conceptually cream skimming has two basic dimensions - product and geography."
Based on this premise, can we conclude the hypothesis "Product and geography are what make cream skimming work." is true?
OPTIONS:
- yes
- no
- it is not possible to tell

A: Based on "cream skimming has two basic dimensions" we can't infer that these two dimensions are what make cream skimming work. The answer is it is not possible to tell.

Premise:
"One of our member will carry out your instructions minutely."
Based on this premise, can we conclude the hypothesis "A member of my team will execute your orders with immense precision." is true?
OPTIONS:
- yes
- no
- it is not possible to tell

A: "one of" means the same as "a member of", "carry out" means the same as "execute", and "minutely" means the same as "immense precision". The answer is yes.

Premise:
"Fun for adults and children."
Based on this premise, can we conclude the hypothesis "Fun for only children." is true?
OPTIONS:
- yes
- no
- it is not possible to tell

A: "adults and children" contradicts "only children". The answer is no.

Premise:
"He turned and smiled at Vrenna."
Based on this premise, can we conclude the hypothesis "He smiled at Vrenna who was walking slowly behind him with her mother." is true?
OPTIONS:
- yes
- no
- it is not possible to tell

A: the premise does not say anything about "Vrenna was walking". The answer is it is not possible to tell.

Premise:
"well you see that on television also"
Based on this premise, can we conclude the hypothesis "You can see that on television, as well." is true?
OPTIONS:
- yes
- no
- it is not possible to tell

A: "also" and "as well" mean the same thing. The answer is yes.

Premise:
"Vrenna and I both fought him and he nearly took us."
Based on this premise, can we conclude the hypothesis "Neither Vrenna nor myself have ever fought him." is true?
OPTIONS:
- yes
- no
- it is not possible to tell

A: "Vrenna and I both" contradicts "neither Vrenna nor myself". The answer is no.

Table A.10: Few-shot CoT prompts for NLI tasks, including ANLI and MNLI, from [189].

Q: George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? (a) dry palms. (b) wet palms. (c) palms covered with oil. (d) palms covered with lotion.

A: Dry surfaces will more likely cause more friction via rubbing than other smoother surfaces, hence dry palms will produce the most heat. The answer is (a).

Q: Which factor will most likely cause a person to develop a fever? (a) a leg muscle relaxing after exercise. (b) a bacterial population in the bloodstream. (c) several viral particles on the skin. (d) carbohydrates being digested in the stomach.

A: Option (b), bacterial population is the most likely cause for a person developing fever. The answer is (b).

Q: Which change in the state of water particles causes the particles to become arranged in a fixed position? (a) boiling. (b) melting. (c) freezing. (d) evaporating.

A: When water is freezed, the particles are arranged in a fixed position; the particles are still moving for all other options. The answer is (c).

Q: When a switch is used in an electrical circuit, the switch can (a) cause the charge to build. (b) increase and decrease the voltage. (c) cause the current to change direction. (d) stop and start the flow of current.

A: The function of a switch is to start and stop the flow of a current. The answer is (d).

Table A.11: Few-shot CoT prompts for ARC-challenge, from [190].

Q: John found that the average of 15 numbers is 40. If 10 is added to each number then the mean of the numbers is? Answer Choices: (a) 50 (b) 45 (c) 65 (d) 78 (e) 64

A: If 10 is added to each number, then the mean of the numbers also increases by 10. So the new mean would be 50. The answer is (a).

Q: If a / b = 3/4 and 8a + 5b = 22,then find the value of a. Answer Choices: (a) 1/2 (b) 3/2 (c) 5/2 (d) 4/2 (e) 7/2

A: If a / b = 3/4, then b = 4a / 3. So 8a + 5(4a / 3) = 22. This simplifies to 8a + 20a / 3 = 22, which means 44a / 3 = 22. So a is equal to 3/2. The answer is (b).

Q: A person is traveling at 20 km/hr and reached his destiny in 2.5 hr then find the distance? Answer Choices: (a) 53 km (b) 55 km (c) 52 km (d) 60 km (e) 50 km

A: The distance that the person traveled would have been 20 km/hr * 2.5 hrs = 50 km. The answer is (e).

Q: How many keystrokes are needed to type the numbers from 1 to 500? Answer Choices: (a) 1156 (b) 1392 (c) 1480 (d) 1562 (e) 1788

A: There are 9 one-digit numbers from 1 to 9. There are 90 two-digit numbers from 10 to 99. There are 401 three-digit numbers from 100 to 500. 9 + 90(2) + 401(3) = 1392. The answer is (b).

Table A.12: Few-shot CoT prompts for AQUA, from [190].

Q: Do hamsters provide food for any animals?

A: Hamsters are prey animals. Prey are food for predators. Thus, hamsters provide food for some animals. The answer is yes.

Q: Could Brooke Shields succeed at University of Pennsylvania?

A: Brooke Shields went to Princeton University. Princeton University is about as academically rigorous as the University of Pennsylvania. Thus, Brooke Shields could also succeed at the University of Pennsylvania. The answer is yes.

