EMPOWER ENTITY SET EXPANSION VIA LANGUAGE MODEL PROBING

BY

YUNYI ZHANG

THESIS

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

Urbana, Illinois

Adviser:

Professor Jiawei Han

ABSTRACT

Entity set expansion, aiming at expanding a small seed entity set with new entities belonging to the same semantic class, is a critical task that benefits many downstream NLP and IR applications, such as question answering, query understanding, and taxonomy construction. Existing set expansion methods bootstrap the seed entity set by adaptively selecting context features and extracting new entities. A key challenge for entity set expansion is to avoid selecting ambiguous context features which will shift the class semantics and lead to accumulative errors in later iterations. In this study, we propose a novel iterative set expansion framework that leverages automatically generated class names to address the semantic drift issue. In each iteration, we select one positive and several negative class names by probing a pre-trained language model, and further score each candidate entity based on selected class names. Experiments on two datasets show that our framework generates high-quality class names and outperforms previous state-of-the-art methods significantly.

To my friends and family, for their love and support.

ACKNOWLEDGMENTS

I would like to first express my appreciation to my advisor Professor Jiawei Han of the Department of Computer Science at University of Illinois Urbana-Champaign. He has provided me with insightful suggestions and guidance throughout my master study. His encouragement and patience have led me towards good research topics and consolidated my decision to pursue a Ph.D. program to continue my research study. I would also like to thank all Data Mining Group members for their inspiring advises, especially Jiaming Shen and Professor Jingbo Shang for their valuable help in this thesis, and Yu Meng for his collaborations in the past two years.

Research was supported in part by US DARPA KAIROS Program No. FA8750-19-2-1004 and SocialSim Program No. W911NF-17-C-0099, National Science Foundation IIS-19-56151, IIS-17-41317, and IIS 17-04532, and the Molecule Maker Lab Institute: An AI Research Institutes program supported by NSF under Award No. 2019897. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily represent the views, either expressed or implied, of DARPA or the U.S. Government.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Entity set expansion aims to expand a small set of seed entities (e.g., {"*United States*", "*China*", "*Canada*"}) with new entities (e.g., "*United Kingdom*", "*Australia*") belonging to the same semantic class (i.e., `Country`). The entities so discovered may benefit a variety of NLP and IR applications, such as question answering [1], query understanding [2], taxonomy construction [3], and semantic search [4, 5].

Most existing entity set expansion methods bootstrap the initial seed set by iteratively selecting context features, while extracting and ranking new entities. For example, SetExpan [6] selects quality skip-gram features based on the current set and ranks new entities using an ensemble mechanism in each iteration; SetCoExpan automatically generates several related but wrong entities to serve as negatives for better context feature selection. A key challenge to these set expansion methods is to avoid selecting *ambiguous patterns* that may introduce erroneous entities from other non-target semantic classes. Take the above class `Country` as an example, we may find some ambiguous patterns like "** located at*" (which will match more general `Location` entities) and "*match against **" (which may be associated with entities in the `Sports Club` class). Including such ambiguous patterns to select new entities will unavoidably introduce wrong entities to the expanded set.

Furthermore, as bootstrapping is an iterative process, those erroneous entities added at early iterations may shift the class semantics, leading to inferior expansion quality at later iterations. Some previous studies propose to incorporate external knowledge to address such issue, including information from other semantic classes or provided by human. [7] assumes different queries are from different semantic classes and thus mutually exclusive to each other. However, such an assumption may not hold if and we do not have any prior knowledge on how the target sets are selected. [8] leverages human-provided negative example entities to expand the target set, which requires domain experts to predict possible errors for each target set and select negative examples accordingly. These methods all require additional knowledge on the dataset, and addressing such "semantic drift" issue without requiring additional user inputs remains an open research problem.

In this study, we propose to empower entity set expansion with class names automatically generated from pre-trained language models [9, 10, 11]. Intuitively, knowing the class name is "country", instead of "state" or "city", can help us identify *unambiguous patterns* and eliminate erroneous entities like "*Europe*" and "*New York*". Moreover, we can acquire such knowledge (i.e., positive and negative class names) by probing a pre-trained language model automatically without relying on human annotated data. We can achieve this by automati-

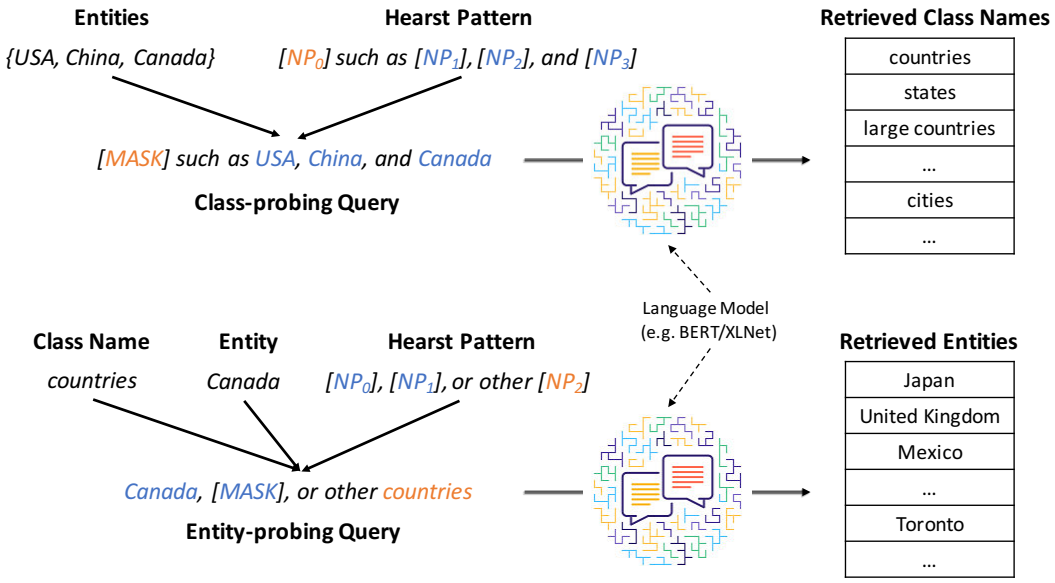

Figure 1.1: Examples of class-probing and entity-probing queries generated based on Hearst patterns.

cally constructing different probing queries using Hearst patterns [12] with hypernyms being masked and feeding them to a pre-trained masked language model. The model will predict the most likely terms to replace the masked token in the queries which in our case are the possible class names.

Motivated by the above intuition, we propose CGExpan, a new iterative framework for entity set expansion that consists of three modules: (1) The first, *class name generation* module, constructs and submits class-probing queries (e.g., "[MASK] such as USA, China, and Canada." in Fig. 1.1) to a language model for retrieving a set of candidate class names. (2) The second, *class name ranking* module, builds an entity-probing query for each candidate class name and retrieves a set of entities. The similarity between this retrieved set and the current entity set serves as a proxy for the class name quality, based on which we rank all candidate class names. An unsupervised ensemble technique [6] is further used to improve the quality of final ranked list from which we select one best class name and several negative class names. (3) The third, *class-guided entity selection* module, scores each entity conditioned on the above selected class names and adds top-ranked entities into the currently expanded set. As better class names may emerge in later iterations, we score and rank all entities (including those already in the expanded set) at each iteration, which helps alleviate the semantic drift issue.

Contributions. In summary, this study makes the following contributions:

1. We propose a new set expansion framework that leverages class names to guide the expansion process and enables filtration of the entire set in each iteration to resolve the semantic drift issue;

2. we design an automatic class name generation algorithm that outputs high-quality class names by dynamically probing pre-trained language models;

3. experiments on two public datasets from different domains demonstrate the superior performance of our approach compared with state-of-the-art methods.

The remaining is organized as follows: Chapter 2 provides some preliminaries for our method; our proposed framework CGExpan is detailed in Chapter 3; Chapter 4 includes our experiment results, together with ablation study, parameter analysis, and case studies; a short survey on related works is presented in Chapter 5, and Chapter 6 concludes the paper with some possible future works.

