Empower Entity Set Expansion via Language Model Probing

Yunyi Zhang[1], **Jiaming Shen**[1], **Jingbo Shang**[2], **Jiawei Han**[1]
[1] University of Illinois at Urbana-Champaign, IL, USA
[2] University of California San Diego, CA, USA
[1]{yzhan238, js2, hanj}@illinois.edu [2] jshang@ucsd.edu

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.

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 (Wang et al., 2008), query understanding (Hua et al., 2017), taxonomy construction (Shen et al., 2018a), and semantic search (Xiong et al., 2017; Shen et al., 2018b).

Most existing entity set expansion methods bootstrap the initial seed set by iteratively selecting context features (e.g., co-occurrence words (Pantel et al., 2009), unary patterns (Rong et al., 2016), and coordinational patterns (Mamou et al., 2018)), while extracting and ranking new entities. A key challenge to set expansion 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). 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. Addressing such "semantic drift" issue without requiring additional user inputs (e.g., mutually exclusive classes (Curran et al., 2007) and negative example entities (Jindal and Roth, 2011)) remains an open research problem.

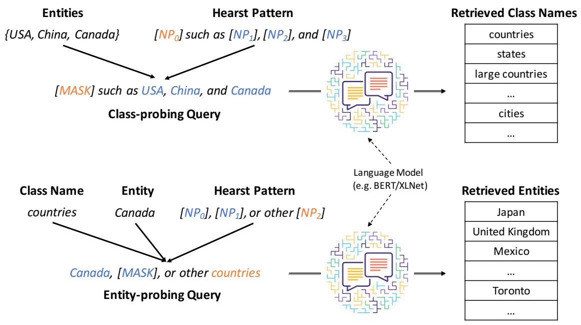


Figure 1: Examples of class-probing and entity-probing queries generated based on Hearst patterns.

In this study, we propose to empower entity set expansion with class names automatically generated from pre-trained language models (Peters et al., 2018; Devlin et al., 2019; Yang et al., 2019). 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

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8151–8160
July 5 - 10, 2020. ©2020 Association for Computational Linguistics

model automatically without relying on human annotated data.

Motivated by the above intuition, we propose 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) 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 (Shen et al., 2017) 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; and (3) experiments on two public datasets from different domains demonstrate the superior performance of our approach compared with state-of-the-art methods.

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 $\mathbf{w} = [w_1, w_2, \dots, w_n]$ and assigns a probability $\mathbf{P}(\mathbf{w})$ to the whole sequence. Recent studies (Peters et al., 2018; Devlin et al., 2019; Yang et al., 2019) 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 $\mathbf{P}(w_i) = \mathbf{P}(w_i | w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_n)$ to any position $1 \leq i \leq n$. Currently, Devlin et al. (2019) propose 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 (Vaswani et al., 2017) architecture and is learned on English Wikipedia as well as BookCorpus. More LM architectures are described in Section 5.

2.2 Problem Formulation

We first define some key concepts and then present our problem formulation.

Entity. An entity is a word or a phrase that refers to a real-world instance. For example, "*U.S.*" refers to the country: United States.

Class Name. A class name 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*".

Probing Query. A probing query is a word sequence containing one [MASK] token. In this work, we utilize Hearst patterns (Hearst, 1992) to construct two types of probing queries: (1) A *class-probing* query aims to predict the class name of some given entities (e.g., "[MASK] such as United States and China"), and (2) an *entity-probing* query aims to retrieve entities that fit into the mask token (e.g., "countries such as [MASK] and Japan").

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

Example 1. Given a seed set of three countries {"*United States*", "*China*", "*Canada*"}, we aim to return a ranked list of entities belonging to the same `country` class such as "*United Kingdom*", "*Japan*", and "*Mexico*".

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 (Hearst, 1992)[1], widely used for hypernymy detection, to construct the class-probing query. 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 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 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[2]. 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.

