CodeTaxo: Enhancing Taxonomy Expansion with Limited Examples via Code Language Prompts

Qingkai Zeng*
qzeng@nd.edu
University of Notre Dame
Notre Dame, IN, USA

Yuyang Bai†
ybai3@nd.edu
University of Notre Dame
Notre Dame, IN, USA

Zhaoxuan Tan
ztan3@nd.edu
University of Notre Dame
Notre Dame, IN, USA

Zhenyu Wu
zwu23@nd.edu
University of Notre Dame
Notre Dame, IN, USA

Shangbin Feng
shangbin@cs.washington.edu
University of Washington
Seattle, WA, USA

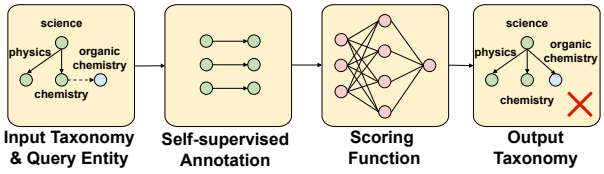
Meng Jiang
mjiang2@nd.edu
University of Notre Dame
Notre Dame, IN, USA

Abstract

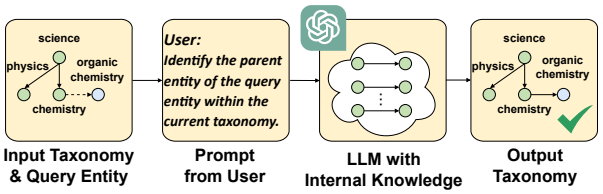
Taxonomies play a crucial role in various applications by providing a structural representation of knowledge. The task of taxonomy expansion involves integrating emerging concepts into existing taxonomies by identifying appropriate parent concepts for these new query concepts. Previous approaches typically relied on self-supervised methods that generate annotation data from existing taxonomies. However, these methods are less effective when the existing taxonomy is small (fewer than 100 entities). In this work, we introduce CodeTaxo, a novel approach that leverages large language models through code language prompts to capture the taxonomic structure. Extensive experiments on five real-world benchmarks from different domains demonstrate that CodeTaxo consistently achieves superior performance across all evaluation metrics, significantly outperforming previous state-of-the-art methods. The code and data are available at https://github.com/QingkaiZeng/CodeTaxo-Pub.

Keywords

Taxonomy Expansion; Large Language Models; In-context Learning



(a) Discriminative Methods: A scoring function is trained to evaluate and select the best parent entity from the current taxonomy for the query entity based on the scoring results [17, 21, 28].



(b) Generative Methods: Prompting LLMs to generate the parent entity from the current taxonomy based on the query entity.

Figure 1: Two Types of Methods for Taxonomy Expansion

1 Introduction

Taxonomy has a hierarchical structure that represents the hypernym-hyponym relations or "is-A" relations between concepts or entities. Researchers have been using relational knowledge to identify semantic relevance for web search [15, 34], personalized recommendation [26, 36], and question answering [32]. Existing taxonomies are constructed mainly by human experts or through crowd-sourcing, but this manual approach is time-consuming, labor-intensive, and often limited in knowledge coverage [5, 12]. As new entities continue to emerge, it becomes essential to keep these taxonomies up-to-date by enriching them with these emerging entities. To address the challenges of manual taxonomy construction and maintenance, taxonomy expansion tasks have been developed to automatically expand the existing taxonomies by integrating emerging entities into the relational graph [12].

Recent works primarily contribute to taxonomy expansion following the discriminative method illustrated in Figure 1a. These methods extract hierarchical information from the existing taxonomy and employ various techniques to model the structural information, such as local Egonet [21], mini-paths [35] and Ego-Tree [28]. As pre-trained language models (PLMs) that have enhanced the performance of various text processing systems, PLMs are leveraged to encode entities' textual descriptions (term and definition) to enrich the semantic representation of entities and improve the performance in taxonomy expansion tasks [17, 28, 29, 31].

The primary methodology adopted by many of the aforementioned methods involves learning hierarchical structural information from annotations generated in a self-supervised manner based on existing taxonomies. However, these methods often encounter limitations when the existing taxonomies are small. In such scenarios, the limited size of the taxonomy fails to generate sufficient self-supervised annotation data, which is crucial for training models to effectively learn the hierarchical structure necessary for taxonomy expansion.

Generative Large Language Models (LLMs) such as GPT-4 [1] and Llama family [8, 27] have recently demonstrated remarkable capabilities in text comprehension and generation, making them highly effective for tasks aimed at generating structural knowledge [4, 23, 24, 33]. Moreover, increasing the number of parameters

*Corresponding Author
†Work done during the visiting student program at the University of Notre Dame.

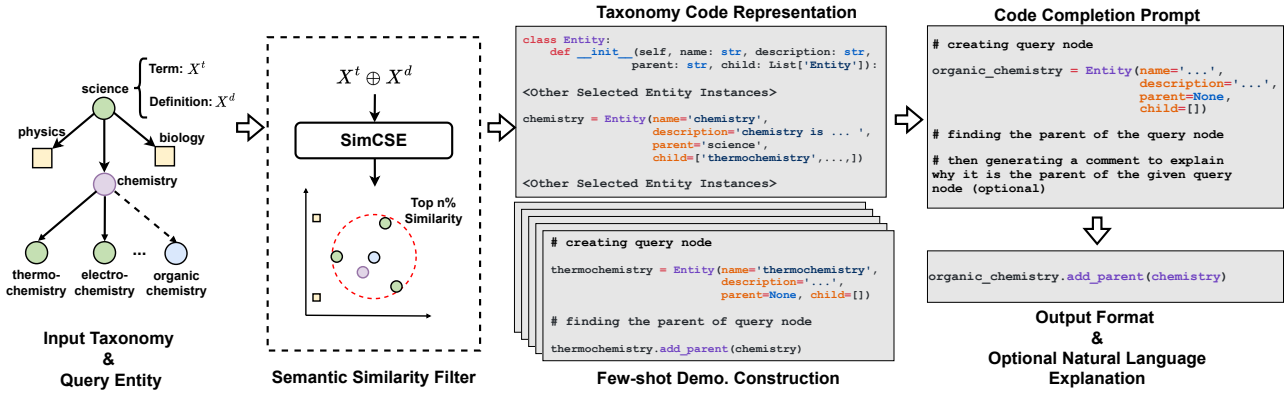


Figure 2: The overview of the framework for CodeTaxo: CodeTaxo reformulates the task of integrating a query entity q into an existing taxonomy $\mathcal{T}_0$ as a code completion task using code-based prompts for LLMs.

in LLMs significantly enhances their generalization capabilities, surpassing smaller pre-trained models, and enabling superior performance in few-shot or zero-shot settings. Consequently, even in scenarios where existing taxonomies cannot provide sufficient self-supervised annotation data for training, LLMs can still effectively perform the task by leveraging the extensive knowledge embedded within their parameters, acquired from large-scale pre-training corpora. In Figure 1b, we illustrate the pipeline to how generative methods are applied to the taxonomy expansion task.