CHAPTER 2: BACKGROUND

In this section, we provide background on language models and define the entity set expansion problem.

2.1 LANGUAGE MODEL

A standard language model (LM) inputs a word sequence $\mathbf{w} = [w_1, w_2, \ldots, w_n]$ and assigns a probability $\mathbf{P}(\mathbf{w})$ to the whole sequence. Recent studies [9, 10, 11] found that language models, simply trained for next word or missing word prediction, can generate high quality *contextualized* word representations which benefit many downstream applications. Specifically, these language models will output an embedding vector for each *word appearance* in a specific context that is usually the entire sentence where the target word occurs, rather than just words appearing before the target word. Therefore, we can also view a LM as a model that inputs a word sequence $\mathbf{w}$ and outputs a probability $\mathbf{P}(w_i) = \mathbf{P}(w_i|w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_n)$ to any position $1 \leq i \leq n$.

Currently, [10] proposes BERT and train the language model with two objectives: (1) a cloze-filling objective which randomly substitutes some words with a special [MASK] token in the input sentence and forces LM to recover masked words, and (2) a binary classification objective that guides LM to predict whether one sentence directly follows another (sentence). BERT leverages Transformer [13] architecture and is learned on English Wikipedia as well as BookCorpus. In this study, we use such model in two ways: (1) use Masked Language Model to predict class names and apply class name guidance, and (2) use average contextualized embeddings for each entity as its global context-free features. Chapter 3 describes more technical details.

Other LM architectures are described in Chapter 5.

2.2 PROBLEM FORMULATION

We will first define some key concepts used in this paper and then present our problem formulation for entity set expansion task.

Definition 2.1 (Entity)**.** *An entity e is a word or a phrase that refers to a real-world instance. For example, "*U.S.*" refers to the country: United States.*[1]

[1]In this work, we will use "term" and "entity" interchangeably.

In the task of entity set expansion, all the target semantic classes are assumed to consist of entities. We observe that such semantic classes always have some textual descriptions that can uniquely identify them without ambiguity. Therefore, we propose to automatically generate and utilize these textual descriptions, defined as *class names* below, to guide the set expansion algorithm.

Definition 2.2 (Class Name)**.** *A class name c is a text representation of a semantic class. For instance,* `country` *could be a class name for the semantic class that includes entities like "*United States*" and "*China*".*

To automatically predict and leverage the class names without requiring extra human inputs, we use a pre-trained masked language model as a knowledge base and construct probing queries to extract information from it.

Definition 2.3 (Probing Query)**.** *A probing query is a sequence of tokens that contain one [*Mask*] token.*

In this work, we utilize Hearst patterns [12], containing 6 lexico-syntactic patterns originally designed for hypernym detection task, to construct the probing queries. Based on different usage scenarios, we further define two types of probing queries for class name prediction and class-guided set expansion, respectively.

Definition 2.4 (Class-probing Query)**.** *A class-probing query aims to predict the class name of some given entities by masking the hypernym term in a Hearst pattern.*

Definition 2.5 (Entity-probing Query)**.** *An entity-probing query aims to retrieve new entities given the class name and several entities in the target semantic class by masking one of the hyponym terms in a Hearst pattern.*

For example, in Figure 1.1, "[Mask] such as USA, China, and Canada" is a class-probing query, and the pre-trained LM will predict "countries" for the masked token; "Canada, [Mask], or other countries" is an entity-probing query, and the pre-trained LM will replace the masked token by other country entities. Note that the LM can only do token level prediction, and we detail how multi-token class names are generated in Section 3.1.

Definition 2.6 (Problem Formulation)**.** *Given a text corpus $\mathcal{D}$ and a seed set of user-provided entities E^0, we aim to output a ranked list of entities that belong to the same semantic class.*

In each iteration of our proposed framework, we will first generate a set $\mathcal{C}$ of possible class names for the entity set, then we select one positive class name c_p and several negative class names C_N based on the given corpus, and finally we use these class names to rank and select new entities to expand the set.

Example 2.1. *Given a text corpus and a seed set of three country entities* { "United States", "China", "Canada"}, *our task is to return a ranked list of entities belonging to the same* `country` *class such as* "United Kingdom", "Japan", *and* "Mexico". *In the expansion process, our algorithm will predict the class name* `country` *as well as some closely related by wrong class names like* `state` *and* `city`.

CHAPTER 3: CLASS-GUIDED ENTITY SET EXPANSION

We introduce our class-guided entity set expansion framework in this section. First, we present our class name generation and ranking modules in Sections 3.1 and 3.2, respectively. Then, we discuss how to leverage class names to guide the iterative expansion process in Section 3.3.

3.1 CLASS NAME GENERATION

The class name generation module inputs a small collection of entities and generates a set of candidate class names for these entities. We build this module by automatically constructing class-probing queries and iteratively querying a pre-trained LM to obtain multi-gram class names.

First, we notice that the class name generation goal is similar to the hypernymy detection task which aims to find a general hypernym (e.g., "mammal") for a given specific hyponym (e.g., "panda"). Therefore, we leverage the six Hearst patterns [12] that are widely used for hypernymy detection, to construct the class-probing query. Following are the patterns we used in our framework, where for each pattern, the noun phrase y is a hypernym of the noun phrase a:

1. NP_y such as NP_a
2. such NP_y as NP_a
3. NP_a or other NP_y
4. NP_a and other NP_y
5. NP_y, including NP_a
6. NP_y, especially NP_a

More specifically, we randomly select three entities in the current set as well as one Hearst pattern (out of six choices) to construct one query. For example, we may choose entities {"*China*", "*India*", "*Japan*"} and pattern "NP_y such as NP_a, NP_b, and NP_c" to construct the query "[MASK] such as China, India, and Japan". By repeating such a random selection process, we can construct a set of queries and feed them into pre-trained language models to obtain predicted masked tokens which are viewed as possible class names.

The above procedure has one limitation—it can only generate unigram class names. To obtain multi-gram class names, we design a modified beam search algorithm to iteratively query a pre-trained LM. Specifically, after we query a LM for the first time and retrieve top K most likely words (for the masked token), we construct K new queries by adding each

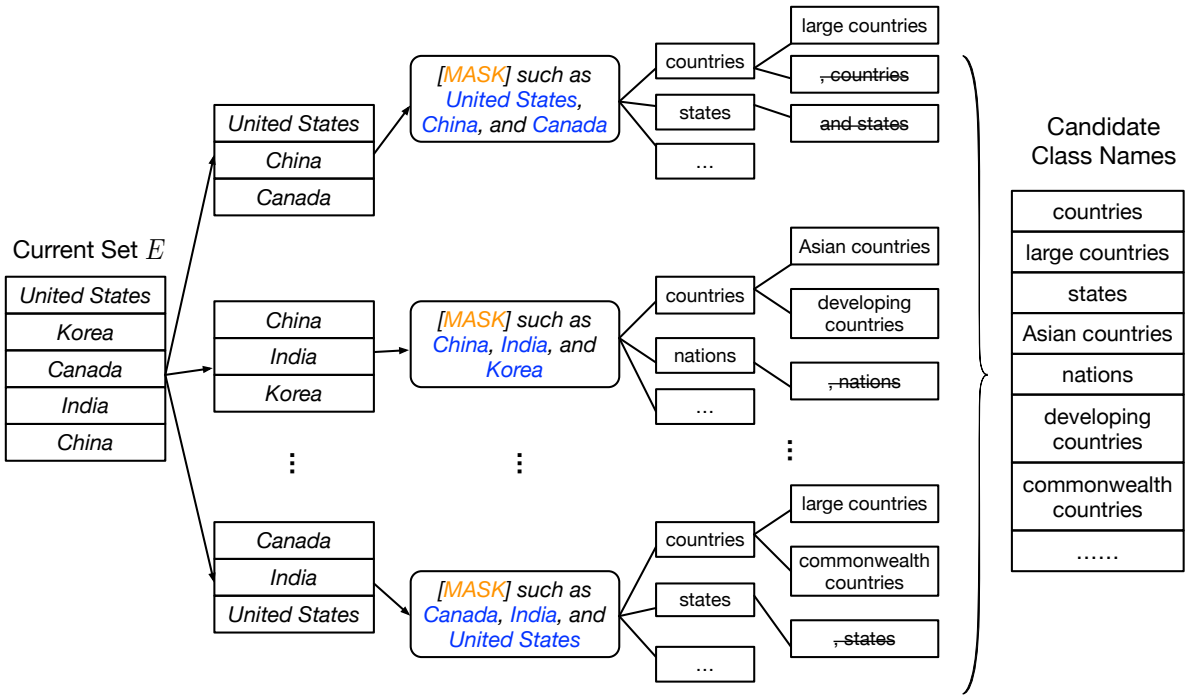

Figure 3.1: An example of Class Name Generation Module of CGExpan framework.