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[3], and 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:

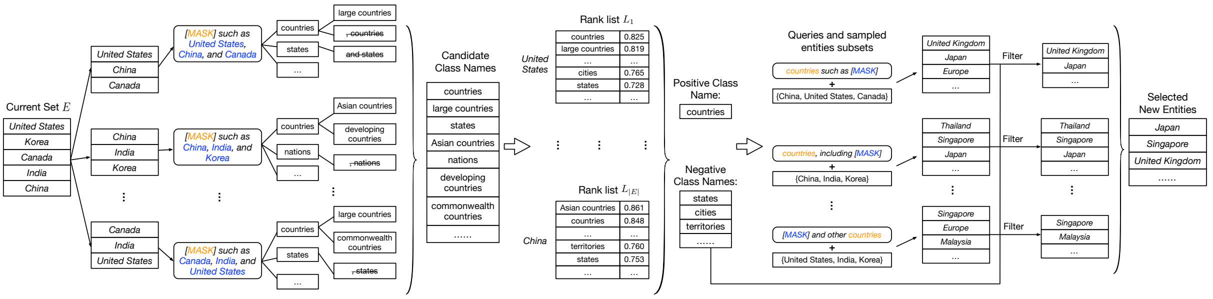
$$M^k(e,c) = \frac{1}{k} \max_{X \subseteq X_e, |X|=k} \sum_{\mathbf{x} \in X} \max_{\mathbf{x}' \in X_c} \mathbf{cos}(\mathbf{x}, \mathbf{x}'), \quad (1)$$

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

[1] For example, the pattern "NP_y such as NP_a" indicates that noun phrase y is a hypernym of noun phrase a.

[2] Therefore, class names likes "and countries" and ", countries" are filtered out.

[3] For example, a query for class name "countries" is "countries such as [MASK]".



(1) Class Name Generation Module   *(2) Class Name Ranking Module*   *(3) Class-Guide Entity Selection Module*

Figure 2: Overview of *one iteration* in CGExpan framework.

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 (Shen et al., 2017) 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.

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 :

$$\text{score}_i^{loc} = M^k(e_i, c_p), \tag{2}$$

where M^k is defined in Eq. (1). We refer to this score as a <u>loc</u>al 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_q.

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:

$$\text{score}_i^{glb} = \frac{1}{|E_s|} \sum_{e \in E_s} \mathbf{cos}(\mathbf{v}_{e_i}, \mathbf{v}_e). \tag{3}$$

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 <u>glob</u>al 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:

$$\text{score}_i = \sqrt{\text{score}_i^{loc} \times \text{score}_i^{glb}}. \tag{4}$$

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)) 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:

$$S(e_i) = \sum_{t=1}^{T} \left(\mathbb{1}(e_i \in E) + s^t(e_i) \right) \times \mathbb{1}(r^i_{c_p} < \min_{c' \in C_N} r^i_{c'}), \tag{5}$$

where $\mathbb{1}(\cdot)$ is an indicator function, r^i_c is the rank of class name c in entity e_i's ranked list L^i_c, 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

$$s^t(e_i) = \frac{1}{r^t_i} \tag{6}$$

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

$$s^t(e_i) = \frac{\text{score}^t_i - \min_{e_j \in R^t} \text{score}^t_j}{\max_{e_j \in R^t} \text{score}^t_j - \min_{e_j \in R^t} \text{score}^t_j} \tag{7}$$

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

To interpret Eq. 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.

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

Table 1: Datasets statistics

Summary. 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.

4 Experiments

4.1 Experiment Setup

Datasets. We conduct our experiments on two public benchmark datasets widely used in previous studies (Shen et al., 2017; Yan et al., 2019): (1) **Wiki**, which is a subset of English Wikipedia articles, and (2) **APR**, which contains all news articles published by Associated Press and Reuters in 2015. Following the previous work, we adopt a phrase mining tool, AutoPhrase (Shang et al., 2018), to construct the entity vocabulary list from the corpus, and select 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 and each seed set contains 3 entities. Table 1 summarizes the statistics for these datasets.