However, two key challenges arise when employing LLMs for taxonomy expansion tasks. First, unlike traditional NLP tasks such as question answering and machine translation, which naturally lend themselves to text-to-text generation, the taxonomic structure required for this task is often difficult to represent using natural language. In particular, the taxonomic structure to be generated must be serialized into text, which involves "flattening" the taxonomy into a sequence of parent-child entity pairs [18]. However, LLMs are primarily pre-trained on unstructured text, making these serialized structured outputs considerably different from the majority of their pre-training data. Moreover, in natural language, semantically related words are usually located within a relatively small range. In contrast, when representing a taxonomy as a linear sequence, entities that are conceptually related may be positioned at greater distances from each other. Second, for large-scale taxonomies, including every entity from the existing taxonomy within the prompt is impractical due to the limited contextual window size of current LLMs and the associated computational costs. Even if we could incorporate all these thousands of entities directly in the prompts, it would result in structural information loss, thereby reducing the clarity with which LLMs can distinguish entity-specific details.

To address these two challenges, in this work, we proposed CodeTaxo, a novel approach to taxonomy expansion that leverages code languages as prompts. Recent progress in large language models of code (Code-LLMs) [20] has demonstrated the potential to employ Code-LLMs for different structure prediction tasks [4, 13, 14, 18, 30]. The core philosophy of these approaches is that code language can more effectively represent structural data compared to natural language. In CodeTaxo, we reformulate the taxonomy expansion task as a code completion problem to exploit the benefits of code languages. We first define a base class `Entity`, which encapsulates the surface name and definition of entities to represent their semantic information. Additionally, `Entity` class includes references to the parent entity and a list of child entities, thereby capturing the taxonomic relations among the entities. Two methods for modifying the taxonomic relations between entities are also included in the `Entity` class. To represent the existing taxonomy that we aim to expand, we instantiate the `Entity` class for each entity within the current taxonomy. To address the limitation posed by the contextual window size of LLMs, which cannot accommodate the entire taxonomy, we selectively include only entities that exhibit high similarity to the query entity in the prompt. For encoding the textual descriptions of entities, we utilize SimCSE [10], measuring similarity through cosine similarity.

The effectiveness of CodeTaxo has been validated through extensive experiments on two sets of small-scale sub-taxonomies from WordNet [3], Graphine [16], and three large-scale taxonomies from SemEval-2016 [5]. The experimental result demonstrates that CodeTaxo in one-shot setting outperforms all self-supervised baseline methods, despite these baselines being well-trained on self-supervised annotation data derived from large-scale taxonomies in SemEval-2016. Specifically, our CodeTaxo in the one-shot setting shows relative improvements in accuracy of 10.26%, 8.89%, and 9.21% on SemEval-Sci, SemEval-Env, and SemEval-Food, respectively. Additionally, we evaluated CodeTaxo using various open-source LLMs, revealing several interesting observations discussed in this work.

In summary, this study makes the following contributions:

- We introduce CodeTaxo, an innovative in-context learning method that utilizes code language prompts to represent taxonomic relationships between entities, thereby improving the effectiveness of taxonomy expansion.
- We develop a similarity-based filter, which employs a small pre-trained model to encode the textual descriptions of entities, ensuring that only highly relevant entities are included in the prompt in relation to the query entity.
- Extensive experiments demonstrate that CodeTaxo significantly enhances the performance of taxonomy expansion across two sets of small-scale sub-taxonomies and three large-scale taxonomies.

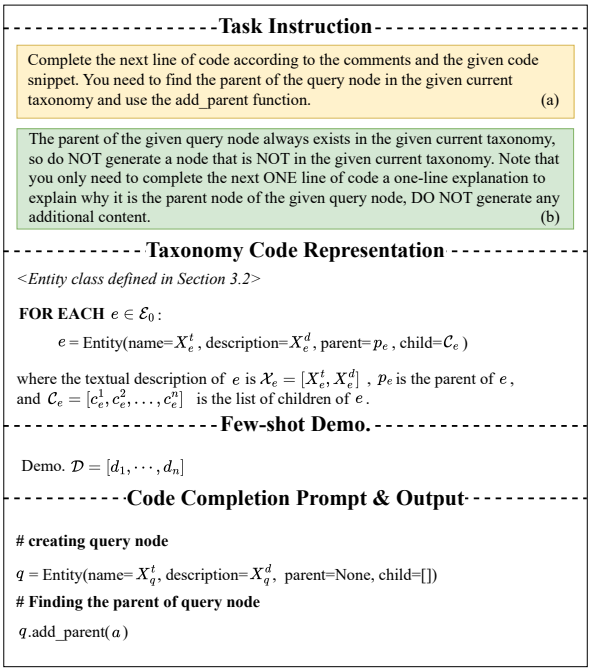


Figure 3: Prompt Overview of CodeTaxo

Scope and Limitation. This study represents an initial effort to utilize LLMs for taxonomy expansion. Our primary objective is to identify an effective in-context learning strategy to leverage the potential of LLMs. We acknowledge that the performance and scalability of CodeTaxo are constrained by the inherent knowledge of LLMs and the limitations of their context window size. While this paper does not address the challenges of expanding LLM knowledge or increasing context window size, we hope that our work will inspire further research in these areas.

2 Problem Definition

In this section, we present the key concepts used in this paper and formally define the taxonomy expansion problem:

Definition 2.1 (Taxonomy). We follow the definition of taxonomy in [11]. A taxonomy $\mathcal{T} = (\mathcal{E}, \mathcal{H})$ is a tree-like structure, where each entity $e \in \mathcal{E}$ is a conceptual entity, and each edge $h \in \mathcal{H}$ represents the hypernymy-hyponymy relation between the two entities connected by it. Each entity e is associated with a set of textual description $\mathcal{X}_e = \{X_e^t, X_e^d\}$, where X_e^t is its term and X_e^d is its definition. Meanwhile, each directed edge $h = \langle p, c \rangle \in \mathcal{H}$ represents a parent-child relationship that points to a child entity c from its most exact hypernymy entity p.