Q: Yes or no: Hydrogen's atomic number squared exceeds number of Spice Girls?

A: Hydrogen has an atomic number of 1. 1 squared is 1. There are 5 Spice Girls. Thus, Hydrogen's atomic number squared is less than 5. The answer is no.

Q: Yes or no: Is it common to see frost during some college commencements?

A: College commencement ceremonies can happen in December, May, and June. December is in the winter, so there can be frost. Thus, there could be frost at some commencements. The answer is yes.

Q: Yes or no: Could a llama birth twice during War in Vietnam (1945-46)?

A: The War in Vietnam was 6 months. The gestation period for a llama is 11 months, which is more than 6 months. Thus, a llama could not give birth twice during the War in Vietnam. The answer is no.

Q: Yes or no: Would a pear sink in water?

A: The density of a pear is about 0.6 g/cm^3, which is less than water. Objects less dense than water float. Thus, a pear would float. The answer is no.

Table A.13: Few-shot CoT prompts for StrategyQA, from [195].

Premise:
"No Weapons of Mass Destruction Found in Iraq Yet."
Based on this premise, can we conclude the hypothesis "Weapons of Mass Destruction Found in Iraq." is true?

A: "No Weapons of Mass Destruction Found" contradicts "Weapons of Mass Destruction Found". The answer is no.

Premise:
"A place of sorrow, after Pope John Paul II died, became a place of celebration, as Roman Catholic faithful gathered in downtown Chicago to mark the installation of new Pope Benedict XVI."
Based on this premise, can we conclude the hypothesis "Pope Benedict XVI is the new leader of the Roman Catholic Church." is true?'

A: "installation of new Pope Benedict XVI." means "Pope Benedict XVI is the new leader". The answer is yes.

Premise:
"A man is due in court later charged with the murder 26 years ago of a teenager whose case was the first to be featured on BBC One's Crimewatch. Colette Aram, 16, was walking to her boyfriend's house in Keyworth, Nottinghamshire, on 30 October 1983 when she disappeared. Her body was later found in a field close to her home. Paul Stewart Hutchinson, 50, has been charged with murder and is due before Nottingham magistrates later."
Based on this premise, can we conclude the hypothesis "Paul Stewart Hutchinson is accused of having stabbed a girl." is true?

A: The premise does not say Paul Stewart Hutchinson "stabbed" this girl. The answer is no.

Premise:
"Herceptin was already approved to treat the sickest breast cancer patients, and the company said, Monday, it will discuss with federal regulators the possibility of prescribing the drug for more breast cancer patients."
Based on this premise, can we conclude the hypothesis "Herceptin can be used to treat breast cancer." is true?

A: "Herceptin was approved to treat breast cancer" implies that "Herceptin can be used to treat breast cancer". The answer is yes.

Table A.14: Few-shot CoT prompts for RTE, from [189].

REFERENCES

[1] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC CCS*, 2016.

[2] A. Akbik, D. Blythe, and R. Vollgraf. Contextual string embeddings for sequence labeling. In *COLING*, 2018.

[3] C. Allen and T. M. Hospedales. Analogies explained: Towards understanding word embeddings. In *ICML*, 2019.

[4] M.-R. Amini, V. Feofanov, L. Pauletto, E. Devijver, and Y. Maximov. Self-training: A survey. *Arxiv*, 2202.12040.

[5] B. An, J. Lyu, Z. Wang, C. Li, C. Hu, F. Tan, R. Zhang, Y. Hu, and C. Chen. Repulsive attention: Rethinking multi-head attention as bayesian inference. In *EMNLP*, 2020.

[6] M. Atzori, S. Balloccu, and A. Bellanti. Unsupervised singleton expansion from free text. In *ICSC*, 2018.

[7] J. Ba and R. Caruana. Do deep nets really need to be deep? In *NeurIPS*, 2014.

[8] D. Balasuriya, N. Ringland, J. Nothman, T. Murphy, and J. Curran. Named entity recognition in wikipedia. In *PWNLP@IJCNLP*, 2009.

[9] T. Bansal, R. Jha, T. Munkhdalai, and A. McCallum. Self-supervised meta-learning for few-shot natural language classification tasks. In *EMNLP*, 2020.

[10] B. bench collaboration. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *ArXiv*, abs/2206.04615, 2022.

[11] D. M. Blei, T. L. Griffiths, M. I. Jordan, and J. B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *NIPS*, 2003.

[12] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[13] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. J. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[14] S. Bubeck, V. Chandrasekaran, R. Eldan, J. A. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y.-F. Li, S. M. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4. *ArXiv*, abs/2303.12712, 2023.

[15] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, U. Stefan, R. Osman, and M. Gašić. Multiwoz - a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *EMNLP*, 2018.

[16] O.-M. Camburu, T. Rocktäschel, T. Lukasiewicz, and P. Blunsom. e-snli: Natural language inference with natural language explanations. In *NeurIPS*, 2018.