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

1. Egoset (Rong et al., 2016): This is a multifaceted set expansion system using context features and Word2Vec embeddings. The original

Methods	**Wiki**			**APR**		
	MAP@10	MAP@20	MAP@50	MAP@10	MAP@20	MAP@50
Egoset (Rong et al., 2016)	0.904	0.877	0.745	0.758	0.710	0.570
SetExpan (Shen et al., 2017)	0.944	0.921	0.720	0.789	0.763	0.639
SetExpander (Mamou et al., 2018)	0.499	0.439	0.321	0.287	0.208	0.120
CaSE (Yu et al., 2019b)	0.897	0.806	0.588	0.619	0.494	0.330
MCTS (Yan et al., 2019)	0.980$^{\nabla}$	0.930$^{\nabla}$	0.790$^{\nabla}$	0.960$^{\nabla}$	0.900$^{\nabla}$	0.810$^{\nabla}$
CGExpan-NoCN	0.968	0.945	0.859	0.909	0.902	0.787
CGExpan-NoFilter	0.990	0.975	0.890	0.979	0.962	0.892
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 2: Mean Average Precision on Wiki and APR. "$^{\nabla}$" means the number is directly from the original paper.

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 (Shen et al., 2017): This method iteratively selects skip-gram context features from the corpus and develops a rank ensemble mechanism to score and select entities.
3. SetExpander (Mamou et al., 2018): 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.
4. CaSE (Yu et al., 2019b): 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.
5. MCTS (Yan et al., 2019): 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.
6. CGExpan: This method is our proposed Class-Guided Set Expansion framework, using BERT (Devlin et al., 2019) 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.
7. CGExpan-NoCN: An ablation of CGExpan that excludes the class name guidance. Therefore, it only incorporates the average BERT representation to select entities.
8. 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.

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

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

$$\text{MAP@}K = \frac{1}{|Q|}\sum_{q \in Q} \text{AP}_K(L_q, S_q),$$

where Q is the set of all seed queries and for each query q, we use $\text{AP}_K(L_q, S_q)$ to denote the traditional average precision at position K given a ranked list of entities L_q and a ground-truth set S_q.

Implementation Details. For CGExpan, we use BERT-base-uncased[4] as our pre-trained LM. For parameter setting, in the class name generation module (Sec. 3.1), we take top-3 predicted tokens in each level of beam search and set the maximum length of generated class names up to 3. When calculating the similarity between an entity and a class name (Eq. 1), we choose $k = 5$, and will later provide a parameter study on k in the experiment. 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.[5]

[4] In principle, other masked LMs such as RoBERTa and XLNet can also be used in our framework.

[5] The code and data are available at https://github.com/yzhan238/CGExpan

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

4.2 Experiment Results

Overall Performance. Table 2 shows the overall performance of different entity set expansion methods. We can see that CGExpan along with its ablations in general outperform all the baselines by a large margin. Comparing with SetExpan, the full model CGExpan 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. In addition, CGExpan-NoCN outperforms most baseline models, meaning that the pre-trained LM itself is powerful to capture entity similarities. However, it still cannot beat CGExpan-NoFilter model, which shows that we can properly guide the set expansion process by incorporating generated class names. 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. 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 lists. Therefore, we use the MRR version for the rest of our experiment, denoted as CGExpan.

Fine-grained Performance Analysis. Table 3 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 one 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. In Table 4, we further compare CGExpan with two "oracle" models that have the access to ground truth class names. Results show that CGExpan can achieve comparative results as those oracle models, which indicates the high quality of generated class names and effectiveness of CGExpan.