Since the taxonomies are naturally incomplete and always changing as emerging entities are introduced. Adding these emerging entities to the existing taxonomy is essential to keep it up-to-date. This study focuses on how to expand taxonomies with given emerging entities. We define the task of taxonomy expansion as follows:

Definition 2.2 (Taxonomy Expansion). Given a set of emerging conceptual entities $\mathcal{E}'$, taxonomy expansion aims to incorporate these entities into an existing *seed taxonomy* $\mathcal{T}_0 = (\mathcal{E}_0, \mathcal{H}_0)$. The goal is to expand $\mathcal{T}_0$ to be a larger taxonomy $\mathcal{T} = (\mathcal{E}_0 \cup \mathcal{E}', \mathcal{H}')$. To insert each query entity $q \in \mathcal{E}'$, we identify an appropriate anchor entity $a \in \mathcal{E}_0$, and introduce a new edge $\langle q, a \rangle$. Consequently, the updated edge set is $\mathcal{H}' = \mathcal{H}_0 \cup_{q \in \mathcal{E}'} \{\langle q, a \rangle\}$.

```python
from typing import List

class Entity:
    def __init__(
                self,
                name: str,
                description: str,
                parent: 'Entity',
                child: List['Entity']
    ):
        self.name = name
        self.description = description
        self.parent = parent
        self.child = child

    def add_parent(self, parent: 'Entity'):
        self.parent = parent
        parent.add_child(self)

    def add_child(self, child: 'Entity'):
        self.child.append(child)
```

Figure 4: Python code in CodeTaxo defining a `Entity` class with two methods for managing parent-child relations.

3 Methodology

In this section, we provide a comprehensive overview of our proposed CodeTaxo designed for addressing the taxonomy expansion task. Specifically, CodeTaxo expands the existing taxonomy by prompting LLMs with code language. The pipeline of CodeTaxo is shown in Figure 2. Our CodeTaxo consists of three parts: Task Instruction, Taxonomy Code Representation, and Few-shot Demonstrations Construction.

3.1 Task Instruction

To enhance the effectiveness and accuracy of LLMs in completing the taxonomy expansion task, we propose a detailed task description along with a set of fundamental rules, denoted as $\mathcal{R}$, for expanding the existing taxonomy via the query entity. As illustrated in Figure 3, component (a) outlines the objectives of the taxonomy expansion task, framing it as a code completion task and specifying `add_parent` function should be employed. In component (b), we emphasize a set of fundamental rules $\mathcal{R}$ for the taxonomy expansion task. These rules include the following: 1. Do not use entities that are not covered in the existing taxonomy $\mathcal{T}_0 = (\mathcal{E}_0, \mathcal{H}_0)$ ($\mathbf{r}_1$); 2. Maintain the output generation format by LLMs, consisting of one line of code followed by one line explaining why the model made that prediction ($\mathbf{r}_2$); 3. Refrain from generating additional content ($\mathbf{r}_3$). Additionally, the rule for generating an explanation for the prediction in $\mathbf{r}_2$ is optional for future analysis. In CodeTaxo, this rule is omitted as generating explanations is not required.

3.2 Taxonomy Code Representation

To represent the existing taxonomy $\mathcal{T}_0 = (\mathcal{V}_0, \mathcal{E}_0)$ as code language, we concatenate the entity class definition, representation of existing taxonomic relations, and the code completion prompt. In this work,

we use Python as the programming language for the code prompt due to its widespread popularity.

3.2.1 Entity Class Definition. First, we define a base type `Entity` to be inherited by each entity mentioned in the taxonomy expansion task. As shown in Figure 4, the code defines a Python class named `Entity` that models a taxonomic structure with parent-child relations. The first line imports the `List` type from the `typing` module, which is used for type hinting. This allows the `child` attribute to be explicitly declared as a list of `Entity` objects.

The `Entity` class encapsulates the attributes and methods for managing hierarchical entities. The `__init__` method initializes an instance of the `Entity` class with the following parameters:

- `name`: A string representing the term of the entity.
- `description`: A string representing the textual description of the entity
- `parent`: An instance of the `Entity` class, denoting the parent entity within the taxonomy.
- `child`: A list of entities, each an instance of the `Entity` class, representing the entity's children.

These instance attributes are assigned as follows: `self.name`, `self.description`, `self.parent`, and `self.child`. Additionally, since we consider that each entity in the taxonomy should only have one parent entity, we do not use the `List` type for the `parent` attribute, unlike the `child` attribute.

The `Entity` class includes two methods for modifying the parent-child relations between entities. The first method, `add_parent`, assigns a parent entity to the current entity. It takes one parameter, `parent`, which is an instance of the `Entity` class. The second method, `add_child`, appends the child entity to the `self.child` list of the current entity. This method also requires one parameter, `child`, which is an instance of the `Entity` class.

3.2.2 Representing the Existing Taxonomy. To facilitate the taxonomy expansion, the initial taxonomy $\mathcal{T}_0$ is encoded using a programming language. Instances of the `Entity` class, as defined in Section 3.2.1, are created for each entity e in the set $\mathcal{E}_0$ of $\mathcal{T}_0$. The taxonomy $\mathcal{T}_0$ is traversed from top to bottom, and for each entry, an entity $e \in \mathcal{E}_0$ is instantiated as follows:

$$e = \text{Entity}(\text{name} = X_e^t, \text{description} = X_e^d, \\ \text{parent} = p_e, \text{child} = C_e)$$

where p_e denotes the parent entity of e, and $C_e = [c_e^1, c_e^2, \ldots, c_e^n]$ represents the list of its child entities.

3.2.3 Semantic Similarity Filter. Including every entity $e \in \mathcal{E}_0$ in the code prompt to represent the existing taxonomy $\mathcal{T}_0$ presents two major challenges. First, the limited contextual window size of LLMs makes it impractical for large-scale taxonomies. Second, it unnecessarily expands the search space, introducing irrelevant entities and redundant information. To mitigate these issues, we propose a Semantic Similarity Filter that selects only entities relevant to the query q for inclusion in the prompt context.

To compute the similarity between a query entity q with its descriptive text $\mathcal{X}q = \{X_q^t, X_q^d\}$ and an entity $e_i \in \mathcal{E}0$ with its descriptive text $\mathcal{X}e_i = \{X_{e_i}^t, X_{e_i}^d\}$, we employ the pre-trained language model (PLM) as textual encoder. We concatenate the query entity q and the i-th entity e_i with special tokens [CLS] and [SEP], then encode the sequence using a pre-trained SimCSE model [10]. SimCSE converts them into m-dimensional representation $\mathbf{q}, \mathbf{e}_i \in \mathbb{R}^m$:

$$\mathbf{q} = \text{PLM}([\text{CLS}] \oplus X_q^t \oplus X_q^d \oplus [\text{SEP}])$$

$$\mathbf{e}_i = \text{PLM}([\text{CLS}] \oplus X_{e_i}^t \oplus X_{e_i}^d \oplus [\text{SEP}])$$

The semantic relevance is calculated using cosine similarity between $\{\mathbf{e}_i\}_{i=1}^n$ and $\mathbf{q}$. We select the Top-k entities with high similarity with query entity q from the entity set $\mathcal{E}_0$ from existing taxonomy $\mathcal{T}_0$ as follow:

$$\mathcal{I} = \underset{\substack{\mathcal{I} \subseteq \{1,2,\ldots,n\}, \\ |\mathcal{I}|=k}}{\text{argmax}} \sum_{i \in \mathcal{I}} \text{cos\_sim}(\mathbf{e}_i, \mathbf{q})$$

where $\mathcal{I}$ is the index set of the selected entities $\mathcal{E}_{sel} = \{e_i | i \in \mathcal{I}\}$ that represents the existing taxonomy. In this work, k is set to 50% of the entities in the taxonomy.