[17] H.-S. Chang, Z. Wang, L. Vilnis, and A. McCallum. Distributional inclusion vector embedding for unsupervised hypernymy detection. In *NAACL-HLT*, 2017.

[18] S. Chen, J. Wang, F. Jiang, and C.-Y. Lin. Improving entity linking by modeling latent entity type information. In *AAAI*, 2020.

[19] T. Chen, Y. Chen, and B. V. Durme. Hierarchical entity typing via multi-level learning to rank. In *ACL*, 2020.

[20] X. Chen, N. Zhang, L. Li, X. Xie, S. Deng, C. Tan, F. Huang, L. Si, and H. Chen. Lightner: A lightweight generative framework with prompt-guided attention for low-resource ner. In *COLING*, 2022.

[21] X. Chen, N. Zhang, X. Xie, S. Deng, Y. Yao, C. Tan, F. Huang, L. Si, and H. Chen. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *WWW*, 2022.

[22] Y. Chen, H. Jiang, L. Liu, S. Shi, C. Fan, M. Yang, and R. Xu. An empirical study on multiple information sources for zero-shot fine-grained entity typing. In *EMNLP*, 2021.

[23] J. P. Chiu and E. Nichols. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 2016.

[24] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. B. Rao, P. Barnes, Y. Tay, N. M. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. C. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. García, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. O. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Díaz, O. Firat, M. Catasta, J. Wei, K. S. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311, 2022.

[25] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, A. Yu, A. Webson, X. Chen, G. Mishra, Z. Dai, S. S. Gu, M. Suzgun, V. Zhao, A. Chowdhery, S. Narang, Y. Huang, A. Dai, H. Yu, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei. Scaling instruction-finetuned language models. *ArXiv*, abs/2210.11416, 2022.

[26] P. Cimiano, A. Hotho, and S. Staab. Comparing conceptual, divise and agglomerative clustering for learning taxonomies from text. In *ECAI*, 2004.

[27] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

[28] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018.

[29] K. Cobbe, V. Kosaraju, M. Bavarian, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168, 2021.

[30] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, M. Primet, and J. Dureau. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *ArXiv*, abs/1805.10190, 2018.

[31] I. Dagan, O. Glickman, and B. Magnini. The pascal recognising textual entailment challenge. In *MLCW*, 2005.

[32] H. Dai, D. Du, X. Li, and Y. Song. Improving fine-grained entity typing with entity linking. In *EMNLP*, 2019.

[33] H. Dai, Y. Song, and X. Li. Exploiting semantic relations for fine-grained entity typing. In *AKBC*, 2020.

[34] H. Dai, Y. Song, and H. Wang. Ultra-fine entity typing with weak supervision from a masked language model. In *ACL/IJCNLP*, 2021.

[35] J. Deng, J. Guo, T. Liu, M. Gong, and S. Zafeiriou. Sub-center arcface: Boosting face recognition by large-scale noisy web faces. In *ECCV*, 2020.

[36] L. Derczynski, E. Nichols, M. Erp, and N. Limsopatham. Results of the wnut2017 shared task on novel and emerging entity recognition. In *NUT@EMNLP*, 2017.

[37] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[38] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[39] K. S. dhar, A. Stolfo, and M. Sachan. Distilling reasoning capabilities into smaller language models. In *ACL*, 2023.

[40] N. Ding, Y. Chen, X. Han, G. Xu, P. Xie, H. Zheng, Z. Liu, J.-Z. Li, and H.-G. Kim. Prompt-learning for fine-grained entity typing. In *EMNLP*, 2022.

[41] N. Ding, G. Xu, Y. Chen, X. Wang, X. Han, P. Xie, H. Zheng, and Z. Liu. Few-nerd: A few-shot named entity recognition dataset. In *ACL/IJCNLP*, 2021.

[42] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. P. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *KDD*, 2014.

[43] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL*, 2019.

[44] J. Dunlosky and J. Metcalfe. *Metacognition*. Sage Publications, 2008.

[45] J. Eisenstein, D. Andor, B. Bohnet, M. Collins, and D. Mimno. Honest students from untrusted teachers: Learning an interpretable question-answering pipeline from a pretrained language model. *ArXiv*, abs/2210.02498, 2022.

[46] O. Etzioni, M. J. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165:91–134, 2005.

[47] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315 5814:972–6, 2007.

[48] A. Fritzler, V. Logacheva, and M. Kretov. Few-shot classification in named entity recognition task. *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019.

[49] R. Fu, J. Guo, B. Qin, W. Che, H. Wang, and T. Liu. Learning semantic hierarchies via word embeddings. In *ACL*, 2014.

[50] J. Gao, M. Galley, and L. Li. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298, 2019.

[51] J. Gao, B. Peng, C. Li, J. Li, S. Shayandeh, L. Liden, and H.-Y. Shum. Robust conversational ai with grounded text generation. *ArXiv*, abs/2009.03457, 2020.

[52] T. Gao, A. Fisch, and D. Chen. Making pre-trained language models better few-shot learners. In *ACL*, 2021.

