SynSetExpan: An Iterative Framework for Joint Entity Set Expansion and Synonym Discovery

Jiaming Shen[1⋆], **Wenda Qiu**[1⋆], **Jingbo Shang**[2], **Michelle Vanni**[3], **Xiang Ren**[4], **Jiawei Han**[1]

[1]University of Illinois Urbana-Champaign, IL, USA, [2]University of California San Diego, CA, USA

[3]U.S. Army Research Laboratory, MD, USA, [4]University of Southern California, CA, USA

[1]{js2, qiuwenda, hanj}@illinois.edu [2]jshang@ucsd.edu [4]michelle.t.vanni.civ@mail.mil [4]xiangren@usc.edu

Abstract

Entity set expansion and synonym discovery are two critical NLP tasks. Previous studies accomplish them separately, without exploring their interdependences. In this work, we hypothesize that these two tasks are tightly coupled because *two synonymous entities tend to have similar likelihoods of belonging to various semantic classes*. This motivates us to design SynSetExpan, a novel framework that enables two tasks to mutually enhance each other. SynSetExpan uses a synonym discovery model to include popular entities' infrequent synonyms into the set, which boosts the set expansion recall. Meanwhile, the set expansion model, being able to determine whether an entity belongs to a semantic class, can generate pseudo training data to fine-tune the synonym discovery model towards better accuracy. To facilitate the research on studying the interplays of these two tasks, we create the first large-scale Synonym-Enhanced Set Expansion (SE2) dataset via crowdsourcing. Extensive experiments on the SE2 dataset and previous benchmarks demonstrate the effectiveness of SynSetExpan for both entity set expansion and synonym discovery tasks.

1 Introduction

Entity set expansion (ESE) aims to expand a small set of seed entities (e.g., {"*United States*", "*Canada*"}) into a larger set of entities that belong to the same semantic class (*i.e.*, `Country`). Entity synonym discovery (ESD) intends to group all terms in a vocabulary that refer to the same real-world entity (e.g., "*America*" and "*USA*" refer to the same country) into a synonym set (hence called a *synset*). Those discovered entities and synsets include rich knowledge and can benefit many downstream applications such as semantic search (Xiong et al., 2017), taxonomy construction (Shen et al., 2018a), and online education (Yu et al., 2019a).

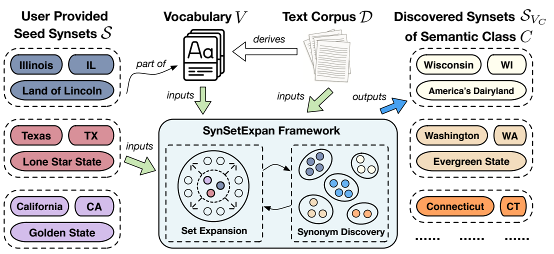


Figure 1: An illustrative example of joint entity set expansion and synonym discovery.

Previous studies regard ESE and ESD as two independent tasks. Many ESE methods (Mamou et al., 2018b; Yan et al., 2019; Huang et al., 2020; Zhang et al., 2020; Zhu et al., 2020) are developed to iteratively select and add the most confident entities into the set. A core challenge for ESE is to find those infrequent long-tail entities in the target semantic class (e.g., "*Lone Star State*" in the class `US_States`) while filtering out false positive entities from other related classes (e.g., "*Austin*" and "*Dallas*" in the class `City`) as they will cause semantic shift to the set. Meanwhile, various ESD methods (Qu et al., 2017; Ustalov et al., 2017a; Wang et al., 2019; Shen et al., 2019) combine string-level features with embedding features to find a query term's synonyms from a given vocabulary or to cluster all vocabulary terms into synsets. A major challenge here is to combine those features with limited supervisions in a way that works for entities from all semantic classes. Another challenge is how to scale a ESD method to a large, extensive vocabulary that contains terms of varied qualities.

To address the above challenges, we hypothesize that ESE and ESD are two tightly coupled tasks and can mutually enhance each other because *two synonymous entities tend to have similar likelihoods of belonging to various semantic classes and vice versa*. This hypothesis implies that (1)

[*]Equal Contributions.

knowing the class membership of one entity enables us to infer the class membership of all its synonyms, and (2) two entities can be synonyms only if they belong to the same semantic class. For example, we may expand the `US_States` class from a seed set {"*Illinois*", "*Texas*", "*California*"}. An ESE model can find frequent state full names (e.g., "*Wisconsin*", "*Connecticut*") but may miss those infrequent entities (e.g., "*Lone Star State*" and "*Golden State*"). However, an ESD model may predict "*Lone Star State*" is the synonym of "*Texas*" and "*Golden State*" is synonymous to "*California*" and directly adds them into the expanded set, which shows synonym information help set expansion. Meanwhile, from the ESE model outputs, we may infer ⟨"*Wisconsin*", "*WI*"⟩ is a synonymous pair while ⟨"*Connecticut*", "*SC*"⟩ is not, and use them to fine-tune an ESD model on the fly. This relieves the burden of using one single ESD model for all semantic classes and improves the ESD model's inference efficiency because we refine the synonym search space from the entire vocabulary to only the ESE model outputs.

In this study, we propose SynSetExpan, a novel framework jointly conducting two tasks (cf. Fig. 1). To better leverage the limited supervision signals in seeds, we design SynSetExpan as an iterative framework consisting of two components: (1) *a ESE model* that ranks entities based on their probabilities of belonging to the target semantic class, and (2) *a ESD model* that returns the probability of two entities being synonyms. In each iteration, we first apply the ESE model to obtain an entity rank list from which we derive a set of *pseudo training* data to fine-tune the ESD model. Then, we use this fine-tuned model to find synonyms of entities in the currently expanded set and adjust the above rank list. Finally, we add top-ranked entities in the adjusted rank list into the currently expanded set and start the next iteration. After the iterative process ends, we construct a synonym graph from the last iteration's output and extract entity synsets (including singleton synsets) as graph communities.