3.2.4 Code Completion Prompt. The code completion prompt involves the instantiation of a query entity q as an instance of the `Entity` class, as defined in Figure 3.2.1. Since the query entity q lacks information about its parent and child entities, it is instantiated as follows:

$$q = \text{Entity}(\text{name} = X_q^t, \text{description} = X_q^d, \\ \text{parent} = \text{None}, \text{child} = [])$$

Here, the name and description are initialized with the query's terms X_q^t and definition X_q^d, respectively, while the parent is set to `None`, and the child list is empty.

We include the requirement "*Find the parent of the query node*" as a comment to guide LLMs in selecting an anchor entity $a \in \mathcal{E}_{sel}$ as the parent entity for entity q. The output is the query q, an instance of the `Entity` class, which invokes the predefined method `add_parent()` to assign a as its parent entity like `q.add_parent(a)`.

We propose incorporating an optional feature in the code completion prompt: "*then generating a comment to explain why it is the parent of the given query node*". This feature allows the LLM to simultaneously generate both the prediction and its rationale, enhancing the explainability of the output and revealing interesting insights, as discussed in Section 4.6.

3.3 Few-shot Demonstration Construction

To enhance LLMs' ability to expand our existing taxonomy, we propose a method for constructing demonstrations using the initial taxonomy $\mathcal{T}_0$. Our demonstration selection strategy focuses on the semantic similarity between the query entity q and entities $e \in \mathcal{E}_0$ in the existing taxonomy. Specifically, we use SimCSE encoding to calculate these similarities, selecting the top-5 entities from the existing set $\mathcal{E}_0$ based on their similarity to q:

$$\mathcal{I}_d = \underset{\substack{\mathcal{I}_d \subseteq \{1,2,\ldots,n\}, \\ |\mathcal{I}_d|=5}}{\text{argmax}} \sum_{i \in \mathcal{I}_d} \text{cos\_sim}(\mathbf{e}_i, \mathbf{q})$$

Here, $\mathcal{I}_d$ represents the indices of entities selected for the demonstration set $\mathcal{E}_{demo} = \{e_i | i \in \mathcal{I}_d\}$. For each demonstration d_i, we treat each entity $e_i \in \mathcal{E}_{demo}$ as a query entity and, following the procedure outlined in Section 3.2.4, add its parent entity using the `add_parent` method.

	#Concepts	#Edges	Depth
WordNet	20.5	19.5	3.0
Graphine	48.2	48.2	4.6
SemEval-Sci	429.0	451.0	8.0
SemEval-Env	261.0	261.0	6.0
SemEval-Food	1,486.0	1,576.0	8.0

Table 1: Statistics of five taxonomy benchmarks. For WordNet and Graphine, we report the average for taxonomies included in these two benchmarks.

4 Experiments

Our proposed CodeTaxo is evaluated on five real-world benchmarks to address these research questions (RQs):

- **RQ1:** How does the performance of the CodeTaxo compare to state-of-the-art baselines in taxonomy expansion task?
- **RQ2:** How does CodeTaxo perform across different large language models?
- **RQ3:** Which components of CodeTaxo most significantly influence the effectiveness of taxonomy expansion, and how can their hyperparameters be optimized?

4.1 Experimental Settings

4.1.1 Datasets. We evaluate the performance of taxonomy expansion methods on small-scale taxonomies using WordNet Sub-taxonomies from [3], and Graphine taxonomies from [16]. Specifically, we use 35 Graphine taxonomies with fewer than 100 entities, selected from a total of 227 taxonomies. For the Graphine dataset, we selected 35 taxonomies with fewer than 100 entities out of 227 total taxonomies. In our experiment with WordNet, we utilized 114 sub-taxonomies from the test sets. Additionally, we evaluate three large-scale taxonomies from SemEval-2016 [5] across science, environment, and food domains. Table 1 presents the statistics of these taxonomies, all of which contain entities and definitions curated by human experts. For all benchmarks, 20% of leaf entities are reserved for testing, with the remaining entities used for training.

4.1.2 Baseline Methods. In the taxonomy expansion task, we evaluated the performance of CodeTaxo against several baseline methods, using both GPT-4o and GPT-4o-mini for in-context learning. The baselines included:

- **TaxoExpan [21]:** adopts GNNs to encode local ego-graphs in taxonomy to enhance entity representation.
- **STEAM [35]:** utilizes the mini-path information to capture the global structure of the taxonomy.
- **HEF [29]:** represents taxonomies as ego-trees to capture hierarchy, fully leveraging the hierarchical structure to improve taxonomy coherence.
- **Musubu [25]:** leverages pre-trained models and fine-tunes them as sentence classifiers using queries generated from Hearst patterns.
- **TEMP [17]:** utilizes a pre-trained model to encode text descriptions of each concept in the taxonomy. It incorporates taxonomic structure information through taxonomy paths.
- **BoxTaxo [11]:** represent the entities via box embeddings instead of single vector embeddings to capture the hierarchical relation between entities.

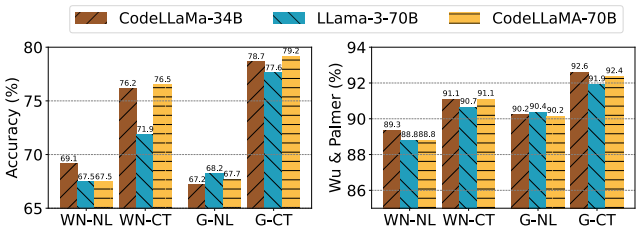


Figure 5: Performance comparison of NL and CodeTaxo (CT) across Llama trained on Code and Natural Language domains. Due to limited contextual window sizes, evaluations were conducted on small-scale sub-taxonomies from WordNet (WN) and Graphine (G).