[53] R. Geng, B. Li, Y. Li, X.-D. Zhu, P. Jian, and J. Sun. Induction networks for few-shot text classification. In *EMNLP/IJCNLP*, 2019.

[54] M. Geva, D. Khashabi, E. Segal, T. Khot, D. Roth, and J. Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *TACL*, 2021.

[55] A. Ghaddar and P. Langlais. Transforming wikipedia into a large-scale fine-grained entity type corpus. In *LREC*, 2018.

[56] D. Gillick, N. Lazic, K. Ganchev, J. Kirchner, and D. Huynh. Context-dependent fine-grained entity type tagging. *ArXiv*, abs/1412.1820, 2014.

[57] J. Gu, Y. Wang, Y. Chen, K. Cho, and V. Li. Meta-learning for low-resource neural machine translation. In *EMNLP*, 2018.

[58] Y. Gu, X. Han, Z. Liu, and M. Huang. Ppt: Pre-trained prompt tuning for few-shot learning. In *ACL*, 2022.

[59] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *ICML*, 2017.

[60] J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *SIGIR*, 2009.

[61] 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. *JAMIA*, 21 5:902–9, 2014.

[62] S. Gupta and C. D. Manning. Improved pattern learning for bootstrapped entity extraction. In *CoNLL*, 2014.

[63] D. Z. Hakkani-Tür, G. Tür, A. Çelikyilmaz, Y.-N. Chen, J. Gao, L. Deng, and Y.-Y. Wang. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *INTERSPEECH*, 2016.

[64] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3rd edition, 2011.

[65] X. Han, W. Zhao, N. Ding, Z. Liu, and M. Sun. Ptr: Prompt tuning with rules for text classification. *ArXiv*, abs/2105.11259, 2021.

[66] X. Han, H. Zhu, P. Yu, Z. Wang, Y. Yao, Z. Liu, and M. Sun. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *EMNLP*, 2018.

[67] J. He, J. Gu, J. Shen, and M. Ranzato. Revisiting self-training for neural sequence generation. In *ICLR*, 2020.

[68] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, 1992.

[69] G. Hinton, O. Vinyals, J. Dean, et al. Distilling the knowledge in a neural network. In *NeurIPS*, 2015.

[70] N. Ho, L. Schmid, and S.-Y. Yun. Large language models are reasoning teachers. In *ACL*, 2023.

[71] M. Hofer, A. Kormilitzin, P. Goldberg, and A. J. Nevado-Holgado. Few-shot learning for named entity recognition in medical text. *ArXiv*, abs/1811.05468, 2018.

[72] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *ACL*, 2018.

[73] L. Hu, T. Yang, C. Shi, H. Ji, and X. Li. Heterogeneous graph attention networks for semi-supervised short text classification. In *EMNLP*, 2019.

[74] S. Hu, N. Ding, H. Wang, Z. Liu, J.-Z. Li, and M. Sun. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. In *ACL*, 2022.

[75] J. Huang, S. S. Gu, L. Hou, Y. Wu, X. Wang, H. Yu, and J. Han. Large language models can self-improve. In *EMNLP*, 2023.

[76] J. Huang, C. Li, K. Subudhi, D. Jose, S. Balakrishnan, W. Chen, B. Peng, J. Gao, and J. Han. Few-shot named entity recognition: An empirical baseline study. In *EMNLP*, 2021.

[77] J. Huang, Y. Meng, and J. Han. Few-shot fine-grained entity typing with automatic label interpretation and instance generation. In *KDD*, 2022.

[78] 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 *WWW*, 2020.

[79] J. Huang, Y. Xie, Y. Meng, Y. Zhang, and J. Han. Corel: Seed-guided topical taxonomy construction by concept learning and relation transferring. In *KDD*, 2020.

[80] S. Ji, T. Zhang, L. Ansari, J. Fu, P. Tiwari, and E. Cambria. Mentalbert: Publicly available pretrained language models for mental healthcare. In *LREC*, 2021.

[81] Z. Jiang, A. Anastasopoulos, J. Araki, H. Ding, and G. Neubig. X-factr: Multilingual factual knowledge retrieval from pretrained language models. In *EMNLP*, 2020.

[82] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig. How can we know what language models know? *TACL*, 8:423–438, 2020.

[83] Y. Jiao, M. Zhong, S. Li, R. Zhao, S. Ouyang, H. Ji, and J. Han. Instruct and extract: Instruction tuning for on-demand information extraction. In *EMNLP*, 2023.

[84] Z. Jie and W. Lu. Dependency-guided lstm-crf for named entity recognition. In *EMNLP/IJCNLP*, 2019.

[85] H. Jin, L. Hou, J.-Z. Li, and T. Dong. Fine-grained entity typing via hierarchical multi graph convolutional networks. In *EMNLP*, 2019.

[86] P. Jindal and D. Roth. Learning from negative examples in set-expansion. In *ICDM*, 2011.

[87] J. Jung, L. Qin, S. Welleck, F. Brahman, C. Bhagavatula, R. L. Bras, and Y. Choi. Maieutic prompting: Logically consistent reasoning with recursive explanations. In *ACL*, 2022.