As previous ESE datasets are too small and contain no synonym information for evaluating our hypothesis, we create the first Synonym Enhanced Set Expansion (SE2) benchmark dataset via crowdsourcing. This new dataset[1] is one magnitude larger than previous benchmarks. It contains a corpus of the entire Wikipedia, a vocabulary of 1.5 million terms, and 1200 seed queries from 60 semantic classes of 6 different types (e.g., Person, Location, Organization, *etc.*).

Contributions. In summary, this study makes the following contributions: (1) we hypothesize that ESE and ESD can mutually enhance each other and propose a novel framework SynSetExpan to jointly conduct two tasks; (2) we construct a new large-scale dataset SE2 that supports fair comparison across different methods and facilitates future research on both tasks; and (3) we conduct extensive experiments to verify our hypothesis and show the effectiveness of SynSetExpan on both tasks.

2 Problem Formulation

We first introduce important concepts in this work, and then present our problem formulation. A **term** is a string (*i.e.*, a word or a phrase) that refers to a real-world entity[2]. An **entity synset** is a set of terms that can be used to refer to the same real-world entity. For example, both "USA" and "America" can refer to the entity *United States* and thus compose an entity synset. We allow the singleton synset and a term may locate in multiple synsets due to its ambiguity. A **semantic class** is a set of entities that share a common characteristic and a **vocabulary** is a term list that can be either provided by users or derived from a corpus.

Problem Formulation. Given (1) a text corpus $\mathcal{D}$, (2) a vocabulary $\mathcal{V}$ derived from $\mathcal{D}$, and (3) a seed set of user-provided entity synonym sets $\mathcal{S}_0$ that belong to the same semantic class C, we aim to (1) select a subset of entities $\mathcal{V}_C$ from $\mathcal{V}$ that all belong to C; and (2) clusters all terms in $\mathcal{V}_C$ into entity synsets $\mathcal{S}_{\mathcal{V}_C}$ where the union of all clusters is equal to $\mathcal{V}_C$. In other words, we expand the seed set $\mathcal{S}_0$ into a more complete set of entity synsets $\mathcal{S}_0 \cup \mathcal{S}_{\mathcal{V}_C}$ that belong to the same semantic class C. A concrete example is presented in Fig. 1.

3 The SynSetExpan Framework

In this study, we hypothesize that entity set expansion and synonym discovery are two tightly coupled tasks and can mutually enhance each other.

Hypothesis 1. *Two synonymous entities tend to have similar likelihoods of belonging to various semantic classes and vice versa.*

The above hypothesis has two implications. First, if two entities e_i and e_j are synonyms and

[1]`http://bit.ly/SE2-dataset`.

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

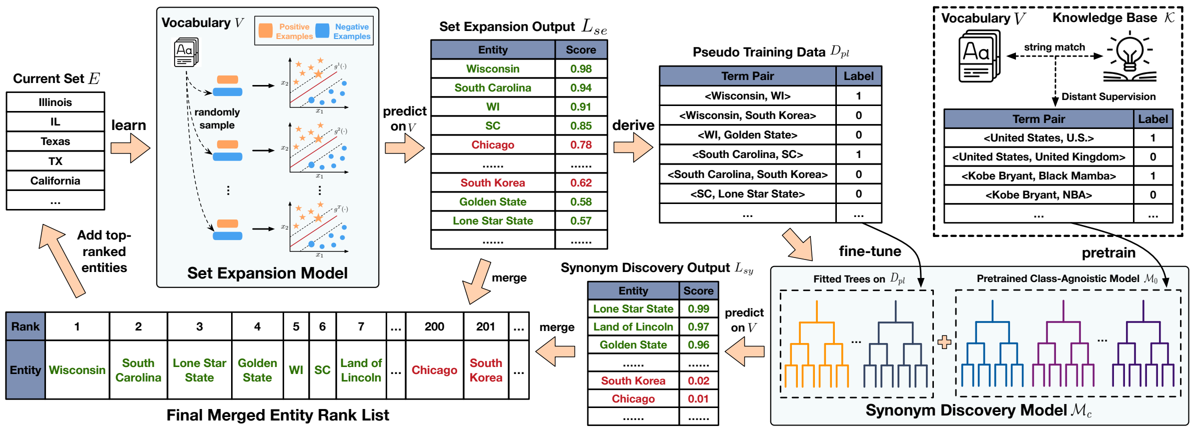


Figure 2: Overview of one iteration in our proposed SynSetExpan framework. Starting from the current set E, we first run a set expansion model to obtain an entity rank list L_{se} based on which we generate pseudo training data D_{pl} to fine-tune a generic synonym discovery model $\mathcal{M}_0$. We then apply this fine-tuned model to get a new rank list L_{sy}; merge it with L_{se} to obtain the final entity rank list, and add top ranked entities into the current set E.

e_i belongs to semantic class C, e_j likely also belongs to class C even if it is currently outside C. This reveals how synonym information can help set expansion by directly introducing popular entities' infrequent synonyms into the set and thus increasing the expansion recall. The second implication is that if two entities are *not* from the same class C, then they are likely not synonyms. This shows how set expansion can help synonym discovery by restricting the synonym search space to set expansion outputs and generating additional training data to fine tune the synonym discovery model.

At the beginning, when we only have limited seed information, this hypothesis may not be directly applicable as we do not have complete knowledge of either entity class memberships or entity synonyms. Therefore, we design our SynSetExpan as an iterative framework, shown in Fig. 2.