- **TaxoPrompt [31]:** adopt prompt tuning on the BERT-based encoder model to capture the taxonomic structure.
- **TaxoInstruct [22]:** a unified framework for taxonomy-related tasks using instruction tuning, focused solely on taxonomy expansion for fair comparison.

4.1.3 Evaluation Metrics. The performance of CodeTaxo and the baseline models for taxonomy expansion tasks is evaluated using commonly adopted metrics, including accuracy (Acc) and Wu & Palmer similarity (Wu&P), as established in prior work [17, 28, 35].

4.2 Results on Taxonomy Expansion (RQ1)

We compare CodeTaxo with baseline methods for taxonomy expansion, as shown in Table 2. The baselines include two categories: self-supervised methods and in-context learning methods. To our knowledge, CodeTaxo is the first approach to utilize prompting LLMs for taxonomy expansion. We also design a natural language prompt (NL) for comparison to assess the effectiveness of CodeTaxo. The NL prompt mirrors CodeTaxo but omits the entity class definition and uses a sentence for each entity, incorporating the entity's term, definition, and its parent and child relationships. We have three major observations as follows:

First, both NL and CodeTaxo significantly outperform all self-supervised baselines on the WordNet and Graphine taxonomies. Notably, CodeTaxo in a one-shot setting shows a 72.06% and 103.06% improvement in accuracy over the best self-supervised baselines (TaxoInstruct and TEMP) on WordNet and Graphine, respectively. This indicates that LLM prompting via code prompts can effectively harness the internal knowledge of LLMs, even with limited annotated data. In contrast, self-supervised methods, which heavily depend on annotation data from existing taxonomies, struggle to learn taxonomic relations when the existing taxonomy is insufficiently large for expansion tasks.

Second, for the large-scale taxonomies in SemEval-2016, CodeTaxo in a one-shot setting still outperforms self-supervised methods, even when the existing taxonomy is sufficiently large for generating annotation data. However, the NL prompt does not exhibit the same performance advantages, particularly in terms of accuracy, when compared with self-supervised methods and CodeTaxo. Specifically, CodeTaxo outperforms the best self-supervised baseline, TaxoPrompt, by 10.26%, 8.89%, and 9.21% in accuracy on SemEval-Sci, SemEval-Env, and SemEval-Food, respectively. Conversely, the NL prompt underperforms TaxoPrompt on SemEval-Sci and SemEval-Env, with a decrease in accuracy of 10.75% and 8.53%,

Dataset	SemEval-Sci		SemEval-Env		SemEval-Food		WordNet		Graphine	
Metric	**Acc**	**Wu&P**	**Acc**	**Wu&P**	**Acc**	**Wu&P**	**Acc**	**Wu&P**	**Acc**	**Wu&P**
Self-supervised Setting										
TaxoExpan [21]	27.8	57.6	11.1	54.8	27.6	54.2	19.8	64.8	24.5	65.9
STEAM [35]	36.5	68.2	36.1	69.6	34.2	67.0	23.2	62.4	20.3	63.1
HEF [28]	53.6	75.6	55.3	71.4	47.9	73.5	16.4	60.3	25.5	66.5
Musubu [25]	44.9	76.2	45.3	65.4	42.3	72.4	28.5	64.0	35.4	75.2
TEMP [17]	57.8	85.3	49.2	77.7	47.6	81.0	29.4	65.7	35.9	73.8
BoxTaxo [11]	31.8	64.7	38.1	75.4	31.4	66.8	26.4	63.9	29.2	68.2
TaxoPrompt [31]	61.4	85.6	57.4	83.6	53.2	83.1	40.3	71.5	33.9	74.4
TaxoInstruct [22]	45.9	76.2	48.8	77.2	34.3	70.2	43.3	71.8	31.8	69.0
1-shot Setting										
NL (GPT-4o)	54.8	88.3	52.5	81.3	55.5	85.6	72.2	90.7	69.8	89.1
CodeTaxo (GPT-4o)	**67.7**	**89.2**	**62.5**	**86.1**	**58.1**	85.3	**74.5**	**91.3**	**72.9**	**91.0**
NL (GPT-4o-mini)	50.0	83.0	35.0	76.1	55.1	**87.2**	60.1	86.0	58.3	85.2
CodeTaxo (GPT-4o-mini)	58.1	85.6	42.5	76.0	55.9	85.3	68.8	89.2	61.5	85.1
5-shot Setting										
NL (GPT-4o)	56.5	84.3	60.0	85.5	52.5	86.9	72.2	90.1	69.3	90.0
CodeTaxo (GPT-4o)	**66.1**	**88.0**	**67.5**	**87.0**	**60.2**	85.7	**76.5**	**91.9**	**77.6**	**93.4**
NL (GPT-4o-mini)	53.2	84.8	42.5	80.2	57.2	87.6	63.4	87.3	63.5	88.6
CodeTaxo (GPT-4o-mini)	59.7	84.8	47.5	78.3	58.9	**87.9**	66.8	88.6	70.3	89.1

Table 2: Performance on taxonomy expansion across two small-scale taxonomies (WordNet and Graphine) and three large-scale taxonomies (SemEval2016: science, environment, food). Bold indicates the highest score; underlined indicates the second-highest. All metrics are in percentages (%).

respectively. Although NL prompting performs better than TaxoPrompt on SemEval-Food, it still shows a 4.68% decrease in accuracy compared to CodeTaxo. It suggests that CodeTaxo is more effective in representing taxonomic structures than NL and performs better than self-supervised training on existing taxonomies.

Finally, our findings indicate that the performance of CodeTaxo as an in-context learning method is consistently influenced by the inherent capabilities of LLMs and the number of demonstrations included in the prompt. Specifically, we evaluated both NL and CodeTaxo using GPT-4o and GPT-4o-mini. Across all benchmarks, the performance of both NL and CodeTaxo on GPT-4o surpasses that on GPT-4o-mini. Additionally, we assessed 5-shot and 1-shot settings for both NL and CodeTaxo, finding that as the number of high-quality demonstrations increases, the performance of both methods improves across all benchmarks.

4.3 Comparison between NL and CodeTaxo on LLMs and Code-LLMs (RQ2)

Given that CodeTaxo is a prompting method specifically designed for programming languages, we conducted a comparative analysis of its effectiveness against natural language prompting on both general-purpose LLMs and Code-LLMs. To ensure a fair evaluation, we selected models from the Llama family, specifically LLaMa-3-70B-instruct and CodeLLaMA-70B-instruct. Additionally, we included an assessment of CodeLLaMA-34B-instruct to examine the impact of model size on performance. Due to the inability of these LLMs to process large-scale taxonomies within their contextual window, our evaluation was limited to WordNet and Graphine. The experimental results are presented in Figure 5. We have several findings as follows:

First, for both the WordNet and Graphine taxonomies, CodeTaxo demonstrated superior accuracy and Wu&P scores across all tested Code-LLMs and general-purpose LLMs. This suggests that CodeTaxo is more effective in representing taxonomic structures than natural language prompts, not only for black-box general-purpose LLMs like GPT-4 but also for open-source LLMs. Furthermore, whether the LLMs are specialized in code tasks does not appear to affect CodeTaxo's efficiency, indicating its robustness as a more efficient method compared to natural language prompting.