retrieved word after the masked token. Taking the former query "[Mask] such as China, India, and Japan" as an example, we may first obtain words like "countries", "nations", and then construct a new query "[Mask] countries such as China, India, and Japan". Probing the LM again with this new query, we can get words like "Asian" or "large", and thus obtain more fine-grained class names like "Asian countries" or "large countries". We repeat this process for maximum three times and keep all generated class names that are noun phrases. By doing so, the class names like "and countries" and ", countries" will be filtered out. As a result, for each Hearst pattern and randomly selected three entities from the current set, we will obtain a set of candidate class names. Finally, we use the union of all these sets as our candidate class name pool, denoted as $\mathcal{C}$. Note that in this module, we focus on the recall of candidate class name pool $\mathcal{C}$, without considering its precision, since the next module will further rank and select these class names based on the provided text corpus.

Example 3.1. *In Figure 3.1, given the current entity set consisting of 5 countries, each time we random sample 3 entities and one Hearst pattern to construct a class-probing query. For example, one such query could be "[Mask] such as USA, China, and Canada", from which we query the LM by iteratively expanding the class name slot and generate names like* `countries` *and then* `large countries`*. After repeating the sampling process, we aggregate all the generated class names to form our candidate class name pool. This pool likely includes*

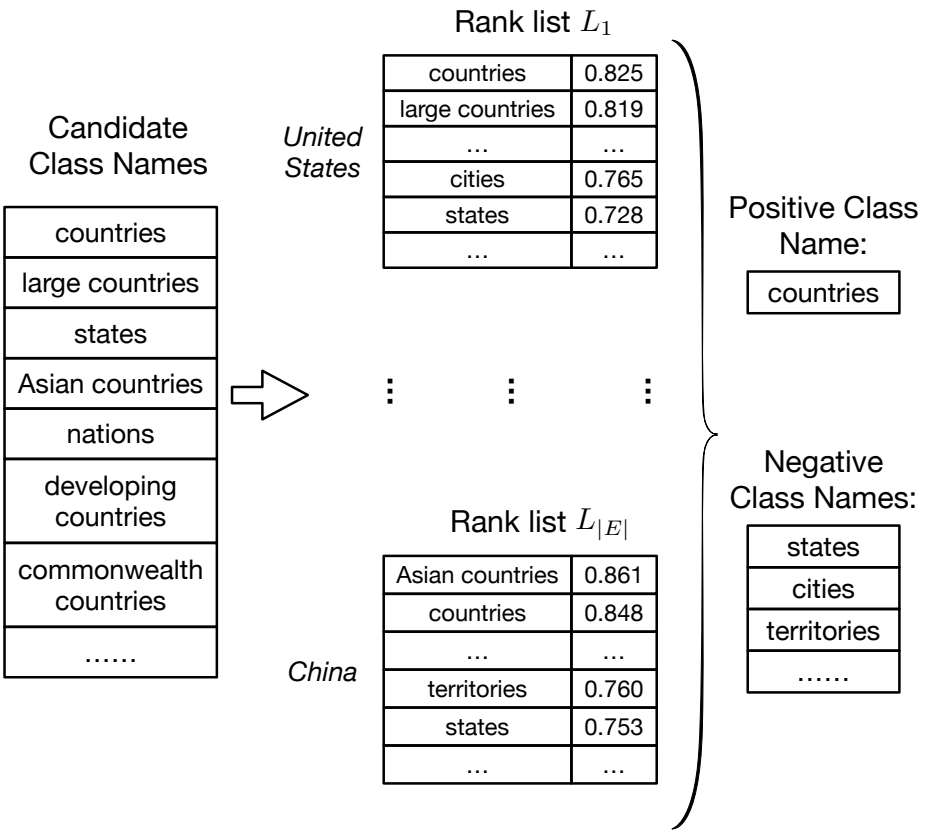

Figure 3.2: An example of Class Name Ranking Module of CGExpan framework.

some related but wrong class names, like `states` *in our example, and we will further rank them using the corpus in our next step.*

3.2 CLASS NAME RANKING

In this module, we rank the above generated candidate class names to select one best class name that represents the whole entity set and some negative class names used in the next module to filter out wrong entities. A simple strategy is to rank these class names based on the number of times it has been generated in the previous module. However, such a strategy is sub-optimal because short unigram class names always appear more frequently than longer multi-gram class names. Therefore, we propose a new method below to measure how well each candidate class name represents the entity set.

First, we introduce a corpus-based similarity measure between an entity e and a class name c. Given the class name c, we first construct 6 *entity-probing* queries by masking the hyponym term in six Hearst patterns. For example, a query for class name "countries"

could be "countries such as [MASK]". Then we query a pre-trained LM to obtain the set of six [MASK] token embeddings, denoted as X_c. Moreover, we use X_e to denote the set of all contextualized representations of the entity e in the given corpus. Then, we define the similarity between e and c, as:

$$M^k(e,c) = \frac{1}{k} \max_{X \subseteq X_e, |X|=k} \sum_{\mathbf{x} \in X} \max_{\mathbf{x}' \in X_c} \mathbf{cos}(\mathbf{x}, \mathbf{x}'), \tag{3.1}$$

where $\mathbf{cos}(\mathbf{x}, \mathbf{x}')$ is the cosine similarity between two vectors $\mathbf{x}$ and $\mathbf{x}'$. The inner max operator finds the maximum similarity between each occurrence of e and the set of entity-probing queries constructed based on c. The outer max operator identifies the top-k most similar occurrences of e with the queries and then we take their average as the final similarity between the entity e and the class name c. This measure is analogous to finding k best *occurrences* of entity e that matches to any of the probing queries of class c, and therefore it improves the previous similarity measures that utilize only the *context-free* representations of entities and class names (e.g., Word2Vec).

After we define the entity-class similarity score, we can choose one entity in the current set and obtain a ranked list of candidate class names based on their similarities with this chosen entity. Then, given an entity set E, we can obtain $|E|$ ranked lists, $L_1, L_2, \ldots, L_{|E|}$, one for each entity in E. Finally, we follow [6] and aggregate all these lists to a final ranked list of class names based on the score $s(c) = \sum_{i=1}^{|E|} \frac{1}{r_c^i}$, where r_c^i indicates the rank position of class name c in ranked list L_i. This final ranked list shows the order of how well each class name can represent the current entity set. Therefore, we choose the best one that ranks in the first position as the positive class , denoted as c_p.

Aside from choosing the positive class name c_p, we also select a set of negative class names for the target semantic class to help bound its semantics. To achieve this goal, we assume that entities in the initial user-provided seed set E^0 definitely belong to the target class. Then, we choose those class names that rank lower than c_p in *all* lists corresponding to entities in E^0, namely $\{L_i | e_i \in E^0\}$, and treat them as the negative class names. We refer to this negative set of class names as C_N and use them to guide the set expansion process below.