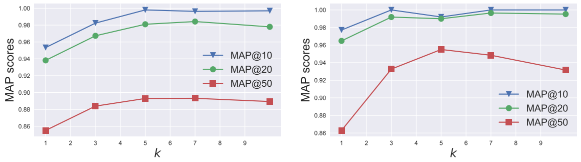


Figure 3: Performance for different k values on Wiki (left) and APR (right).

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. (1)). Figure 3 studies how this parameter k would affect the overall performance. We find 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

Class Name Selection. Table 5 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 `sport 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.

From Table 5 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.

Also, in principle, synonyms of the positive class name can be wrongly selected as negative class names, which also happens but very rarely in our experiments. However, since these synonyms con-

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`	`sport`, `competition`, ...
{“*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 5: 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.

sistently rank lower than the positive one for the initial seeds based on the given corpus, 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.

Entity Selection. Table 6 shows expanded entity sets for two sample queries. After correctly predicting true positive class names and selecting relevant negative class names, CGExpan utilizes them to filter out those related but erroneous entities, including two TV shows in `television network` class and three entities in `political party` class. As a result, CGExpan can outperform CGExpan-NoFilter.

5 Related Work

Entity Set Expansion. Traditional entity set expansion systems such as Google Sets (Tong and Dean, 2008) and SEAL (Wang and Cohen, 2007, 2008) typically submit a query consisting of seed entities to a general-domain search engine and extract new entities from retrieved web pages. These methods require an external search engine for online seed-oriented data collection, which can be costly. Therefore, more recent studies propose to expand the seed set by offline processing a corpus. These corpus-based set expansion methods can be categorized into two general approaches: (1) *one-time entity ranking* which calculates entity distributional similarities and ranks all entities once without back and forth refinement (Mamou et al., 2018; Yu et al., 2019b), and (2) *iterative bootstrapping* which aims to bootstrap the seed entity set by iteratively selecting context features and ranking new entities (Rong et al., 2016; Shen et al., 2017; Yan et al., 2019; Zhu et al., 2019; Huang et al., 2020). Our method in general belongs to the later category. Finally, there are some studies that incorporate extra knowledge to expand the entity set, including negative examples (Curran et al., 2007; McIntosh and Curran, 2008; Jindal and Roth, 2011), semi-structured web table (Wang et al., 2015), and external knowledge base (Yu et al., 2019a). Particularly, Wang et al. (2015) 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.

Language Model Probing. Traditional language models aim at assigning a probability for an input word sequence. Recent studies have shown that by training on next word or missing word prediction task, language models are able to generate contextualized word representations that benefit many downstream applications. ELMo (Peters et al., 2018) proposes to learn a BiLSTM model that captures both forward and backward contexts. BERT (Devlin et al., 2019) leverages the Transformer architecture and learns to predict randomly masked tokens in the input word sequence and to classify the neighboring relation between pair of input sentences. Based on BERT’s philosophy, RoBERTa (Liu et al., 2019) conducts more careful hyper-parameter tuning to improve the performance on downstream tasks. XLNet (Yang et al., 2019) 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.

Aside from generating contextualized representations, pre-trained language models can also serve as knowledge bases when being queried appropriately. Petroni et al. (2019) introduce the language model analysis probe and *manually define* probing queries for each relation type. By submitting those probing queries to a pre-trained LM, they show that we can retrieve relational knowledge and achieve competitive performance on various NLP tasks. More recently, Bouraoui et al. (2020) further analyze BERT’s ability to store relational knowledge by using BERT to automatically select high-

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 6: Expanded entity sets for two sample queries, with erroneous entities colored red and marked with a “*”.

quality templates from text corpus for new relation prediction. 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.

6 Conclusions

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. 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. Another interesting direction is to generate a class name hierarchy via language model probing.

Acknowledgments

Research was sponsored in part by US DARPA KAIROS Program No. FA8750-19-2-1004 and SocialSim Program No. W911NF-17-C-0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, and DTRA HDTRA11810026. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and should not be interpreted as necessarily representing the views, either expressed or implied, of DARPA or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright annotation hereon. We thank anonymous reviewers for valuable feedback.