Secondly, our analysis reveals that CodeTaxo derives greater benefit from Code-LLMs compared to natural language prompts. Specifically, when transitioning from natural language prompting to code language prompting on LLaMa-3-70B-instruct and CodeLLaMA-70B-instruct, CodeTaxo exhibited a 13.33% improvement in accuracy on WordNet, compared to a 6.51% improvement with natural language prompts. Also, we find that the performance of prompting CodeLLaMA-34B-instruct via code language is better on both WordNet and Graphine taxonomy, even though CodeLLaMA-34B-instruct is much more light than LLaMa-3-70B-instruct.

4.4 Efficiency Analysis of CodeTaxo (RQ3)

4.4.1 Semantic Similarity Filter Efficiency. This section examines the effect of selecting Top-K entities using the Semantic Similarity

Method	Def.	SemEval-Sci		SemEval-Env		SemEval-Food		WordNet		Graphine	
		Acc	Wu&P	Acc	Wu&P	Acc	Wu&P	Acc	Wu&P	Acc	Wu&P
1-shot Setting											
NL (GPT-4o)	✓	54.8	88.3	52.5	81.3	55.5	85.6	72.2	90.7	69.8	89.1
	×	59.7	89.0	57.5	82.8	56.4	87.0	68.1	89.1	68.8	90.1
CodeTaxo (GPT-4o)	✓	**67.7**	**89.2**	**62.5**	**86.1**	**58.1**	85.3	**74.5**	**91.3**	**72.9**	**91.0**
	×	56.5	84.5	55.0	85.1	56.8	86.1	66.4	88.4	69.8	88.8
5-shot Setting											
NL (GPT-4o)	✓	56.5	84.3	60.0	85.5	52.5	86.9	72.2	90.1	69.3	90.0
	×	59.7	89.6	50.0	79.3	55.5	87.6	70.5	89.9	68.8	88.9
CodeTaxo (GPT-4o)	✓	**66.1**	**88.0**	**67.5**	**87.0**	**60.2**	85.7	**76.5**	**91.9**	**77.6**	**93.4**
	×	51.6	80.6	65.0	86.7	57.6	86.1	67.8	88.8	68.8	89.7

Table 3: Impact of Entity Definition Sentences (Def.) on CodeTaxo and NL Performance in 1-Shot and 5-Shot Settings.

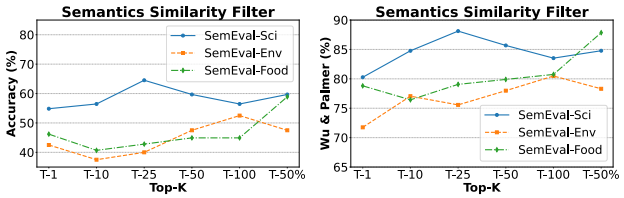


Figure 6: Effect of Top-K relevant entities selected through SimCSE-based Semantic Similarity Filter.

Dataset	1-shot		5-shot	
	NL	CodeTaxo	NL	CodeTaxo
SemEval-Sci	15737.2	9701.4	16095.1	10342.6
SemEval-Env	8965.7	5693.6	9325.0	6321.1
SemEval-Food	48908.1	30536.5	49266.4	31176.3
WordNet	948.9	1369.2	1306.4	1962.3
Graphine	2486.0	3223.9	2855.2	3893.3

Table 4: Comparison of average tokens used by NEthical ConsiderationsL and CodeTaxo across 5 benchmarks in 1-shot and 5-shot settings.

Filter on model performance. We conduct experiments using GPT-4o-mini across the three taxonomies in SemEval2016. As shown in Figure 6, across all three taxonomies, increasing the number of Top-K entities generally enhances performance. This improvement is attributed to retaining more entities, which lowers the chance of filtering out the ground truth, thereby boosting prediction accuracy. However, there is a trade-off between search space size and coverage. A smaller search space can improve accuracy by narrowing the model's focus but risks excluding the ground truth. For example, in the SemEval-Sci, the model achieved optimal performance with a Hit@25 score of 78% by retaining the top 25 entities, demonstrating the filter's effectiveness in balancing search space and coverage.

In our experiments, we retained the top 50% of entities, ensuring that Hit@n exceeded 90% across all benchmarks. This approach reduces the search space while maintaining a high likelihood of including the ground truth, thereby enhancing the model's precision in taxonomy expansion.

Setting	Config.		SemEval-Sci		SemEval-Env		SemEval-Food	
	Demo.	Filter	Acc	Wu&P	Acc	Wu&P	Acc	Wu&P
1-shot	×	×	50.0	84.0	47.5	81.1	56.4	85.3
	×	✓	61.3	84.4	55.0	83.2	54.2	84.6
	✓	×	61.3	85.9	47.5	79.0	57.2	86.7
	✓	✓	67.7	89.2	62.5	86.1	58.1	85.3
5-shot	×	×	58.1	86.0	55.0	82.8	56.4	86.5
	×	✓	59.7	84.7	57.5	83.7	58.5	85.8
	✓	×	61.3	88.5	55.0	85.3	57.6	86.5
	✓	✓	66.1	88.0	67.5	87.02	60.2	85.7

Table 5: Ablation Study of two major modules in the CodeTaxo: All metrics are presented in percentages (%). Configurations indicate whether Demonstration Selection (Demo.) and Semantic Similarity Filter (Filter) were employed.

4.4.2 Number of Tokens. Table 4 compares average token usage across benchmarks and prompt types (CodeTaxo vs. NL) in 1-shot and 5-shot settings. The findings highlight CodeTaxo's efficiency in reducing token usage while maintaining effectiveness. Notably, in SemEval2016, CodeTaxo cuts token usage by approximately 37.6% in the SemEval-Food task compared to natural language prompts. However, in the WordNet and Graphine datasets, CodeTaxo uses slightly more tokens due to the need to define Entity classes and methods. Overall, the significant reduction in token usage in SemEval2016 underscores CodeTaxo's efficiency, especially in contexts with limited token windows.