Example 3.2. *Figure 3.2 shows an illustration for class name ranking module, where for simplicity we show the ranked class names using 2 entities from the current set. For example, the first ranked list is generated using the similarity score between each candidate and the entity "*United States*". Then, we will aggregate these lists by mean reciprocal rank. In this example, the score for* `countries` *will be* $\frac{1}{1} + \frac{1}{2} = \frac{3}{2}$*. We use the class name with the highest*

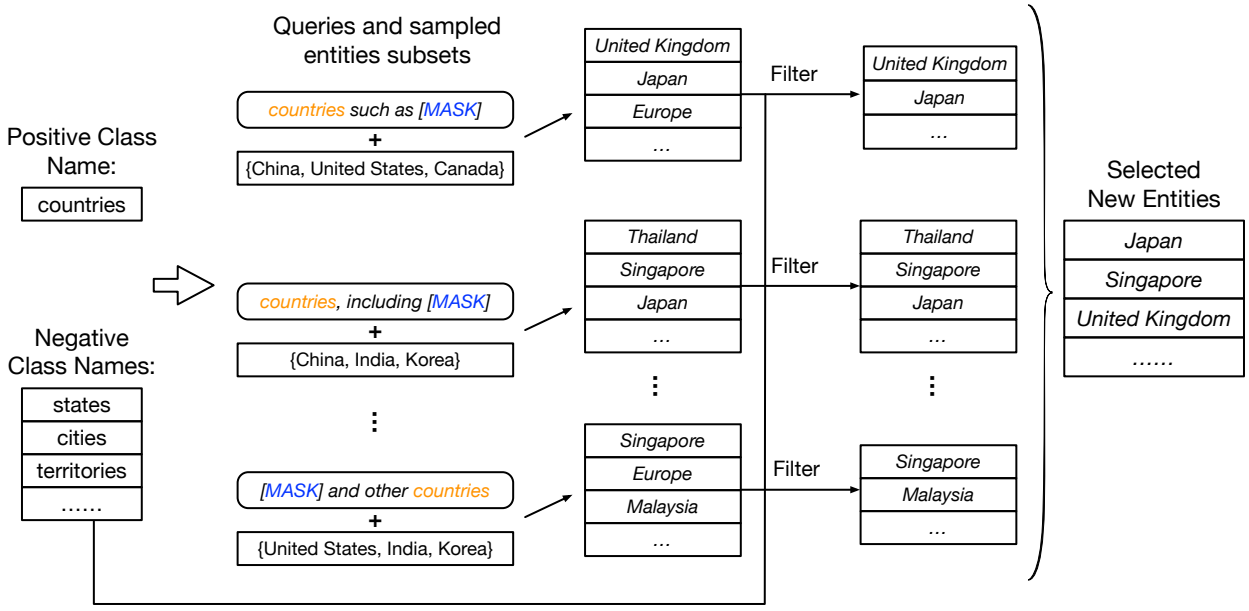

Figure 3.3: An example of Class-Guided Entity Selection Module of CGExpan framework.

score as our positive class name (`countries`), and those ranked consistently lower than it as negatives (`states`).

3.3 CLASS-GUIDED ENTITY SELECTION

In this module, we leverage the above selected positive and negative class names to help select new entities to add to the set. We first introduce two entity scoring functions and then present a new rank ensemble algorithm for entity selection.

The first function utilizes the positive class name c_p and calculates each entity e_i's score:

$$\text{score}_i^{loc} = M^k(e_i, c_p), \tag{3.2}$$

where M^k is defined in Eq. (3.1). We refer to this score as a *local* score because it only looks at top-k best occurrences in the corpus where the contextualized representation of entity e_i is most similar to the representation of class name c_p.

The second scoring function calculates the similarity between each candidate entity and existing entities in the current set, based on their *context-free* representations. For each entity e, we use the average of all its contextualized embedding vectors as its context-free representation, denoted as $\mathbf{v}_e$. Given the current entity set E, we first sample several entities

from E, denoted as E_s, and calculate the score for each candidate entity e_i as:

$$\text{score}_i^{glb} = \frac{1}{|E_s|} \sum_{e \in E_s} \mathbf{cos}(\mathbf{v}_{e_i}, \mathbf{v}_e). \tag{3.3}$$

Note here we sample a small set E_s (typically of size 3), rather than using the entire set E. Since the current entity set E may contain wrong entities introduced in previous steps, we do not use all the entities in E and compute the candidate entity score only once. Instead, we randomly select multiple subsets of entities from the current set E, namely E_s, obtain a ranked list of candidate entities for each sampled subset, and aggregate all ranked lists to select the final entities. Such a sampling strategy can reduce the effect of using wrong entities in E, as they are unlikely to be sampled multiple times, and thus can alleviate potential errors that are introduced in previous iterations. We refer to this score as a *global* score because it utilizes context-free representations which better reflect entities' overall positions in the embedding space and measure the entity-entity similarity in a more global sense. Such a global score complements the above local score and we use their geometric mean to finally rank all candidate entities:

$$\text{score}_i = \sqrt{\text{score}_i^{loc} \times \text{score}_i^{glb}}. \tag{3.4}$$

As the expansion process iterates, wrong entities may be included in the set and cause semantic drifting. We develop a novel rank ensemble algorithm that leverages those selected class names to improve the quality and robustness of entity selection. First, we repeatedly sample E_s (used for calculating score_i^{glb} in Eq. (3.3)) T times from current entity set E, and obtain T entity ranked lists $\{R^m\}_{m=1}^{T}$. Second, we follow the class name ranking procedure in Section 3.2 to obtain $|E|$ class ranked lists $\{L^n\}_{n=1}^{|E|}$, one for each entity $e_i \in E$. Note here each L^n is actually a ranked list over $\{c_p\} \cup C_N$, namely the set of selected one positive class name and all negative class names. Intuitively, an entity belonging to our target semantic class should satisfy two criteria: (1) it appears at the top positions in multiple entity ranked lists, and (2) within its corresponding class ranked list, the selected best class name c_p should be ranked above any one of the negative class name in C_N. Combining these two criteria, we define a new rank aggregation score as follows:

$$\begin{aligned} S(e_i) = &\sum_{t=1}^{T} \big(\mathbb{1}(e_i \in E) + s^t(e_i)\big) \\ &\times \mathbb{1}(r_{c_p}^i < \min_{c' \in C_N} r_{c'}^i), \end{aligned} \tag{3.5}$$

where $\mathbb{1}(\cdot)$ is an indicator function, r_c^i is the rank of class name c in entity e_i's ranked list L_c^i, and $s^t(e_i)$ the individual aggregation score of e_i deduced from the ranked list R^t, for which we test two aggregation methods: (1) mean reciprocal rank, where

$$s^t(e_i) = \frac{1}{r_i^t} \tag{3.6}$$

and r_i^t is the rank of entity e_i in the t-th ranked list R^t; and (2) the combination of scores (CombSUM), where

$$s^t(e_i) = \frac{\text{score}_i^t - \min_{e_j \in R^t} \text{score}_j^t}{\max_{e_j \in R^t} \text{score}_j^t - \min_{e_j \in R^t} \text{score}_j^t} \tag{3.7}$$

is the ranking score of e_i in the ranked list R^t after min-max feature scaling.

To interpret Eq. (3.5), the first summation term reflects our criterion (1) and its inner indicator function ensuring an entity in the current set E prone to have a large rank aggregation score if not been filtered out below. The second term reflects our criterion (2) by using an indicator function that filters out all entities which are more similar to a negative class name than the positive class name. Note here we calculate the aggregation score for all entities in the vocabulary list, including those already in the current set E, and it is possible that some entity in E will be filtered out because it has 0 value in the second term. This makes a huge difference comparing with previous iterative set expansion algorithms which all assume that once an entity is included in the set, it will stay in the set forever. Consequently, our method is more robust to the semantic drifting issue than previous studies.