[88] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher. Ctrl: A conditional transformer language model for controllable generation. *ArXiv*, abs/1909.05858, 2019.

[89] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[90] Y. Kluger, R. Basri, J. T. Chang, and M. B. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, 13 4:703–16, 2003.

[91] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. In *NeurIPS*, 2022.

[92] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby. Big transfer (bit): General visual representation learning. In *ECCV*, 2020.

[93] A. Korattikara Balan, V. Rathod, K. P. Murphy, and M. Welling. Bayesian dark knowledge. In *NeurIPS*, 2015.

[94] L. Kuhn, Y. Gal, and S. Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *ArXiv*, abs/2302.09664, 2023.

[95] A. K. Lampinen, I. Dasgupta, S. C. Y. Chan, K. Matthewson, M. H. Tessler, A. Creswell, J. L. McClelland, J. X. Wang, and F. Hill. Can language models learn from explanations in context? In *ACL Findings*, 2022.

[96] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. In *NAACL*, 2016.

[97] D.-H. Lee, M. Agarwal, A. Kadakia, J. Pujara, and X. Ren. Good examples make a faster learner: Simple demonstration-based learning for low-resource ner. In *ACL*, 2022.

[98] C. Li, X. Gao, Y. Li, X. Li, B. Peng, Y. Zhang, and J. Gao. Optimus: Organizing sentences via pre-trained modeling of a latent space. In *EMNLP*, 2020.

[99] C. Li, X. Li, L. Zhang, B. Peng, M. Zhou, and J. Gao. Self-supervised pre-training with hard examples improves visual representations. *ArXiv*, abs/2012.13493, 2020.

[100] J. Li, C. Xiong, and S. C. Hoi. Mopro: Webly supervised learning with momentum prototypes. In *ICLR*, 2021.

[101] Y. Li, Z. Lin, S. Zhang, Q. Fu, B. Chen, J.-G. Lou, and W. Chen. On the advance of making language models better reasoners. *ArXiv*, abs/2206.02336, 2022.

[102] C. Liang, Y. Yu, H. Jiang, S. Er, R. Wang, T. Zhao, and C. Zhang. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *KDD*, 2020.

[103] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. *ArXiv*, abs/2305.20050, 2023.

[104] D. Lin and X. Wu. Phrase clustering for discriminative learning. In *ACL/IJCNLP*, pages 1030–1038, 2009.

[105] S. C. Lin, J. Hilton, and O. Evans. Teaching models to express their uncertainty in words. *TMLR*, 2022.

[106] W. Lin, R. Yangarber, and R. Grishman. Bootstrapped learning of semantic classes from positive and negative examples. In *ICML*, 2003.

[107] Y. Lin and H. Ji. An attentive fine-grained entity typing model with latent type representation. In *EMNLP*, 2019.

[108] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2015.

[109] W. Ling, D. Yogatama, C. Dyer, and P. Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *ACL*, 2017.

[110] X. Ling and D. S. Weld. Fine-grained entity recognition. In *AAAI*, 2012.

[111] J. Liu, P. Pasupat, D. Cyphers, and J. R. Glass. Asgard: A portable architecture for multilingual dialogue systems. In *ICASSP*, 2013.

[112] J. Liu, P. Pasupat, Y. Wang, D. Cyphers, and J. R. Glass. Query understanding enhanced by hierarchical parsing structures. *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013.

[113] T. Liu, J. Yao, and C.-Y. Lin. Towards improving neural named entity recognition with gazetteers. In *ACL*, 2019.

[114] X. Liu, Y. Song, S. Liu, and H. Wang. Automatic taxonomy construction from keywords. In *KDD*, 2012.

[115] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. *ArXiv*, abs/2303.16634, 2023.

[116] Y. Liu, F. Meng, J. Zhang, J. Xu, Y. Chen, and J. Zhou. Gcdt: A global context enhanced deep transition architecture for sequence labeling. In *ACL*, 2019.

[117] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *ArXiv*, abs/1907.11692, 2019.

[118] M. Lukasik, S. Bhojanapalli, A. K. Menon, and S. Kumar. Does label smoothing mitigate label noise? In *ICML*, 2020.

[119] X. Ma and E. Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*, 2016.

[120] Y. Ma, E. Cambria, and S. Gao. Label embedding for zero-shot fine-grained named entity typing. In *COLING*, 2016.

[121] L. C. Magister, J. Mallinson, J. Adamek, E. Malmi, and A. Severyn. Teaching small language models to reason. In *ACL*, 2023.

[122] 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 *EMNLP*, 2018.

[123] J. Maynez, S. Narayan, B. Bohnet, and R. T. McDonald. On faithfulness and factuality in abstractive summarization. In *ACL*, 2020.

[124] K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and editing factual associations in gpt. In *NeurIPS*, 2022.

[125] Y. Meng, J. Huang, G. Wang, Z. Wang, C. Zhang, Y. Zhang, and J. Han. Discriminative topic mining via category-name guided text embedding. In *WWW*, 2020.