Framework Overview. Before the iterative process starts, we first learn a general synonym discovery model $\mathcal{M}_0$ using distant supervision from a knowledge base (cf. Sect. 3.1). Then, in each iteration, we learn a set expansion model based on the currently expanded set E (initialized as all entities in user-provided seed synsets $\mathcal{S}_0$) and apply it to obtain a rank list of entities in $\mathcal{V}$, denoted as L_{se} (cf. Sect. 3.2). Next, we generate *pseudo training data* from L_{se} and use it to construct a new *class-dependent* synonym discovery model $\mathcal{M}_c$ by fine-tuning $\mathcal{M}_0$. After that, for each entity in $\mathcal{V}$, we apply $\mathcal{M}_c$ to predict its probability of being the synonym of *at least one* entity in E and use such synonym information to adjust L_{se} (cf. Sect. 3.3). Finally, we add top-ranked entities in the adjusted rank list into the current set and start the next iteration. After the iterative process ends, we identify entity synsets from the final iteration's output using a graph-based clustering method (cf. Sect. 3.4).

3.1 Proposed Synonym Discovery Model

Given a pair of entities, our synonym discovery model returns the probability that they are synonymous. We use two types of features for entity pairs[3]: (1) *lexical features* based on entity surface names (e.g., Jaro-Winkler similarity (Wang et al., 2019), token edit distance (Fei et al., 2019), etc), and (2) *semantic features* based on entity embeddings (e.g., cosine similarity between two entities' SkipGram embeddings). As these feature values have different scales, we use a tree-based boosting model XGBoost (Chen and Guestrin, 2016) to predict whether two entities are synonyms. Another advantage of XGBoost is that it is an additive model and supports incremental model fine-tuning. We will discuss how to use set expansion results to fine-tune a synonym discovery model in Sect. 3.2.

To learn the synonym discovery model, we first acquire distant supervision data by matching each term in the vocabulary $\mathcal{V}$ with the canonical name of one entity (with its unique ID) in a knowledge base (KB). If two terms are matched to the same entity in KB and their embedding similarity is larger than 0.5, we treat them as synonyms. To generate a non-synonymous term pair, we follow the same "mixture" sampling strategy proposed in (Shen et al., 2019), that is, 50% of negative pairs come from random sampling and the other 50% of negative pairs are those "hard" negatives which are required to share at least one token. Some concrete

[3]We list all features in supplementary materials Section A.

examples are shown in Fig. 2. Finally, based on such generated distant supervision data, we train our XGBoost-based synonym discovery model using binary cross entropy loss.

3.2 Proposed Set Expansion Model

Given a set of seed entities E_0 from a semantic class C, we aim to learn a set expansion model that can predict the probability of a new entity (term) $e_i \in \mathcal{V}$ belonging to the same class C, *i.e.*, $\mathbf{P}(e_i \in C)$. We follow previous studies (Melamud et al., 2016; Mamou et al., 2018a) to represent each entity using a set of 6 embeddings learned on the given corpus $\mathcal{D}$, including SkipGram, CBOW in word2vec (Mikolov et al., 2013), fastText (Bojanowski et al., 2016), SetExpander (Mamou et al., 2018b), JoSE (Meng et al., 2019) and averaged BERT contextualized embeddings (Devlin et al., 2019). Given the bag-of-embedding representation $[\mathbf{f}^1(e_i), \mathbf{f}^2(e_i), \dots, \mathbf{f}^B(e_i)]$ of entity e_i and the seed set E_0, we define the entity feature $\mathbf{x}_i = \|_{b=1}^{6}\|_{j=1}^{|E_0|}\left[\sqrt{d^b_{ij}},\ d^b_{ij},\ (d^b_{ij})^2\right]$, where "$\|$" represents the concatenation operation, and $d^b_{ij} = \cos(\mathbf{f}^b(e_i), \mathbf{f}^b(e_j))$ is the cosine similarity between two embedding vectors. One challenge of learning the set expansion model is the lack of supervision signals — we only have a few "positive" examples (*i.e.*, entities belonging to the target class) and no "negative" examples. To solve this challenge, we observe that *the size of target class is usually much smaller than the vocabulary size*. This means if we randomly select one entity from the vocabulary, most likely it will not belong to the target semantic class. Therefore, we can construct a set of $|E_0| \times K$ negative examples by random sampling. We also test selecting only entities that have a low embedding similarity with the entities in the current set. However, our experiment shows this restricted sampling does not improve the performance. Therefore, we choose to use the simple yet effective "random sampling" approach and refer to K as "negative sampling ratio". Given a total of $|E_0| \times (K+1)$ examples, we learn a SVM classifier $g(\cdot)$ based on the above defined entity features.

To further improve set expansion quality, we repeat the above process T times (*i.e.*, randomly sample T different sets of $|E_0| \times K$ negative examples for learning T separate classifiers $\{g^t(\cdot)\}|_{t=1}^{T}$) and construct an ensemble classifier. The final classifier predicts the probability of an entity e_i belonging to the class C by averaging all individual classifiers' outputs (*i.e.*, $\mathbf{P}(e_i \in C) = \frac{1}{T}\sum_{t=1}^{T} g^t(e_i)$. Finally, we rank all entities in the vocabulary based on their predicted probabilities.