Example 3.3. *As shown in Figure 3.3, each time we sample one Hearst pattern and several entities from the current set. If we take the first row as an example, the pattern is used to construct an entity probing query "countries such as [*MASK*]" to calculate local score with Eq. 3.2, and the sampled entities* {"China", "United States", "Canada"} *are used to calculate the global score with Eq. 3.3. Two scores will be combined to get a ranked list of new entities. We repeat this process and aggregate all such ranked lists, where we filter out those entities more similar to any negative class name (e.g.,* "Europe" *which is more similar to* `states` *than* `countries`*).*

3.4 SUMMARY

Algorithm 3.1 summarizes the CGExpan framework. Starting with a small seed entity set, we iteratively apply the above three modules to obtain an entity ranked list and add top-ranked entities into the set. We repeat the whole process until either (1) the expanded

set reaches a pre-defined target size or (2) the size of the set does not increase for three consecutive iterations. Notice that, by setting a large target size, more true entities belonging to the target semantic class will be selected to expand the set, which increases the recall, but wrong entities are also more likely to be included, which decreases the precision. However, as the output of the set expansion framework is a ranked list, the most confident high-quality entities will still be ranked high in the list.

Algorithm 3.1: CGExpan Framework

Input: Candidate entities V, initial seed set E^0, expected set size S, model parameters $\{T_1, T_2, k, r\}$.
Output: The expanded set E.
$E = E^0$.
while $|E| < S$ *and* $|E|$ *not increases for 3 iterations* **do**
  `// Class Name Generation (Sec. 3.1)`
  $\mathcal{C} = \{\}$
  for $t = 1, 2, \ldots, T_1$ **do**
    Sample 3 entities from E.
    Sample a Hearst pattern p.
    Generate a set of candidate class names $\mathcal{C}_t$ up to tri-gram.
    $\mathcal{C} = \mathcal{C} \cup \mathcal{C}_t$
  end
  `// Class Name Ranking (Sec. 3.2)`
  for $i = 1, 2, \ldots, |E|$ **do**
    Calculate similarity between E_i and each $c \in \mathcal{C}$ via Eq. (3.1) with k
    Construct a ranked list of class names L_i
  end
  Calculate mrr score from the ranked lists $\{L\}$
  $c_p = \max_{c \in \mathcal{C}} mrr(c)$
  $C_N = \{c | c$ ranked lower than c_p for $L_1, \ldots, L_{|E^0|}\}$
  `// Class-Guided Entity Selection (Sec. 3.3)`
  Calculate the local scores between c_p and each candidate entity via Eq. (3.2)
  for $t = 1, 2, \ldots, T_2$ **do**
    Sample 3 entities E_s from E
    Calculate the global scores for each candidate entity using E_s via Eq. (3.3)
    Calculate the combined scores via Eq. (3.4)
    Construct the ranked list of entities R^t.
  end
  Calculate the $mmrr$ score based on $\{R\}$ and $\{L\}$
  $E = E \cup \{e | mmrr(e) > \frac{T_2}{r}\}$
end
Return E.

CHAPTER 4: EXPERIMENTS

4.1 EXPERIMENT SETUP

In this section, we first introduce our experiment setup, including the datasets, evaluation metric, compared baseline methods, and some implementation details for our proposed method.

4.1.1 Datasets

We conduct our experiments on two public benchmark datasets widely used in previous studies [6, 14, 15]:

- **Wiki**, which is a subset of English Wikipedia articles from May 2011, and
- **APR**, which contains all news articles published by Associated Press and Reuters in 2015.

Following previous works, we apply a phrase mining tool, AutoPhrase [16], to construct the entity vocabulary list for APR from the corpus. The vacabulary of Wiki is extracted using distant supervision. We use the same 8 semantic classes for the Wiki dataset as well as 3 semantic classes for the APR dataset. Each semantic class has 5 seed sets to expand and each set contains 3 seed entities. Please refer to the original paper [6] for how the queries and ground truth semantic classes are constructed. Table 4.1 summarizes the statistics for these datasets.

4.1.2 Compared methods

We compare the following corpus-based entity set expansion methods.

1. **Egoset** [17]: This is a multifaceted set expansion system using context features and Word2Vec embeddings. The original framework aims to expand the set in multiple facets. Here we treat all expanded entities as in one semantic class due to little ambiguity in the seed set.
2. **SetExpan** [6]: This method iteratively selects skip-gram context features using the current set from the corpus and develops a rank ensemble mechanism to score and select new entities.[1]

[1] `https://github.com/mickeystroller/SetExpan`

Dataset	# Test Queries	# Entities	# Sentences
Wiki	40	33K	1.50M
APR	15	76K	1.01M

Table 4.1: Datasets statistics

3. **SetExpander** [18]: This method trains different embeddings based on different types of context features and leverages additional human-annotated sets to build a classifier on top of learned embeddings to predict whether an entity belongs to the set. [2]

4. **CaSE** [19]: This method combines entity skip-gram context feature and embedding features to score and rank entities once from the corpus. The original paper has three variants and we use the CaSE-W2V variant since it is the best model claimed in the paper. [3]

5. **MCTS** [14]: This method bootstraps the initial seed set by combing the Monte Carlo Tree Search algorithm with a deep similarity network to estimate delayed feedback for pattern evaluation and to score entities given selected patterns. [4]

6. **CGExpan**: This method is our proposed Class-Guided Set Expansion framework, using BERT [10] as the pre-trained language model. We include two versions of our full model, namely CGExpan-Comb and CGExpan-MRR, that use the combination of score and mean reciprocal rank for rank aggregation, respectively.

4.1.3 Evaluation Metric

We follow previous studies and evaluate set expansion results using Mean Average Precision at different top K positions (MAP@K) as below:

$$\text{MAP@}K = \frac{1}{|Q|} \sum_{q \in Q} \text{AP@}K(L^q, S^q), \tag{4.1}$$

where Q is the set of all queries and, for each query q, we use AP@$K(L^q, S^q)$ to denote the average precision at position K given a ranked list of entities L^q and a ground-truth set S^q

[2] https://github.com/NervanaSystems/nlp-architect/tree/master/nlp_architect/solutions/set_expansion

[3] https://github.com/PxYu/entity-expansion

[4] https://github.com/lingyongyan/mcts-bootstrapping

including all entities for the target semantic class:

$$\text{AP@}K = \sum_{i=1}^{K} \frac{\mathbb{1}(L_i^q \in S^q) \times \text{P@}i}{\min(K, |S^q|)} \tag{4.2}$$

and $\text{P@}i = \frac{1}{i}\sum_{r=1}^{i} \mathbb{1}(L_r^q \in S^q)$ is the traditional precision at i.

4.1.4 Implementation Details

For CGExpan, we use BERT-base-uncased provided in Huggingface's `Transformers` library [20] as our pre-trained LM and do not perform any further fine-tuning on our datasets. In principle, other masked LMs such as RoBERTa and XLNet can also be used in our framework, which we leave for future studies. To get the contextualized embedding vectors for an entity in the corpus, we first substitute the entire entity with a [MASK] token and use the output of the last Transformer layer for this [MASK] token as the embedding vector for this entity in this context. By doing this, we can better compare an entity's context with our probing queries (e.g. "countries such as [MASK].")

In the class name generation module (Sec. 3.1), each time we randomly sample 3 entities in the current set and one Hearst pattern to construct the initial probing query for the root node. When we grow the tree, we take top-3 predicted tokens in each level of beam search and set the maximum length of generated class names up to 3. Furthermore, in each iteration, we sample 30 entities subsets from current set with replacement and take the union of the generated class names for these 30 class name trees as our candidate class names. When calculating similarity between an entity and a class name (Eq. (3.1)), we use $k = 5$ occurrences of the entity that are most similar to any entity-probing query constructed with the class name in most experiment. Later we will provide a parameter study on k in the experiment in Section 4.2.4, showing that our model is insensitive to the choice of k as long as its value is larger than 5. Finally, in the class-guided entity selection module (Sec. 3.3), we randomly sample 18 size-3 entity subsets to get 18 entity rank lists for later rank ensemble.