[126] Y. Meng, J. Huang, Y. Zhang, and J. Han. Generating training data with language models: Towards zero-shot language understanding. In *NeurIPS*, 2022.

[127] Y. Meng, Y. Zhang, J. Huang, X. Wang, Y. Zhang, H. Ji, and J. Han. Distantly-supervised named entity recognition with noise-robust learning and language model augmented self-training. In *EMNLP*, 2021.

[128] 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 *EMNLP*, 2020.

[129] Y. Meng, Y. Zhang, J. Huang, Y. Zhang, C. Zhang, and J. Han. Hierarchical topic mining via joint spherical tree and text embedding. In *KDD*, 2020.

[130] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

[131] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[132] D. M. Mimno, W. Li, and A. McCallum. Mixtures of hierarchical topics with pachinko allocation. In *ICML*, 2007.

[133] S. Min, W. Shi, M. Lewis, X. Chen, W. tau Yih, H. Hajishirzi, and L. Zettlemoyer. Nonparametric masked language modeling. In *ACL*, 2022.

[134] E. Mitchell, C. Lin, A. Bosselut, C. D. Manning, and C. Finn. Memory-based model editing at scale. *ArXiv*, abs/2206.06520, 2022.

[135] D. Mollá, M. Van Zaanen, D. Smith, et al. Named entity recognition for question answering. In *ALTW*, 2006.

[136] S. Murty, P. Verga, L. Vilnis, I. Radovanovic, and A. McCallum. Hierarchical losses and new resources for fine-grained entity typing and linking. In *ACL*, 2018.

[137] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[138] N. Nakashole, G. Weikum, and F. M. Suchanek. Patty: A taxonomy of relational patterns with semantic types. In *EMNLP-CoNLL*, 2012.

[139] S. Narang, C. Raffel, K. Lee, A. Roberts, N. Fiedel, and K. Malkan. Wt5?! training text-to-text models to explain their predictions. *ArXiv*, abs/2004.14546.

[140] R. Navigli and P. Velardi. Learning word-class lattices for definition and hypernym extraction. In *ACL*, 2010.

[141] R. Obeidat, X. Z. Fern, H. Shahbazi, and P. Tadepalli. Description-based zero-shot fine-grained entity typing. In *NAACL*, 2019.

[142] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

[143] A. Patel, S. Bhattamishra, and N. Goyal. Are nlp models really able to solve simple math word problems? In *NAACL*, 2021.

[144] B. Peng, C. Li, P. He, M. Galley, and J. Gao. Instruction tuning with gpt-4. *ArXiv*, abs/2304.03277, 2023.

[145] B. Peng, C. Li, J. Li, S. Shayandeh, L. Liden, and J. Gao. Soloist: Few-shot task-oriented dialog with a single pre-trained auto-regressive model. *TACL*, 2021.

[146] B. Peng, C. Zhu, C. Li, X. Li, J. Li, M. Zeng, and J. Gao. Few-shot natural language generation for task-oriented dialog. In *EMNLP*, 2020.

[147] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. S. Zettlemoyer. Deep contextualized word representations. In *NAACL-HLT*, 2018.

[148] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel. Language models as knowledge bases? In *EMNLP*, 2019.

[149] J. Phang, T. Févry, and S. R. Bowman. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *ArXiv*, abs/1811.01088, 2018.

[150] D. Pruthi, R. Bansal, B. Dhingra, L. B. Soares, M. Collins, Z. C. Lipton, G. Neubig, and W. W. Cohen. Evaluating explanations: How much do explanations from the teacher aid students? In *TACL 2021*.

[151] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[152] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016.

[153] W. Ralph, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, and N. X. et al. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium*, 2013.

[154] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[155] X. Ren, W. He, M. Qu, L. Huang, H. Ji, and J. Han. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *EMNLP*, 2016.

[156] X. Ren, W. He, M. Qu, C. R. Voss, H. Ji, and J. Han. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *KDD*, 2016.

[157] A. Ritter, Mausam, O. Etzioni, and S. Clark. Open domain event extraction from twitter. In *KDD*, 2012.

[158] X. Rong, Z. Chen, Q. Mei, and E. Adar. Egoset: Exploiting word ego-networks and user-generated ontology for multifaceted set expansion. In *WSDM*, 2016.

[159] A. RoyChowdhury, P. Chakrabarty, A. Singh, S. Jin, H. Jiang, L. Cao, and E. G. Learned-Miller. Automatic adaptation of object detectors to new domains using self-training. In *CVPR*, 2019.

[160] A. Salemi, S. Mysore, M. Bendersky, and H. Zamani. Lamp: When large language models meet personalization. *ArXiv*, abs/2304.11406, 2023.

[161] E. F. T. K. Sang and F. D. Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *HLT-NAACL*, 2003.

[162] V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, T. L. Scao, A. Raja, et al. Multitask prompted training enables zero-shot task generalization. In *ICLR*, 2022.

[163] K. Shaalan. A survey of arabic named entity recognition and classification. *Computational Linguistics*, 2014.