Algorithm 1: SynSetExpan Framework.

```
Input: A seed set $\mathcal{S}_0$, a vocabulary $\mathcal{V}$, a knowledge base $\mathcal{K}$, maximum iteration
       number *max_iter*, maximum size of expanded set $Z$, and model
       hyper-parameters $\{K, T, N, H\}$.
Output: A complete set of entity synsets $\mathcal{S}_{\mathcal{V}_C}$.
1  Learn a general ESD model $\mathcal{M}_0$ using distant supervision in $\mathcal{K}$;
2  $E \leftarrow$ Union of all synsets in $\mathcal{S}_0$;
3  for *iter from 1 to max_iter* do
4      $L_{se} \leftarrow$ ESEModel$(E, \mathcal{V}, K, T)$;
5      Generate pseudo training data $D_{pl}$ from $L_{se}$;
6      Construct a class-specific ESD model $\mathcal{M}_c$ by fine-tuning $\mathcal{M}_0$ on $D_{pl}$;
7      Apply $\mathcal{M}_c$ on entities in $\mathcal{V}$ and adjust $L_{se}$;
8      Add top $\lceil \frac{Z}{\text{max\_iter}} \rceil$ entities in the adjusted rank list into $E$;
9  Construct a synonym graph $\mathcal{G}$ based on final set $E$;
10 $\mathcal{S}_{\mathcal{V}_C} \leftarrow$ Louvain$(\mathcal{G})$;
11 Return $\mathcal{S}_{\mathcal{V}_C}$.
```

3.3 Two Models' Mutual Enhancements

Set Expansion Enhanced Synonym Discovery. In each iteration, we generate a set of *pseudo training* data D_{pl} from the ESE model output L_{se}, to fine-tune the general synonym discovery model $\mathcal{M}_0$. Specifically, we add an entity pair $\langle e_x, e_y \rangle$ into D_{pl} with label 1, if they are among the top 100 entities in L_{se} and $\mathcal{M}_0(e_x, e_y) \geq 0.9$. For each positive pair $\langle e_x, e_y \rangle$, we generate N negative pairs by randomly selecting $\lceil N/2 \rceil$ entities from L_{se} whose set expansion output probabilities are less than 0.5 and pairing them with both e_x and e_y. The intuition is that those randomly selected entities likely come from different semantic classes with entity e_x and e_y, and thus based on our hypothesis, they are unlikely to be synonyms. After obtaining D_{pl}, we fine-tune model $\mathcal{M}_0$ by fitting H additional trees on D_{pl} and incorporate them into the existing bag of trees in $\mathcal{M}_0$. We discuss the detailed choices of N and H in the experiment.

Synonym Enhanced Set Expansion. Given a fine-tuned class-specific synonym discovery model $\mathcal{M}_c$, the current set E, we calculate a new score for each entity $e_i \in \mathcal{V}$ as follows:

$$\text{sy-score}(e_i) = \max\{\mathcal{M}_c(e_i, e_j) | e_j \in E\}. \quad (1)$$

The above score measures the probability that e_i is the synonym of one entity in E. Based on Hypothesis 1, we know an entity with a large sy-score is likely belonging to the target class. Therefore, we use a multiplicative measure to combine this sy-score with the set expansion model's original output $\mathbf{P}(e_i \in C)$ as follows:

$$\text{final-score}(e_i) = \sqrt{\mathbf{P}(e_i \in C) \times \text{sy-score}(e_i)}. \quad (2)$$

An entity will have a large sy-score as long as it is the synonym of *one single* entity in E. Such a property is particularly important for capturing long-tail infrequent entities. For example, suppose we expand `US_States` class from a seed set {"*Illinois*", "*IL*", "*Texas*", "*TX*"}. The original set expansion model, biased toward popular entities, assigns a low score 0.57 to "*Lone Star State*" and a large score 0.78 to "*Chicago*". However, the synonym discovery model predicts, with over 99% probability, that "*Lone Star State*" is the synonym of "*Texas*" and thus has a sy-score 0.99. Meanwhile, "*Chicago*" has no synonym in the seed set and thus has a low sy-score 0.01. As a result, the final score of "*Lone Star State*" is larger than that of "*Chicago*". Moreover, we emphasize that Eq. 2 uses synonym scores to enhance, not replace, set expansion scores. A correct entity $e^\star$ that has no synonym in current set E will indeed be ranked after other correct entities that have synonyms in E. However, this is not problematic because (1) all compared entities are correct, and (2) we will not remove $e^\star$ from final results because it still outscores other erroneous entities that have the same low sy-score as $e^\star$ but much lower set expansion scores.

3.4 Synonym Set Construction

After the iterative process ends, we have a synonym discovery model $\mathcal{M}_c$ that predicts whether two entities are synonymous and an entity list E that includes entities from the same semantic class. To further derive entity synsets, we first construct a weighted synonym graph $\mathcal{G}$ where each node n_i represents one entity $e_i \in E$ and each edge (n_i, n_j) with weight w_{ij} indicates $M_c(e_i, e_j) = w_{ij}$. Then, we apply the Louvain algorithm (Blondel et al., 2008) (a popular non-overlapping community detection method) to find all clusters in $\mathcal{G}$ and treat them as entity synsets. Note here we narrow the original full vocabulary $\mathcal{V}$ to the set expansion model's final output E based on our hypothesis. We summarize our whole framework in Algorithm 1 and discuss its computational complexity in supplementary materials.

Corpus Size	# Entities	# Classes	# Queries
1.9B Tokens	1.5M	60	1200

Table 1: Our SE2 dataset statistics

4 The SE2 Dataset

To verify our hypothesis and evaluate the SynSetExpan framework, we need a dataset that contains a corpus, a vocabulary with labeled synsets, a set of complete semantic classes, and a list of seed queries. However, to the best of our knowledge, there is no such a public benchmark dataset[4]. Therefore, we build the first Synonym Enhanced Set Expansion (**SE2**) benchmark dataset in this study[5].

4.1 Dataset Construction

We construct the SE2 dataset in four steps.