Also, since MAP@K for $K = 10, 20, 50$ are typically used for set expansion evaluations, we follow the convention and choose 50 as the target set size in our experiments. The code and data are available at `https://github.com/yzhan238/CGExpan`

Methods	Wiki			APR		
	MAP@10	MAP@20	MAP@50	MAP@10	MAP@20	MAP@50
Egoset [17]	0.904	0.877	0.745	0.758	0.710	0.570
SetExpan [6]	0.944	0.921	0.720	0.789	0.763	0.639
SetExpander [18]	0.499	0.439	0.321	0.287	0.208	0.120
CaSE [19]	0.897	0.806	0.588	0.619	0.494	0.330
MCTS [14]	0.980^{∇}	0.930^{∇}	0.790^{∇}	0.960^{∇}	0.900^{∇}	0.810^{∇}
CGExpan-Comb	0.991	0.974	0.895	0.983	0.984	0.937
CGExpan-MRR	**0.995**	**0.978**	**0.902**	**0.992**	**0.990**	**0.955**

Table 4.2: Mean Average Precision on Wiki and APR. "$^{\nabla}$" means the number is directly from the original paper.

4.2 EXPERIMENT RESULTS

In this section, we present and analyze our main experiment results, together with an ablation study and a parameter study for our similarity calculation.

4.2.1 Overall Performance

Table 4.2 shows the overall performance of different entity set expansion methods. We can see that the two variants of CGExpan can consistently outperform all the baselines on both Wiki and APR datasets in terms of top-10/20/50 MAP scores. Comparing with SetExpan, the CGExpan-MRR achieves 24% improvement in MAP@50 on the Wiki dataset and 49% improvement in MAP@50 on the APR dataset, which verifies that our class-guided model can refine the expansion process and reduce the effect of erroneous entities on later iterations.

We observe that, iterative bootstrapping methods, including Egoset, SetExpan, MCTS, and CGExpan, tend to perform better than one-time entity ranking methods SetExpander and CaSE, inferring that iteratively refining the feature and representations for the target set and re-ranking the candidate entities can lead to better and stabler expansion results. Among these iterative methods, CGExpan achieves the best performance, especially in terms of MAP@50 score, confirming that class names can serve as better features for the semantic classes and by leveraging them our method can reduce the effect of semantic drift issue during expansion.

We also notice that the two versions of our full model overall have comparable performance, but CGExpan-MRR consistently outperforms CGExpan-Comb. To explain such a difference, empirically we observe that high-quality entities tend to rank high in most of the ranked

Methods	Wiki			APR		
	MAP@10	MAP@20	MAP@50	MAP@10	MAP@20	MAP@50
SetExpan [6]	0.944	0.921	0.720	0.789	0.763	0.639
CGExpan	**0.995**	**0.978**	**0.902**	**0.992**	**0.990**	**0.955**
-NoCN	0.968	0.945	0.859	0.909	0.902	0.787
-NoFilter	0.990	0.975	0.890	0.979	0.962	0.892

Table 4.3: Ablation studies on Wiki and APR for CGExpan

lists and thus favored more by the MRR score. Therefore, we use the MRR version for the rest of our experiments, denoted as CGExpan.

4.2.2 Ablation Study

To study the effectiveness of positive and negative class names individually for our proposed model, we also conduct ablation studies for CGExpan. Specifically, we compare CGExpan and a strong baseline SetExpan with two variants of our model:

- **CGExpan-NoCN**: An ablation of CGExpan that excludes the class name guidance, and thus it only incorporates the average BERT representation to select entities.
- **CGExpan-NoFilter**: An ablation of CGExpan that excludes the negative class name selection step and uses only the single positive class name in the entity selection module.

Table 4.3 shows performance of each model on Wiki and APR corpus. From the table we can see that CGExpan-NoCN outperforms the strong baseline model SetExpan by a large margin, meaning that the pre-trained LM itself is powerful to capture entity similarities. However, it still cannot beat CGExpan-NoFilter model, showing that our model can generate quality class names and our ranking algorithm can further leverage such information to guide the expansion process. Moreover, by comparing our full model with CGExpan-NoFilter, we can see that negative class names indeed help the expansion process by estimating a clear boundary for the target class and filtering out erroneous entities. Such an improvement is particularly obvious on the APR dataset. We think that reason is that, since BERT is pretrained on Wikipedia data, it can already generate high-quality entity similarity estimation with average contextualized embeddings and perform well even without the guidance of class names. However, on APR dataset that BERT has not been pretrained with, the guidance of our predicted class names becomes more important to serve as unambiguous features for the target class and help expand the seed set.

CGExpan vs. Other	MAP@10	MAP@20	MAP@50
vs. SetExpan	100%	94.5%	87.3%
vs. CGExpan-NoFilter	100%	94.5%	58.2%
vs. CGExpan-NoCN	100%	94.5%	70.9%

Table 4.4: Ratio of seed entity set queries on which the first method reaches better or the same performance as the second method.

Methods	Wiki MAP@{10/20/50}	APR MAP@{10/20/50}
Oracle-Full	0.991/0.976/0.891	1.000/1.000/0.964
Oracle-NoFilter	0.994/0.983/0.887	0.988/0.966/0.894
CGExpan	0.995/0.978/0.902	0.992/0.990/0.955

Table 4.5: Compared to oracle models knowing ground truth class names, CGExpan automatically generates class names and achieves comparative performances.

4.2.3 Fine-grained Performance Analysis

Table 4.4 reports more fine-grained comparison results between two methods. Specifically, we calculate the ratio of seed entity set queries (out of total 55 queries) on which our full method achieves better or the same performance as the other method. We can see that CGExpan clearly outperforms SetExpan and its two variants on the majority of queries. Such a result confirms that our proposed method can generate quality class names and use them to guide the expansion process and generate better results.

In Table 4.5, we further compare CGExpan with two "oracle" models that have the access to ground truth class names. Specifically, Oracle-Full is our full model except that the positive class name is replaced by the ground true name provided by the original dataset, while Oracle-NoFilter only uses the ground true class name without predicting and leveraging any negative class name. Results show that CGExpan can achieve comparative results as those oracle models, which indicates the high quality of generated class names by our model and the robustness of CGExpan as the predicted when class names are not perfect.

4.2.4 Parameter Study

In CGExpan, we calculate the similarity between an entity and a class name based on its k occurrences that are most similar to the class name (cf. Eq. (3.1)). Figure 4.1 studies how this parameter k would affect the overall performance on Wiki and APR datasets. We find

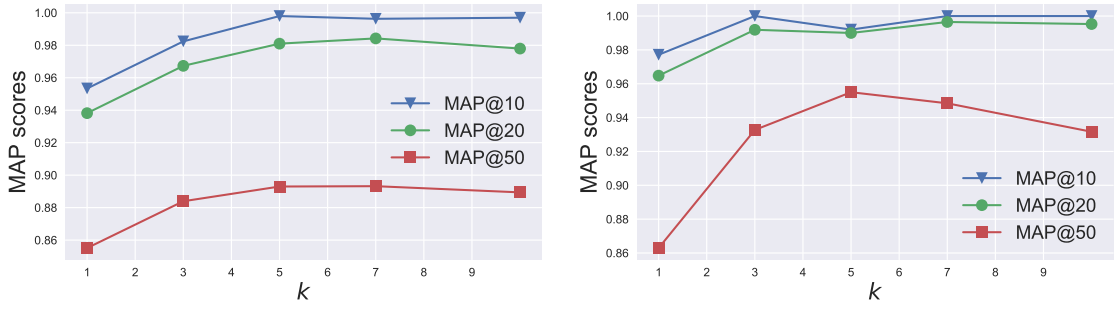

Figure 4.1: Performance for different k values on Wiki (left) and APR (right).

that the model performance first increases when k increases from 1 to 5 and then becomes stable (in terms of MAP@10 and MAP@20) when k further increases to 10. Overall, we find $k = 5$ is enough for calculating entity-class similarity and CGExpan is insensitive to k as long as its value is larger than 5.

4.3 CASE STUDIES

In this section, we provide some case studies for the predicted class names and entity expansion results for our method.

4.3.1 Class Name Selection

Table 4.6 shows some results of our class name ranking module for several queries from different semantic classes in the Wiki dataset. We see that CGExpan is able to select the correct class name and thus injects the correct semantics in later entity selection module. Moreover, as shown in the last column, CGExpan can identify several negative class names that provide a tight boundary for the target semantic class, including `sports` and `competition` for `sports league` class, as well as `city` and `country` for `Chinese province` class. These negative class names help CGExpan avoid adding those related but erroneous entities into the set and thus estimate a better boundary for the target semantic class.