4.5 Ablation Study (RQ3)

4.5.1 Insight of Definition Sentences. We performed an ablation study on definition sentences, a vital data source for taxonomy expansion tasks, using two prompting methods: NL and CodeTaxo. Our results in Table 3 show that without definition sentences, CodeTaxo suffers a substantial drop in accuracy and Wu&P across all benchmarks in both 1-shot and 5-shot settings, highlighting its reliance on semantic information from definitions to establish taxonomic relationships. Interestingly, NL performed better without definition sentences in specific benchmarks (SemEval-Sci, SemEval-Env, SemEval-Food) in the 1-shot setting, and in SemEval-Sci and SemEval-Food in the 5-shot setting. This suggests that NL struggles to process definition information effectively, potentially leading to incorrect predictions when overloaded with definitional content.

Query	NL Output	CodeTaxo Output	CodeTaxo Explanation
functional genomics *Def.* functional genomics is a field of molecular biology that attempts to describe gene functions and interactions	**genetics** *Def.* genetics is a branch of biology concerned with the study of genes, genetic variation, and heredity in organisms.	**genomics** *Def.* genomics is an interdisciplinary field of biology focusing on the structure, function, evolution, mapping, and editing of genomes	**genomics** *Exp.* Functional genomics is a subfield of genomics that focuses on understanding the functions and interactions of genes within the genome.
geostrategy *Def.* geostrategy, a subfield of geopolitics, is a type of foreign policy guided principally by geographical factors as they inform, constrain, or affect political and military planning	**politics** *Def.* politics is the set of activities that are associated with making decisions in groups, or other forms of power relations between individuals, such as the distribution of resources or status	**geopolitics** *Def.* geopolitics on politics and international relations	**geopolitics** *Exp.* : Geostrategy is a subfield of geopolitics, which focuses on geographic factors influencing political and military planning.

Table 6: Case study comparing the outputs of the CodeTaxo and NL prompt using the SemEval-Sci benchmarks. The table presents the definitions (Def.) of each model's prediction, additionally with the CodeTaxo explanations (Exp.) provided in the last column. Yellow highlights emphasize the specific focus of the query within its definition, as correctly captured by CodeTaxo, while green highlights indicate broader, less precise concepts used by the NL model.

4.5.2 Effevtiveness of Demo. Selection and Semantic Similarity Filter. We performed an ablation study on the three SemEval2016 benchmarks mentioned above to assess the effectiveness of the two primary modules in CodeTaxo: Demonstration Selection (Demo.) and the Semantic Similarity Filter (Filter). Due to the relatively small size of the taxonomies in WordNet and Graphine, filtering redundant entities from the existing taxonomies was unnecessary. The results, presented in Table 5, indicate that selecting demonstrations related to the query entity and filtering out unrelated entities in the existing taxonomy significantly improves taxonomy expansion. This finding suggests that incorporating more relevant contextual information and reducing redundant information to narrow the search space is beneficial for both accuracy and the Wu & Palmer (Wu&P) score across all SemEval2016 benchmarks.

4.6 Case Study

This section presents a case study to illustrate the effectiveness of our CodeTaxo framework. We compare the outputs of CodeTaxo and the NL prompt, with model predictions and the corresponding definition detailed in Table 6, where CodeTaxo's predictions align with the ground truth. We additionally provide the model output using the prompt mentioned in Section 3.2.4, to allow the model to generate explanations and facilitate better discussion.

In the first case, the query *functional genomics*, which focuses on *gene functions and interactions*, is correctly classified by CodeTaxo under genomics. The explanation generated by CodeTaxo emphasizes the specific focus of functional genomics within genomics by noting its attention to the "functions and interactions of genes within the genome", reflecting its specific focus within the broader field. In contrast, the NL model incorrectly selects *genetics*, a more general term, as the parent entity.

In the second case, CodeTaxo accurately identifies *geopolitics* as the parent entity of *geostrategy*, and the explanation provided by CodeTaxo shows its emphasis on *geographic factors*. The NL model, however, selects the broader category of politics, failing to capture the specific geographic considerations inherent to *geostrategy*.

These cases demonstrate that CodeTaxo leverages definition information more effectively, enabling a nuanced understanding of taxonomy structures and leading to more precise predictions.

5 Related Works

5.1 Taxonomy Expansion

In taxonomy expansion, various approaches have been developed to integrate emerging entities into existing taxonomies. Aly *et al.* [2] and Ma *et al.* utilized hyperbolic embeddings to capture taxonomic relations, while Jiang *et al.* [11] employed box embedding instead of single vector embedding to encode taxonomic relations. Manzoor *et al.* [19] introduced implicit edge semantics to enhance entity representations. Self-supervised methods, such as Egonet [21], minipath [35], and Ego-Tree [28], have also been explored to model structural information within taxonomies. To leverage more semantic information from the textural description of entities, Liu *et al.* [17], Takeoka *et al.* [25] and Xu *et al.* [31] fine-tuned BERT-based models to leverage textual descriptions of entities. Shen *et al.* [22] proposed a unified framework combining various taxonomy construction tasks for instruction tuning. To our knowledge, CodeTaxo is the first work to perform taxonomy expansion via prompting LLMs.

5.2 Code-LLMs for Structured Tasks

Recent studies have demonstrated the strong performance of Code-LLMs in complex reasoning tasks, including symbolic reasoning [7, 18], event structure prediction [6, 30], mathematical reasoning [9], and knowledge graph construction [4, 13]. These works highlight Code-LLMs' ability to transform unstructured text into structured representations, enabling advanced reasoning tasks. In this paper,

we focus on enhancing Code-LLMs' ability to comprehend and expand existing taxonomies through emerging query entities.

6 Conclusion

In this paper, we introduce CodeTaxo, a novel approach to taxonomy expansion that leverages code-based prompts to effectively utilize the inherent knowledge within large language models (LLMs). Our method addresses key challenges in traditional taxonomy expansion by reformulating the task as a code completion problem and employing a Semantic Similarity Filtering mechanism to optimize the use of LLMs' contextual capacity. Extensive experiments on small-scale and large-scale taxonomies demonstrate that CodeTaxo achieves state-of-the-art performance, in both one-shot settings and five-shot settings. We envision CodeTaxo as a powerful framework for integrating emerging entities into existing taxonomies by accurately identifying appropriate parent entities and also providing new insights for leveraging LLMs in structured knowledge tasks.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Rami Aly, Shantanu Acharya, Alexander Ossa, Arne Köhn, Chris Biemann, and Alexander Panchenko. 2019. Every Child Should Have Parents: A Taxonomy Refinement Algorithm Based on Hyperbolic Term Embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4811–4817.

[3] Mohit Bansal, David Burkett, Gerard De Melo, and Dan Klein. 2014. Structured learning for taxonomy induction with belief propagation. In *ACL*. 1041–1051.