1. Corpus and Vocabulary Selection. We use the Wikipedia 20171201 dump as our evaluation corpus as it contains a diverse set of semantic classes and enough context information for methods to discover those sets. We extract all noun phrases with frequency above 10 as our selected vocabulary.

2. Semantic Class Selection. We identify 60 major semantic classes based on the *DBpedia-Entity v2* (Hasibi et al., 2017) and *WikiTable* (Bhagavatula et al., 2015) entities found in our corpus. These 60 classes cover 6 different entity types (e.g., Person, Location, Organization). As such generated classes may miss some correct entities, we enlarge each class via crowdsourcing in the following step.

3. Query Generation and Class Enrichment. We first generate 20 queries for each semantic class. Then, we aggregate the top 100 results from all baseline methods (cf. Sect. 5) and obtain 17,400 $\langle$class, entity$\rangle$ pairs. Next, we employ crowdworkers on Amazon Mechanical Turk to check all those pairs. Workers are asked to view one semantic class and six candidate entities, and to select all entities that belong to the given class. On average, workers spend 40 seconds on each task and are paid \$0.1. All $\langle$class, entity$\rangle$ pairs are labeled by three workers independently and the inter-annotator agreement is 0.8204, measured by Fleiss's Kappa (k). Finally, we enrich each semantic class C_j by adding the entity e_i whose corresponding pair $\langle C_j, e_i\rangle$ is labeled "True" by at least two workers.

[4]More discussions on existing set expansion datasets are available in supplementary materials Section C.

[5]More details and analysis can be found in the Section D and E of supplementary materials.

Class Type	ESE	ESD (Lexical)	ESE (Semantic)
Location	0.3789	0.2132	0.6599
Person	0.2322	0.2874	0.5526
Product	0.0848	0.3922	0.4811
Facility	0.0744	0.2345	0.4466
Organization	0.1555	0.2566	0.4935
Misc	0.4282	0.2743	0.5715

Table 2: Difficulty of each semantic class for entity set expansion (ESE) and entity synonym discovery (ESD).

4. Synonym Set Curation. To construct synsets in each class, we first run all baseline methods to generate a candidate pool of possible synonymous term pairs. Then, we treat those pairs with both terms mapped to the same entity in WikiData as positive pairs and ask two human annotators to label the remaining 7,625 pairs. The inter-annotator agreement is 0.8431, measured by Fleiss's Kappa. Then, we construct a synonym graph where each node is a term and each edge connects two synonymous terms. Finally, we extract all connected components in this graph and treat them as synsets.

4.2 Dataset Analysis

We analyze some properties of the SE2 dataset from the following three aspects.

1. Semantic class size. The 60 semantic classes in our SE2 dataset consist on average 145 entities (with a minimum of 16 and a maximum of 864) for a total of 8697 entities. After we group these entities into synonym sets, these 60 classes consist of on average 118 synsets (with a minimum of 14 and a maximum of 800) for a total of 7090 synsets. The average synset size is 1.258 and the maximum size of one synset is 11.

2. Set expansion difficulty of each class. We define the set expansion difficulty of each semantic class as follows:

$$\text{Set-Expansion-Difficulty}(C) = \frac{1}{|C|}\sum_{e \in C} \frac{|C - \text{Top}k(e)|}{|C|}, \tag{3}$$

where $\text{Top}k(e)$ represents the set of k most similar entities to entity e in the vocabulary. We set k to be 10,000 in this study. Intuitively, this metric calculates the average portion of entities in class C that cannot be easily found by another entity in the same class. As shown in Table 2, the most difficult classes are those Location classes[6] and the easiest ones are Facility classes.

3. Synonym discovery difficulty of each class. We continue to measure the difficulty of finding synonym pairs in each class. Specifically, we calculate two metrics: (1) *Lexical difficulty* defined as the average Jaro-Winkler distance between the surface names of two synonyms, and (2) *Semantic difficulty* defined as the average cosine distance between two synonymous entities' embeddings. Table 2 lists the results. We find Product classes have the largest lexical difficulty and Location classes have the largest semantic difficulty.

[6]We exclude MISC type because by its definition classes of this type will be very random.

5 Experiments

5.1 Entity Set Expansion

Datasets. We evaluate SynSetExpan on three public datasets. The first two are benchmark datasets widely used in previous studies (Shen et al., 2017; Yan et al., 2019; Zhang et al., 2020): (1) **Wiki**, which contains 8 semantic classes, 40 seed queries, and a subset of English Wikipedia articles, and (2) **APR**, which includes 3 semantic classes, 15 seed queries, and all news articles published by Associated Press and Reuters in 2015. Note that these two datasets do not contain synonym information and are used primarily to evaluate our set expansion model performance. We decide not to augment these two datasets with additional synonym information (as we did in our SE2 dataset) in order to keep the integrity of two existing benchmarks. The third one is our proposed SE2 dataset which has 60 semantic classes, 1200 seed queries, and a corpus of 1.9 billion tokens. Clearly, our SE2 is an order of magnitude larger than previous benchmarks and covers a wider range of semantic classes.