From Table 4.6 we can see that it happens when the predicted positive class name is not exactly the ground true class name in the original dataset. However, since we use both the generated class names and currently expanded entities as guidance and select new entities according to the context features in the provided corpus, those imperfect class names can still guide the set expansion process and perform well empirically. Such a conclusion verifies the results from our previous experiment comparing CGExpan with its variant that uses oracle class names.

Also, in principle, synonyms of the positive class name can be wrongly selected as negative class names, since we only select one class name as the positive (e.g., `country` versus `nation`). Empirically we find that such phenomena will happen but very rarely in our experiments. However, since these synonyms consistently rank lower than the positive one for the initial seeds based on the given corpus, which is why they are treated as negative based on our selection philosophy, they are indeed not good class names for this specific corpus. Thus, misclassifying them will not have much influence on the performance of our model.

4.3.2 Entity Selection

Table 4.7 shows expanded entity sets for two sample queries, namely `tv channel` from Wiki and `political party` from APR. We observe that, if we do not include any information from class names (CGExpan-NoCN), the expansion quality decrease drastically as set becomes larger. If we include the positive class name to guide the expansion (CGExpan-NoFilter), the expansion quality is improved since class names can serve as unambiguous features for the expansion. Finally, since CGExpan can correctly predict the true positive class names and also select relevant negative class names, CGExpan further utilizes negative ones to filter out those related but erroneous entities, including two TV shows in `tv channel` class and three entities in `political party` class. As a result, CGExpan can outperform its ablations and generate quality expansion results consistently.

Seed Entity Set	Ground True Class Name	Positive Class Name	Negative Class Names
{"*Intel*", "*Microsoft*", "*Dell*"}	company	company	product, system, bank, ...
{"*United States*", "*China*", "*Canada*"}	country	country	state, territory, island, ...
{"*ESPNews*", "*ESPN Classic*", "*ABC*"}	tv channel	television network	program, sport, show, ...
{"*NHL*", "*NFL*", "*American league*"}	sports league	professional league	competition, sport, ...
{"*democratic*", "*labor*", "*tories*"}	party	political party	organization, candidate, ...
{"*Hebei*", "*Shandong*", "*Shanxi*"}	Chinese province	chinese province	city, country, state, ...
{"*tuberculossi*", "*Parkinson's disease*", "*esophageal cancer*"}	disease	chronic disease	symptom, condition, ...
{"*Illinois*", "*Arizona*", "*California*"}	US state	state	county, country, ...

Table 4.6: Class names generated for seed entity sets. The 2nd column is the ground true class name in the original dataset. The 3rd and 4th columns are positive and negative class names predicted by CGExpan, respectively.

Seed Entity Set	CGExpan		CGExpan-NoCN		CGExpan-NoFilter	
{"*ESPN*", "*Discovery Channel*", "*Comedy Central*"}	1	"*Pb*"	1	"*NBC*"	1	"*Pb*"
	2	"*ABC*"	2	"*CBS*"	2	"*Mtv*"
	3	"*CBS*"	3	"*Disney Channel*"	3	"*ABC*"
		...		...		...
	35	"*Telemundo*"	35	"*ESPN Radio*"*	35	"*MyNetworkTV*"
	36	"*Fox Sports Net*"	36	"*BBC America*"	36	"*ESPN2*"
	37	"*Dateline NBC*"	37	"*G4*"	37	"*the Today Show*"*
	38	"*Channel 4*"	38	"*Sirius Satellite Radio*"*	38	"*Access Hollywood*"*
	39	"*The History Channel*"	39	"*TNT*"	39	"*Cartoon Network*"
		...		...		...
{"*democratic party*", "*republican party*", "*labor party*"}	1	"*republican*"	1	"*national party*"	1	"*republican*"
	2	"*likud*"	2	"*labour party*"	2	"*likud*"
	3	"*liberal democrats*"	3	"*gop establishment*"*	3	"*liberal democrats*"
		...		...		...
	40	"*komeito*"	40	"*republican jewish coalition*"*	40	"*young voters*"*
	41	"*centrist liberal democrats*"	41	"*british parliament*"*	41	"*bjp*"
	42	"*aipac*"*	42	"*tea party patriots*"*	42	"*religious*"*
	43	"*aam aadmi party*"	43	"*centrist liberal democrats*"	43	"*congress*"*
	44	"*ennahda*"	44	"*federal government*"*	44	"*lib dem*"
		...		...		

Table 4.7: Expanded entity sets for two sample queries, with erroneous entities colored red and marked with a "*".

CHAPTER 5: RELATED WORK

5.1 ENTITY SET EXPANSION

Web-based Entity Set Expansion Earlier works on entity set expansion are developed mostly based on some web search engines. Google Sets [21], one of the earliest entity set expansion methods and now discontinued, calculates entity similarities using latent semantic indexing technique. Later, SEAL [22, 23] is proposed to first submit a query consisting of the seed entities to a search engine, construct a heterogeneous network using the top-ranked pages, and mine entities with a graph-based ranking algorithm. More recently, Lyretail [24] improves the expansion performance on long-tail entities by introducing a supervised page-specific extractor. The above methods require some external search engines for online seed-oriented data collection, which can be costly. In comparison, our method focus on corpus-based entity set expansion without relying on online data collection, and the related works are described below.

Corpus-based Entity Set Expansion More recent studies propose to expand the seed entity set by offline processing a corpus. Compared to previous web search based methods, this line of works do not require an online search engine for data collection and thus can be more efficient. These corpus-based methods can be categorized into two general approaches: one-time entity ranking, and iterative bootstrapping.

One-time entity ranking methods entity distributional similarities and ranks all entities once without back and forth refinement. Pantel et al. [25] develop an entity set expansion pipeline by utilizing a web-scale term similarity matrix. SetExpander [18] trains multiple term embeddings and build a classifier with some human-annotated sets. CaSE [19] also exploit syntactic features together with term embeddings for a one-time entity ranking algorithm. Recently, a two-stage entity set expansion method is proposed [26], where a pre-trained language model is used to select representative sentences as features for each seed set.

Iterative bootstrapping method aims to bootstrap the seed entity set by iteratively selecting context features and ranking new entities. Egoset [17] is a multifaceted set expansion system using context features and term embeddings and mines entities from a word ego-network. SetExpan [6] develops selects quality skip-grams as context features and uses a rank ensemble module to rank entities robustly. MCTS [14] combines the Montre Carlo Tree Search algorithm with a deep similarity network to select quality patterns and bootstrap the

seed set. FUSE [27] clusters skip-gram features to detect multiple facets and match the coherent facets for the seed set to expand it. SetCoExpan [15] generates auxiliary sets with closely related but wrong entities for the target set to guide the expansion process. Our method in general belongs to this line of work.

Finally, there are some studies that incorporate extra knowledge to expand the entity set, including negative examples [7, 8, 28], semi-structured web table [29], and external knowledge base [17]. Particularly, [29] also propose to use a class name to help expand the target set. However, their method requires a *user-provided* class name and utilizes web tables as additional knowledge, while our method can automatically generate both positive and negative class names and utilize them to guide the set expansion process.

5.2 PRETRAINED LANGUAGE MODEL

Traditional language models aim at assigning a probability for an input word sequence. Recent studies have shown that by pre-training some deep models with massive text corpora, language models are able to generate contextualized word representations and achieve competitive performance on many NLP tasks after simply task-specific fine-tuning. ELMo [9] proposes to learn a multiple-layer BiLSTM model that captures both forward and backward contexts. GPT [30, 31] first uses Transformer decoder as backbone model and pre-trains it with standard auto-regressive language modeling task. BERT [10] leverages the Transformer architecture and learns to predict randomly masked tokens in the input word sequence (masked language model) and to classify the neighboring relation between pair of input sentences (next sentence prediction). Based on BERT's philosophy, RoBERTa [32] conducts more careful hyperparameter tuning and proposes to modify the next sentence prediction task to improve the performance on downstream tasks. ALBERT [33], another important variant of BERT, improves the space efficiency by factorizing word embedding matrix and sharing parameters between Transformer layers. XLNet [11] further combines the ideas from ELMo and BERT and develops an autoregressive model that learns contextualized representation by maximizing the expected likelihood over permutations of the input sequence. ELECTRA [34] pre-trains a generator-discriminator network with replaced token detection task, which learns from all input tokens and thus more computationally efficient.