[164] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han. Automated phrase mining from massive text corpora. *TKDE*, 30:1825–1837, 2017.

[165] 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 *ECML/PKDD*, 2017.

[166] J. Shen, Z. Wu, D. Lei, C. Zhang, X. Ren, M. T. Vanni, B. M. Sadler, and J. Han. Hiexpan: Task-guided taxonomy construction by hierarchical tree expansion. In *KDD*, 2018.

[167] W. Shi, X. Han, M. Lewis, Y. Tsvetkov, L. Zettlemoyer, and S. Yih. Trusting your evidence: Hallucinate less with context-aware decoding. *ArXiv*, abs/2305.14739, 2023.

[168] S. Shimaoka, P. Stenetorp, K. Inui, and S. Riedel. Neural architectures for fine-grained entity type classification. In *EACL*, 2017.

[169] T. Shin, Y. Razeghi, R. L. L. IV, E. Wallace, and S. Singh. Eliciting knowledge from language models using automatically generated prompts. In *EMNLP*, 2020.

[170] V. Shwartz, Y. Goldberg, and I. Dagan. Improving hypernymy detection with an integrated path-based and distributional method. In *ACL*, 2016.

[171] C. Snell, D. Klein, and R. Zhong. Learning by distilling context. *ArXiv*, abs/2209.15189, 2022.

[172] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.

[173] R. Snow, D. Jurafsky, and A. Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*, 2004.

[174] L. B. Soares, N. FitzGerald, J. Ling, and T. Kwiatkowski. Matching the blanks: Distributional similarity for relation learning. In *ACL*, 2019.

[175] A. Stubbs and Ö. Uzuner. Annotating longitudinal clinical narratives for de-identification: The 2014 i2b2/uthealth corpus. *Journal of biomedical informatics*, 58 Suppl:S20–9, 2015.

[176] S.-Y. Su, K.-L. Lo, Y.-T. Yeh, and Y.-N. V. Chen. Natural language generation by hierarchical decoding with linguistic patterns. In *NAACL*, 2018.

[177] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018.

[178] Y. Tay, M. Dehghani, V. Q. Tran, X. García, J. Wei, X. Wang, H. W. Chung, D. Bahri, T. Schuster, H. Zheng, D. Zhou, N. Houlsby, and D. Metzler. Ul2: Unifying language learning paradigms. In *ICLR*, 2023.

[179] M. Thelen and E. Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *EMNLP*, 2002.

[180] K. Tian, E. Mitchell, A. Zhou, A. Sharma, R. Rafailov, H. Yao, C. Finn, and C. D. Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *ArXiv*, abs/2305.14975, 2023.

[181] Y. Tian, Y. Wang, D. Krishnan, J. Tenenbaum, and P. Isola. Rethinking few-shot image classification: a good embedding is all you need? *ArXiv*, abs/2003.11539, 2020.

[182] M. Trajanoska, R. Stojanov, and D. Trajanov. Enhancing knowledge graph construction using large language models. *ArXiv*, abs/2305.04676, 2023.

[183] J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *ACL*, 2010.

[184] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *NIPS*, 2016.

[185] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*, 2019.

[186] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*, 2018.

[187] 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 *KDD '13*, 2013.

[188] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, and L. Carin. Joint embedding of words and labels for text classification. In *ACL*, 2018.

[189] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, and D. Zhou. Rationale-augmented ensembles in language models. *ArXiv*, abs/2207.00747, 2022.

[190] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023.

[191] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.

[192] J. Weeds, D. J. Weir, and D. McCarthy. Characterising measures of lexical distributional similarity. In *COLING*, 2004.

[193] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned language models are zero-shot learners. In *ICLR*, 2022.

[194] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, et al. Emergent abilities of large language models. *TMLR*, 2022.

[195] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

[196] R. Weischedel and A. Brunstein. Bbn pronoun and coreference and entity type corpus. *Linguistic Data Consortium*, 2005.

[197] A. Williams, N. Nangia, and S. R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*, 2018.

[198] S. Wiseman and K. Stratos. Label-agnostic sequence labeling by copying nearest neighbors. In *ACL*, 2019.

[199] L. S. Wong, G. Grand, A. K. Lew, N. D. Goodman, V. K. Mansinghka, J. Andreas, and J. B. Tenenbaum. From word models to world models: Translating from natural language to the probabilistic language of thought. *ArXiv*, abs/2306.12672, 2023.

[200] Z. Xi, S. Jin, Y. Zhou, R. Zheng, S. Gao, T. Gui, Q. Zhang, and X. Huang. Self-polish: Enhance reasoning in large language models via problem refinement. *ArXiv*, abs/2305.14497, 2023.

[201] Q. Xie, E. Hovy, M.-T. Luong, and Q. V. Le. Self-training with noisy student improves imagenet classification. In *CVPR*, 2020.

[202] J. Xin, Y. Lin, Z. Liu, and M. Sun. Improving neural fine-grained entity typing with knowledge attention. In *AAAI*, 2018.