Compared Methods. We compare the following corpus-based set expansion methods: (1) **EgoSet** (Rong et al., 2016): A method initially proposed for multifaceted set expansion using skip-grams and word2vec embeddings. Here, we treat all extracted entities forming in one set as our queries have little ambiguity. (2) **SetExpan** (Shen et al., 2017): A bootstrap method that first computes entity similarities based on selected quality contexts and then expands the entity set using rank ensemble. (3) **SetExpander** (Mamou et al., 2018b): A one-time entity ranking method based on multi-context term similarity defined on multiple embeddings. (4) **MCTS** (Yan et al., 2019): A bootstrap method combining the Monte Carlo Tree Search algorithm with a deep similarity network to estimate delayed feedback for pattern evaluation and entity scoring. (5) **CaSE** (Yu et al.,

Methods	SE2			Wiki			APR		
	MAP@10	MAP@20	MAP@50	MAP@10	MAP@20	MAP@50	MAP@10	MAP@20	MAP@50
Egoset (Rong et al., 2016)	0.583	0.533	0.433	0.904	0.877	0.745	0.758	0.710	0.570
SetExpan (Shen et al., 2017)	0.473	0.418	0.341	0.944	0.921	0.720	0.789	0.763	0.639
SetExpander (Mamou et al., 2018b)	0.520	0.475	0.397	0.499	0.439	0.321	0.287	0.208	0.120
MCTS (Yan et al., 2019)	—	—	—	0.980	0.930	0.790	0.960	0.900	0.810
CaSE (Yu et al., 2019c)	0.534	0.497	0.420	0.897	0.806	0.588	0.619	0.494	0.330
SetCoExpan (Huang et al., 2020)	—	—	—	0.976	0.964	0.905	0.933	0.915	0.830
CGExpan (Zhang et al., 2020)	0.601	0.543	0.438	**0.995**	**0.978**	0.902	**0.992**	**0.990**	0.955
SynSetExpan-NoSYN	0.612	0.567	0.484	0.991	**0.978**	**0.904**	0.985	**0.990**	**0.960**
SynSetExpan	**0.628****	**0.584***	**0.502***	—	—	—	—	—	—

Table 3: Set expansion results on three datasets. MCTS and SetCoExpan do not scale to the SE2 dataset. SynSetExpan-Full is inapplicable for Wiki and APR datasets because they contain no synonym information. The superscript * indicates the improvement is statistically significant compared to SynSetExpan-NoSYN.

2019b): Another one-time entity ranking method using both term embeddings and lexico-syntactic features. (6) **SetCoExpan** (Huang et al., 2020): A set expansion framework which generates auxiliary sets that are closely related to the target set and leverages them to guide the expansion process. (7) **CGExpan** (Zhang et al., 2020): Current state-of-the-art method that generates the target set name by querying a pre-trained language model and utilizes generated names to expand the set. (8) **SynSetExpan**: Our proposed framework which jointly conducts two tasks and enables synonym information to help set expansion. (9) **SynSetExpan-NoSYN**: A variant of our proposed SynSetExpan framework without the synonym discovery model. All implementation details and hyperparameter choices are discussed in supplementary materials Section F.

Evaluation Metrics. We follow previous studies and evaluate our results using Mean Average Precision at different top K positions: 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. To compare the performance of multiple models, we conduct statistical significance test using the two-tailed paired t-test with 99% confidence level.

Experimental Results. We analyze the set expansion performance from the following aspects.

1. Overall Performance. Table 3 presents the overall set expansion results. We can see that SynSetExpan-NoSYN achieves comparable performances with the current state-of-the-art methods on Wiki and APR datasets[7], and outperforms previous methods on SE2 dataset, which demonstrates

[7]We feel both CGExpan and our method have reached the performance limit on Wiki and APR as both datasets are relatively small and contain only a few coarse-grained classes.

Class Type	MAP@10	MAP@20	MAP@50
Person	86.7%	80.0%	93.3%
Organization	83.3%	83.3%	100%
Location	69.2%	65.4%	80.8%
Facility	85.7%	71.4%	100%
Product	100%	66.7%	100%
Misc	66.7%	66.7%	100%
Overall	**78.3%**	**71.7%**	**90.0%**

Table 4: Ratio of semantic classes on which SynSetExpan outperforms SynSetExpan-NoSYN.

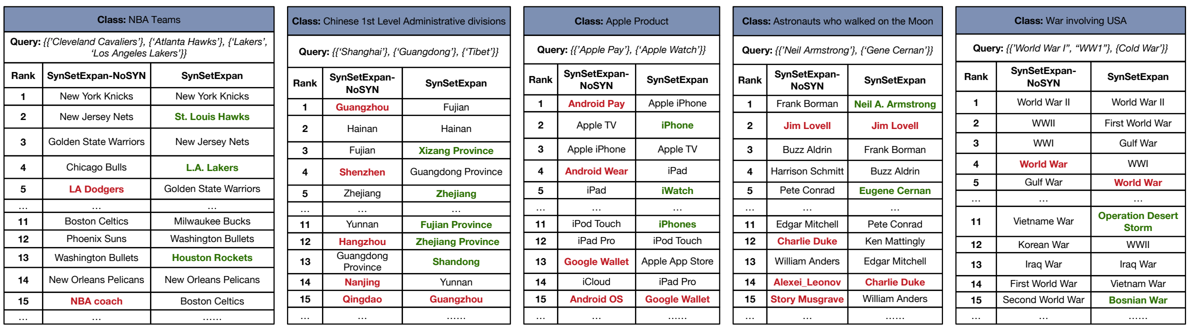
SynSetExpan vs. Other	MAP@10	MAP@20	MAP@50
vs. CGExpan	78.9%	85.4%	93.8%
vs. SynSetExpan-NoSYN	72.7%	83.0%	91.4%

Table 5: Ratio of seed queries from the SE2 dataset on which the first method outperforms the second one.

the effectiveness of our set expansion model alone. Besides, by comparing SynSetExpan-NoSYN with SynSetExpan on SE2 dataset, we show that adding synonym information indeed helps set expansion.

2. Fine-grained Performance Analysis. To provide a detailed analysis on how SynSetExpan improves over SynSetExpan-NoSYN, we group semantic classes based on their types and calculate the ratio of classes on which SynSetExpan outperforms SynSetExpan-NoSYN. Table 4 shows the results and we can see that on most classes SynSetExpan is better than SynSetExpan-NoSYN, especially for the MAP@50 metric. In Table 5, we further analyze the ratio of seed set queries (out of total 1200 queries) on which one method achieves better or the same performance as the other method. We can see that SynSetExpan can win on the majority of queries, which further shows that SynSetExpan can effectively leverage synonym information to enhance set expansion.