Pretrained language models have also been studied on the entity set expansion task. SetCoExpan [15] first uses average BERT embeddings for each entity to substitute the Word2vec embeddings [35] used in previous works. Such embedding features are combined with context features for the expansion process. [26] base their set expansion algorithm on the contextualized embeddings generated by a pre-trained BERT model to select representative

sentences for the seed set, which are used to rank new entities with similar context from the corpus. Compared with the above methods that utilize the contextualized embeddings, our CGExpan also uses pre-trained language model as a knowledge base and explicitly predicts positive and negative class names for the seed set to guide the expansion.

5.3 LANGUAGE MODEL PROBING

Aside from generating contextualized representations, pre-trained language models can also serve as knowledge bases when being queried appropriately. [36] introduces the language model analysis probe (LAMA) and *manually define* probing queries for each relation type. By submitting those probing queries to a pre-trained LM, they show that they can retrieve relational knowledge and achieve competitive performance on various NLP tasks. Later, several methods are proposed to improve from LAMA: [37] automatically discovers better prompts, [38] adds negation and mispriming to the probing task, [39] removes overly easy questions and proposes E-BERT for unsupervised QA, [40] analyzes BERT's ability to store relational knowledge by automatically selecting high-quality templates, [41] and [42] studies the ability of pre-trained language model on textual reasoning task, and [43] provides extra context documents to improve the factual unsupervised QA.

Comparing with previous work, in this paper, we show that probing pre-trained language model works for entity set expansion task, and we propose a new entity set expansion framework that combines corpus-independent LM probing with corpus-specific context information for better expansion performance. Also, dislike previous work where prompts are manually defined or mined from corpus, we simply utilize some lexico-syntactic patterns from hypernym detection task to automatically construct our probing queries, and experiment results show that our proposed method can generate high-quality class names and boosts the performance of entity set expansion task.

5.4 CATEGORY GENERATION

Our class name generation and selection modules are also closely related to a recently proposed new task, Category Generation [44]. The task is to generate a ranked list of category names given a set of entities and the new category will be attached to a node in an existing category taxonomy. Their method require some abstractive summarization models to generate candidate class names and rank the candidates in two-phase while considering their position in the hierarchy.

Compared with category generation task, our class name prediction module, serving as an auxiliary task for our entity set expansion framework, does not rely on any existing taxonomy structure and focuses more on coarse types due to the property of current ESE datasets. As a result, we only need to query a pre-trained LM to generate high-quality class names without any training, and rank them using some corpus-based similarity measure. Although we target slightly different outputs, both coarse and fine-grained class names predicted for entity sets can be useful for many downstream tasks, including named entity typing and taxonomy construction.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

In this paper, we propose a new entity set expansion framework that can use a pre-trained LM to generate candidate class names for the seed set, rank them according to the provided text corpus, and guide the entity selection process with the selected class names. With the help of high-quality class names and a new ranking philosophy, our method can alleviate the semantic drift problem caused by ambiguous context features and the overly reliance on the already expanded set in previous studies. Extensive experiments on the Wiki and APR datasets demonstrate the effectiveness of our framework on both class name prediction and entity set expansion.

In the future, we plan to expand the method scope from expanding concrete entity sets to more abstract concept sets. For example, we may expand the set {"*machine translation*", "*information extraction*", "*syntactic parsing*"} to acquire more `NLP task` concepts. Such a task is more challenging than entity set expansion, as concept terms often appear in more diverse context. Another interesting direction is taxonomy construction, for which we can investigate the potential of language model probing to generate a class name hierarchy, which can benefit different downstream tasks.

REFERENCES

[1] R. C. Wang, N. Schlaefer, W. W. Cohen, and E. Nyberg, "Automatic set expansion for list question answering," in *EMNLP*, 2008.

[2] W. Hua, Z. Wang, H. Wang, K. Zheng, and X. Zhou, "Understand short texts by harvesting and analyzing semantic knowledge," in *TKDE*, 2017.

[3] 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.

[4] C. Xiong, R. Power, and J. P. Callan, "Explicit semantic ranking for academic search via knowledge graph embedding," in *WWW*, 2017.

[5] J. Shen, J. Xiao, X. He, J. Shang, S. Sinha, and J. Han, "Entity set search of scientific literature: An unsupervised ranking approach," in *SIGIR*, 2018.

[6] 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.

[7] J. R. Curran, T. Murphy, and B. Scholz, "Minimising semantic drift with mutual exclusion bootstrapping," in *PAACL*, 2007.

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

[9] 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.

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

[11] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *NeurIPS*, 2019.

[12] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *COLING*, 1992.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[14] L. Yan, X. Han, L. Sun, and B. He, "Learning to bootstrap for entity set expansion," in *EMNLP*, 2019.

[15] 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 *WebConf*, 2020.

[16] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han, "Automated phrase mining from massive text corpora," *TKDE*, 2018.

[17] 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.

[18] 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.

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

[20] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," *ArXiv*, vol. abs/1910.03771, 2019.

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

[22] R. C. Wang and W. W. Cohen, "Language-independent set expansion of named entities using the web," in *ICDM*, 2007.

[23] R. C. Wang and W. W. Cohen, "Iterative set expansion of named entities using the web," in *ICDM*, 2008.

[24] Z. Chen, M. Cafarella, and H. Jagadish, "Long-tail vocabulary dictionary extraction from the web," in *WSDM*, 2016.

[25] P. Pantel, E. Crestan, A. Borkovsky, A.-M. Popescu, and V. Vyas, "Web-scale distributional similarity and entity set expansion," in *EMNLP*, 2009.

[26] G. Kushilevitz, S. Markovitch, and Y. Goldberg, "A two-stage masked LM method for term set expansion," in *ACL*, 2020.

[27] W. Zhu, H. Gong, J. Shen, C. Zhang, J. Shang, S. Bhat, and J. Han, "Fuse: Multi-faceted set expansion by coherent clustering of skip-grams," in *ECML-PKDD*, 2020.

[28] T. McIntosh and J. R. Curran, "Weighted mutual exclusion bootstrapping for domain independent lexicon and template acquisition," in *ALTA*, 2008.

[29] C. Wang, K. Chakrabarti, Y. He, K. Ganjam, Z. Chen, and P. A. Bernstein, "Concept expansion using web tables," in *WWW*, 2015.

[30] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018.

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

[32] 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*, vol. abs/1907.11692, 2019.

[33] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *ICLR*, 2019.

[34] 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.

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

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

[37] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?" *TACL*, 2020.

[38] N. Kassner and H. Schütze, "Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly," in *ACL*, 2020.

[39] N. Poerner, U. Waltinger, and H. Schütze, "E-bert: Efficient-yet-effective entity embeddings for bert," in *EMNLP*, 2020.

[40] Z. Bouraoui, J. Camacho-Collados, and S. Schockaert, "Inducing relational knowledge from bert," in *AAAI*, 2020.

[41] A. Talmor, Y. Elazar, Y. Goldberg, and J. Berant, "olmpics-on what language model pre-training captures," *TACL*, 2020.

[42] Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi, "Piqa: Reasoning about physical commonsense in natural language," in *AAAI*, 2020.

[43] F. Petroni, P. Lewis, A. Piktus, T. Rocktäschel, Y. Wu, A. H. Miller, and S. Riedel, "How context affects language models' factual predictions," in *AKBC*, 2020.

[44] S. Zhang, K. Balog, and J. Callan, "Generating categories for sets of entities," *CIKM*, 2020.