[4] Zhen Bi, Jing Chen, Yinuo Jiang, Feiyu Xiong, Wei Guo, Huajun Chen, and Ningyu Zhang. 2024. Codekgc: Code language model for generative knowledge graph construction. *ACM Transactions on Asian and Low-Resource Language Information Processing* 23, 3 (2024), 1–16.

[5] Georgeta Bordea, Els Lefever, and Paul Buitelaar. 2016. Semeval-2016 task 13: Taxonomy extraction evaluation (texeval-2). In *Proceedings of the 10th international workshop on semantic evaluation (semeval-2016)*. 1081–1091.

[6] Yangyi Chen, Xingyao Wang, Manling Li, Derek Hoiem, and Heng Ji. 2023. ViStruct: Visual Structural Knowledge Extraction via Curriculum Guided Code-Vision Representation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 13342–13357.

[7] Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, et al. [n. d.]. Binding Language Models in Symbolic Languages. In *The Eleventh International Conference on Learning Representations*.

[8] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783* (2024).

[9] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*. PMLR, 10764–10799.

[10] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 6894–6910.

[11] Song Jiang, Qiyue Yao, Qifan Wang, and Yizhou Sun. 2023. A single vector is not enough: Taxonomy expansion via box embeddings. In *Proceedings of the ACM Web Conference 2023*. 2467–2476.

[12] David Jurgens and Mohammad Taher Pilehvar. 2016. Semeval-2016 task 14: Semantic taxonomy enrichment. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*. 1092–1102.

[13] Peng Li, Tianxiang Sun, Qiong Tang, Hang Yan, Yuanbin Wu, Xuan-Jing Huang, and Xipeng Qiu. 2023. CodeIE: Large Code Generation Models are Better Few-Shot Information Extractors. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 15339–15353.

[14] Zixuan Li, Yutao Zeng, Yuxin Zuo, Weicheng Ren, Wenxuan Liu, Miao Su, Yucan Guo, Yantao Liu, Xiang Li, Zhilei Hu, et al. 2024. KnowCoder: Coding Structured Knowledge into LLMs for Universal Information Extraction. *arXiv preprint arXiv:2403.07969* (2024).

[15] Bang Liu, Weidong Guo, Di Niu, Jinwen Luo, Chaoyue Wang, Zhen Wen, and Yu Xu. 2020. GIANT: scalable creation of a web-scale ontology. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 393–409.

[16] Zequn Liu, Shukai Wang, Yiyang Gu, Ruiyi Zhang, Ming Zhang, and Sheng Wang. 2021. Graphine: A Dataset for Graph-aware Terminology Definition Generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 3453–3463.

[17] Zichen Liu, Hongyuan Xu, Yanlong Wen, Ning Jiang, Haiying Wu, and Xiaojie Yuan. 2021. TEMP: Taxonomy Expansion with Dynamic Margin Loss through Taxonomy-Paths. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 3854–3863.

[18] Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language Models of Code are Few-Shot Commonsense Learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 1384–1403.

[19] Emaad Manzoor, Rui Li, Dhananjay Shrouty, and Jure Leskovec. 2020. Expanding taxonomies with implicit edge semantics. In *Proceedings of The Web Conference 2020*. 2044–2054.

[20] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950* (2023).

[21] Jiaming Shen, Zhihong Shen, Chenyan Xiong, Chi Wang, Kuansan Wang, and Jiawei Han. 2020. TaxoExpan: Self-supervised taxonomy expansion with position-enhanced graph neural network. In *Proceedings of The Web Conference 2020*. 486–497.

[22] Yanzhen Shen, Yu Zhang, Yunyi Zhang, and Jiawei Han. 2024. A Unified Taxonomy-Guided Instruction Tuning Framework for Entity Set Expansion and

Taxonomy Expansion. *arXiv preprint arXiv:2402.13405* (2024).

[23] Kai Sun, Yifan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. 2024. Head-to-Tail: How Knowledgeable are Large Language Models (LLMs)? AKA Will LLMs Replace Knowledge Graphs?. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 311–325.

[24] Yushi Sun, Hao Xin, Kai Sun, Yifan Ethan Xu, Xiao Yang, Xin Luna Dong, Nan Tang, and Lei Chen. 2024. Are Large Language Models a Good Replacement of Taxonomies? *arXiv preprint arXiv:2406.11131* (2024).

[25] Kunihiro Takeoka, Kosuke Akimoto, and Masafumi Oyamada. 2021. Low-resource taxonomy enrichment with pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2747–2758.

[26] Yanchao Tan, Carl Yang, Xiangyu Wei, Chaochao Chen, Longfei Li, and Xiaolin Zheng. 2022. Enhancing recommendation with automated tag taxonomy construction in hyperbolic space. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 1180–1192.

[27] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[28] Suyuchen Wang, Ruihui Zhao, Xi Chen, Yefeng Zheng, and Bang Liu. 2021. Enquire one's parent and child before decision: Fully exploit hierarchical structure for self-supervised taxonomy expansion. In *Proceedings of the Web Conference 2021*. 3291–3304.

[29] Suyuchen Wang, Ruihui Zhao, Yefeng Zheng, and Bang Liu. 2022. QEN: Applicable Taxonomy Completion via Evaluating Full Taxonomic Relations. In *Proceedings of the ACM Web Conference 2022*. 1008–1017.

[30] Xingyao Wang, Sha Li, and Heng Ji. 2023. Code4Struct: Code Generation for Few-Shot Event Structure Prediction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 3640–3663.

[31] Hongyuan Xu, Yunong Chen, Zichen Liu, Yanlong Wen, and Xiaojie Yuan. 2022. TaxoPrompt: A Prompt-based Generation Method with Taxonomic Context for Self-Supervised Taxonomy Expansion.. In *IJCAI*. 4432–4438.

[32] Shuo Yang, Lei Zou, Zhongyuan Wang, Jun Yan, and Ji-Rong Wen. 2017. Efficiently answering technical questions—a knowledge graph approach. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[33] Hongbin Ye, Ningyu Zhang, Hui Chen, and Huajun Chen. 2022. Generative Knowledge Graph Construction: A Review. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 1–17.

[34] Xiaoxin Yin and Sarthak Shah. 2010. Building taxonomy of web search intents for name entity queries. In *Proceedings of the 19th international conference on World wide web*. 1001–1010.

[35] Yue Yu, Yinghao Li, Jiaming Shen, Hao Feng, Jimeng Sun, and Chao Zhang. 2020. Steam: Self-supervised taxonomy expansion with mini-paths. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1026–1035.

[36] Yuchen Zhang, Amr Ahmed, Vanja Josifovski, and Alexander Smola. 2014. Taxonomy discovery for personalized recommendation. In *Proceedings of the 7th ACM international conference on Web search and data mining*. 243–252.