3. Case Studies. Figure 3 shows some expanded semantic classes by SynSetExpan. We can see that the set expansion task benefits a lot from the synonym information. Take the semantic class



Class: NBA Teams
Query: *{{'Cleveland Cavaliers'}, {'Atlanta Hawks'}, {'Lakers', 'Los Angeles Lakers'}}*

Rank	SynSetExpan-NoSYN	SynSetExpan
1	New York Knicks	New York Knicks
2	New Jersey Nets	**St. Louis Hawks**
3	Golden State Warriors	New Jersey Nets
4	Chicago Bulls	**L.A. Lakers**
5	**LA Dodgers**	Golden State Warriors
…	…	…
11	Boston Celtics	Milwaukee Bucks
12	Phoenix Suns	Washington Bullets
13	Washington Bullets	**Houston Rockets**
14	New Orleans Pelicans	New Orleans Pelicans
15	**NBA coach**	Boston Celtics
…	…	……

Class: Chinese 1st Level Administrative divisions
Query: *{{'Shanghai'}, {'Guangdong'}, {'Tibet'}}*

Rank	SynSetExpan-NoSYN	SynSetExpan
1	**Guangzhou**	Fujian
2	Hainan	Hainan
3	Fujian	**Xizang Province**
4	**Shenzhen**	Guangdong Province
5	Zhejiang	**Zhejiang**
…	…	…
11	Yunnan	**Fujian Province**
12	**Hangzhou**	**Zhejiang Province**
13	Guangdong Province	**Shandong**
14	**Nanjing**	Yunnan
15	**Qingdao**	**Guangzhou**
…	…	……

Class: Apple Product
Query: *{{'Apple Pay'}, {'Apple Watch'}}*

Rank	SynSetExpan-NoSYN	SynSetExpan
1	**Android Pay**	Apple iPhone
2	Apple TV	**iPhone**
3	Apple iPhone	Apple TV
4	**Android Wear**	iPad
5	iPad	**iWatch**
…	…	…
11	iPod Touch	**iPhones**
12	iPad Pro	iPod Touch
13	**Google Wallet**	Apple App Store
14	iCloud	iPad Pro
15	**Android OS**	**Google Wallet**
…	…	……

Class: Astronauts who walked on the Moon
Query: *{{'Neil Armstrong'}, {'Gene Cernan'}}*

Rank	SynSetExpan-NoSYN	SynSetExpan
1	Frank Borman	**Neil A. Armstrong**
2	**Jim Lovell**	**Jim Lovell**
3	Buzz Aldrin	Frank Borman
4	Harrison Schmitt	Buzz Aldrin
5	Pete Conrad	**Eugene Cernan**
…	…	…
11	Edgar Mitchell	Pete Conrad
12	**Charlie Duke**	Ken Mattingly
13	William Anders	Edgar Mitchell
14	**Alexei_Leonov**	**Charlie Duke**
15	**Story Musgrave**	William Anders
…	…	……

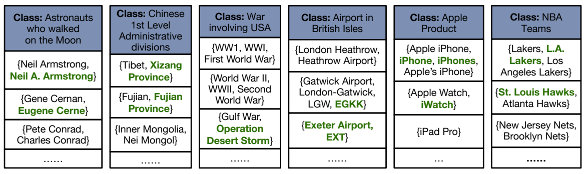
Class: War involving USA
Query: *{{'World War I", "WW1"}, {Cold War'}}*

Rank	SynSetExpan-NoSYN	SynSetExpan
1	World War II	World War II
2	WWII	First World War
3	WWI	Gulf War
4	**World War**	WWI
5	Gulf War	**World War**
…	…	…
11	Vietname War	**Operation Desert Storm**
12	Korean War	WWII
13	Iraq War	Iraq War
14	First World War	Vietnam War
15	Second World War	**Bosnian War**
…	…	……

Figure 3: Case studies on entity set expansion. Erroneous entities are colored in red. Entities discovered only by SynSetExpan in top-20 results are colored in green.

Method	SE2			PubMed		
	AP	AUC	F1	AP	AUC	F1
SVM	0.1870	0.8547	0.3300	0.2250	0.8206	0.4121
XGB-S (Chen and Guestrin, 2016)	0.7654	0.9696	0.6389	0.5012	0.8625	0.4968
XGB-E (Chen and Guestrin, 2016)	0.4762	0.8750	0.4810	0.4906	0.9190	0.5388
DPE (Qu et al., 2017)	0.7972	0.9792	0.6392	0.6338	0.8979	0.6038
SynSetMine (Shen et al., 2019)	0.7562	0.9782	0.6347	0.6757	0.9453	0.6287
SynSetExpan-NoFT	0.8197	0.9844	0.7159	0.6615	0.9445	0.6204
SynSetExpan	**0.8736**	**0.9953**	**0.7592**	**0.7152**	**0.9695**	**0.6388**

Table 6: Synonym discovery results on both **SE2** dataset and PubMed dataset.