References

Zied Bouraoui, Jose Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge from bert. In *AAAI*.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *PAACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*.

Wen Hua, Zhongyuan Wang, Haixun Wang, Kai Zheng, and Xiaofang Zhou. 2017. Understand short texts by harvesting and analyzing semantic knowledge. In *TKDE*.

Jiaxin Huang, Yiqing Xie, Yu Meng, Jiaming Shen, Yunyi Zhang, and Jiawei Han. 2020. Guiding corpus-based set expansion by auxiliary sets generation and co-expansion. In *WebConf*.

Prateek Jindal and Dan Roth. 2011. Learning from negative examples in set-expansion. In *ICDM*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Jonathan Mamou, Oren Pereg, Moshe Wasserblat, Alon Eirew, Yael Green, Shira Guskin, Peter Izsak, and Daniel Korat. 2018. Term set expansion based nlp architect by intel ai lab. In *EMNLP*.

Tara McIntosh and James R. Curran. 2008. Weighted mutual exclusion bootstrapping for domain independent lexicon and template acquisition. In *ALTA*.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *EMNLP*.

Xin Rong, Zhe Chen, Qiaozhu Mei, and Eytan Adar. 2016. Egoset: Exploiting word ego-networks and user-generated ontology for multifaceted set expansion. In *WSDM*.

Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R. Voss, and Jiawei Han. 2018. Automated phrase mining from massive text corpora. *TKDE*.

Jiaming Shen, Zeqiu Wu, Dongming Lei, Jingbo Shang, Xiang Ren, and Jiawei Han. 2017. Setexpan: Corpus-based set expansion via context feature selection and rank ensemble. In *ECML/PKDD*.

Jiaming Shen, Zeqiu Wu, Dongming Lei, Chao Zhang, Xiang Ren, Michelle T. Vanni, Brian M. Sadler, and Jiawei Han. 2018a. Hiexpan: Task-guided taxonomy construction by hierarchical tree expansion. In *KDD*.

Jiaming Shen, Jinfeng Xiao, Xinwei He, Jingbo Shang, Saurabh Sinha, and Jiawei Han. 2018b. Entity set search of scientific literature: An unsupervised ranking approach. In *SIGIR*.

Simon Tong and Jeff Dean. 2008. System and methods for automatically creating lists. US Patent 7,350,187.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Chi Wang, Kaushik Chakrabarti, Yeye He, Kris Ganjam, Zhimin Chen, and Philip A. Bernstein. 2015. Concept expansion using web tables. In *WWW*.

Richard C. Wang and William W. Cohen. 2007. Language-independent set expansion of named entities using the web. In *ICDM*.

Richard C. Wang and William W. Cohen. 2008. Iterative set expansion of named entities using the web. In *ICDM*.

Richard C. Wang, Nico Schlaefer, William W. Cohen, and Eric Nyberg. 2008. Automatic set expansion for list question answering. In *EMNLP*.

Chenyan Xiong, Russell Power, and James P. Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *WWW*.

Lingyong Yan, Xianpei Han, Le Sun, and Ben He. 2019. Learning to bootstrap for entity set expansion. In *EMNLP*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurlPS*.

Jifan Yu, Chenyu Wang, Gan Luo, Lei Hou, Juan-Zi Li, Zhiyuan Liu, and Jie Tang. 2019a. Course concept expansion in moocs with external knowledge and interactive game. In *ACL*.

Puxuan Yu, Zhiqi Huang, Razieh Rahimi, and James D Allan. 2019b. Corpus-based set expansion with lexical features and distributed representations. In *SIGIR*.

Wanzheng Zhu, Hongyu Gong, Jiaming Shen, Chao Zhang, Jingbo Shang, Suma Bhat, and Jiawei Han. 2019. Fuse: Multi-faceted set expansion by coherent clustering of skip-grams. *ArXiv*, abs/1910.04345.