[203] Y. Yaghoobzadeh, H. Adel, and H. Schütze. Noise mitigation for neural entity typing and relation extraction. In *EACL*, 2017.

[204] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto. Luke: Deep contextualized entity representations with entity-aware self-attention. In *EMNLP*, 2020.

[205] G. H. Yang and J. P. Callan. A metric-based framework for automatic taxonomy induction. In *ACL/IJCNLP*, 2009.

[206] K. Yang, W. Lei, D. Liu, W. Qi, and J. Lv. Pos-constrained parallel decoding for non-autoregressive generation. In *ACL*, 2021.

[207] S. Yang, L. Zou, Z. Wang, J. Yan, and J.-R. Wen. Efficiently answering technical questions - a knowledge graph approach. In *AAAI*, 2017.

[208] Y. Yang and A. Katiyar. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *EMNLP*, 2020.

[209] X. Ye and G. Durrett. The unreliability of explanations in few-shot in-context learning. In *NeurIPS*, 2022.

[210] K. M. Yoo, D. Park, J. Kang, S.-W. Lee, and W. Park. Gpt3mix: Leveraging large-scale language models for text augmentation. In *EMNLP Findings*, 2021.

[211] M. A. Yosef, S. Bauer, J. Hoffart, M. Spaniol, and G. Weikum. Hyena: Hierarchical type classification for entity names. In *COLING*, 2012.

[212] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NeurIPS*, 2014.

[213] M. Yu, X. Guo, J. Yi, S. Chang, S. Potdar, Y. Cheng, G. Tesauro, H. Wang, and B. Zhou. Diverse few-shot text classification with multiple metrics. In *NAACL-HLT*, 2018.

[214] P. Yu, Z. Huang, R. Rahimi, and J. Allan. Corpus-based set expansion with lexical features and distributed representations. In *SIGIR*, 2019.

[215] Z. Yuan and D. Downey. Otyper: A neural architecture for open named entity typing. In *AAAI*, 2018.

[216] M. Yuksekgonul, V. Chandrasekaran, E. Jones, S. Gunasekar, R. Naik, H. Palangi, E. Kamar, and B. Nushi. Attention satisfies: A constraint-satisfaction lens on factual errors of language models. *ArXiv*, abs/2309.15098, 2023.

[217] O. Zaidan, J. Eisner, and C. Piatko. Using "annotator rationales" to improve machine learning for text categorization. In *NAACL*, 2007.

[218] E. Zelikman, Y. Wu, J. Mu, and N. D. Goodman. Star: Bootstrapping reasoning with reasoning. In *NeurIPS*, 2022.

[219] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi. Defending against neural fake news. In *NeurIPS*, 2019.

[220] C. Zhang, F. Tao, X. Chen, J. Shen, M. Jiang, B. M. Sadler, M. Vanni, and J. Han. Taxogen: Unsupervised topic taxonomy construction by adaptive term embedding and clustering. In *KDD*, 2018.

[221] H. Zhang, D. Long, G. Xu, M. Zhu, P. Xie, F. Huang, and J. Wang. Learning with noise: Improving distantly-supervised fine-grained entity typing via automatic relabeling. In *IJCAI*, 2020.

[222] T. Zhang, C. Xia, C.-T. Lu, and P. S. Yu. Mzet: Memory augmented zero-shot fine-grained named entity typing. In *COLING*, 2020.

[223] Y. Zhang, A. Ahmed, V. Josifovski, and A. J. Smola. Taxonomy discovery for personalized recommendation. In *WSDM*, 2014.

[224] T. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh. Calibrate before use: Improving few-shot performance of language models. In *ICML*, 2021.

[225] Y. Zhao, C. Li, P. Yu, and C. Chen. Remp: Rectified metric propagation for few-shot learning. *CVPR Workshops*, 2020.

[226] C. Zheng, Z. Liu, E. Xie, Z. Li, and Y. Li. Progressive-hint prompting improves reasoning in large language models. *ArXiv*, abs/2304.09797, 2023.

[227] M. Zhitomirsky-Geffet and I. Dagan. The distributional inclusion hypotheses and lexical entailment. In *ACL*, 2005.

[228] B. Zhou, D. Khashabi, C.-T. Tsai, and D. Roth. Zero-shot open entity typing as type-compatible grounding. In *EMNLP*, 2018.

[229] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu, S. Zhang, G. Ghosh, M. Lewis, L. Zettlemoyer, and O. Levy. Lima: Less is more for alignment. *ArXiv*, abs/2305.11206, 2023.

[230] D. Zhou, N. Scharli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, O. Bousquet, Q. Le, and E. Chi. Least-to-most prompting enables complex reasoning in large language models. In *ICLR*, 2023.

[231] K. Zhou, D. Jurafsky, and T. Hashimoto. Navigating the grey area: Expressions of overconfidence and uncertainty in language models. *ArXiv*, abs/2302.13439, 2023.

[232] M. Ziyadi, Y. Sun, A. Goswami, J. Huang, and W. Chen. Example-based named entity recognition. *ArXiv*, abs/2008.10570, 2020.