Class: Astronauts who walked on the Moon	**Class:** Chinese 1st Level Administrative divisions	**Class:** War involving USA	**Class:** Airport in British Isles	**Class:** Apple Product	**Class:** NBA Teams
{Neil Armstrong, **Neil A. Armstrong**}	{Tibet, **Xizang Province**}	{WW1, WWI, First World War}	{London Heathrow, Heathrow Airport}	{Apple iPhone, **iPhone**, **iPhones**, Apple's iPhone}	{Lakers, **L.A. Lakers**, Los Angeles Lakers}
{Gene Cernan, **Eugene Cerne**}	{Fujian, **Fujian Province**}	{World War II, WWII, Second World War}	{Gatwick Airport, London-Gatwick, LGW, **EGKK**}	{Apple Watch, **iWatch**}	{**St. Louis Hawks**, Atlanta Hawks}
{Pete Conrad, Charles Conrad}	{Inner Mongolia, Nei Mongol}	{Gulf War, **Operation Desert Storm**}	{**Exeter Airport, EXT**}	{iPad Pro}	{New Jersey Nets, Brooklyn Nets}
……	……	……	……	…	……

Figure 4: Case studies on synonym discovery. Entities discovered only by SynSetExpan are colored in green.

NBA_Teams for example, we find "*L.A. Lakers*" (*i.e.*, the synonym of "*Los Angeles Lakers*") as well as "*St. Louis Hawks*" (*i.e.*, the former name of "*Atlanta Hawks*") and further use them to improve the set expansion result. Moreover, by introducing synonyms, we can lower the rank of those erroneous entities (e.g., "*LA Dodgers*" and "*NBA coach*").

5.2 Synonym Discovery

Datasets. We evaluate SynSetExpan for synonym discovery task on two datasets: (1) **SE2**, which contains 60,186 synonym pairs (3,067 positive pairs and 57,119 negative pairs), and (2) **PubMed**, a public benchmark used in (Qu et al., 2017; Shen et al., 2019), which contains 203,648 synonym pairs (10,486 positive pairs and 193,162 negative pairs). More details can be found in supplementary materials Section G.1.

Compared Methods. We compare following synonym discovery methods: (1) **SVM**: A classification method trained on given term pair features. We use the same feature set described in Sect. 3.1. (2) **XGBoost** (Chen and Guestrin, 2016): Another classification method trained on given term pair features. Here, we test its two variants: **XGB-S** which only leverages lexical features based on entity surface names, and **XGB-E** which only utilizes entity embedding features. (3) **DPE** (Qu et al., 2017): A distantly supervised method integrating embedding features and textual patterns for synonym discovery. (4) **SynSetMine** (Shen et al., 2019): Another distantly supervised framework that learns to represent the entire entity synonym set. (5) **SynSetExpan**: Our proposed framework that fine-tunes synonym discovery model using set expansion results. (6) **SynSetExpan-NoFT**: A variant of SynSetExpan without using the model fine-tuning. More implementation details and hyper-parameter choices are discussed in supplementary materials Section G.

Evaluation Metrics. As all compared methods output the probability of two input terms being synonyms, we first use two threshold-free metrics for evaluation — Average Precision (**AP**) and Area Under the ROC Curve (**AUC**). Second, we transform the output probability to a binary decision using threshold 0.5 and evaluate the model performance using standard **F1** score.

Experimental Results. Table 6 shows the overall synonym discovery results. First, we can see that the SynSetExpan-NoFT model can outperform both XGB-S and XGB-E methods significantly, which shows the importance of using both types of features for predicting synonyms. Second, we find that SynSetExpan can further improve SynSetExpan-NoFT via model fine-tuning, which demonstrates that set expansion can help synonym discovery. Finally, we notice that our SynSetExpan framework, with the fine-tuning mechanism enabled, can achieve the best performance across all evaluation metrics. In Figure 4, we show some synsets discovered by SynSetExpan. We can see that SynSetExpan is able to detect different types of entity synsets across various semantic classes. Furthermore, we highlight those entities discovered only after model fine-tuning, and we can see clearly that with fine-tuning, our SynSetExpan framework can detect more accurate synsets.

6 Related Work

Entity Set Expansion. Entity set expansion can benefit many downstream applications such as question answering (Wang and Cohen, 2008), literature search (Shen et al., 2018b), and online education (Yu et al., 2019a). Traditional entity set expansion systems such as GoogleSet (Tong and Dean, 2008) and SEAL (Wang and Cohen, 2007) require seed-oriented online data extraction, which can be time-consuming and costly. Thus, more recent studies (Shen et al., 2017; Mamou et al., 2018b; Yu et al., 2019c; Huang et al., 2020; Zhang et al., 2020) are proposed to expand the seed set by offline processing a given corpus. These corpus-based methods include two general approaches: (1) *one-time entity ranking* (Pantel et al., 2009; He and Xin, 2011; Mamou et al., 2018b; Kushilevitz et al., 2020) which calculates all candidate entities' distributional similarities with seed entities and makes a one-time ranking without back and forth refinement, and (2) *iterative bootstrapping* (Rong et al., 2016; Shen et al., 2017; Huang et al., 2020; Zhang et al., 2020) which starts from seed entities to extract quality textual patterns; applies the extracted patterns to obtain more quality entities, and iterates this process until sufficient entities are discovered. In this work, in addition to just adding entities into the set, we go beyond one step and aim to organize those expanded entities into synonym sets. Furthermore, we show those detected synonym sets can in turn help to improve set expansion results.

Synonym Discovery. Early efforts on synonym discovery focus on finding entity synonyms from structured or semi-structured data such as query logs (Ren and Cheng, 2015), web tables (He et al., 2016), and synonymy dictionaries (Ustalov et al., 2017b,a). In comparison, this work aims to develop a method to extract synonym sets directly from raw text corpus. Given a corpus and a term list, one can leverage surface string (Wang et al., 2019), co-occurrence statistics (Baroni and Bisi, 2004), textual pattern (Yahya et al., 2014), distributional similarity (Wang et al., 2015), or their combinations (Qu et al., 2017; Fei et al., 2019) to extract synonyms. These methods mostly find synonymous *term pairs* or *a rank list* of query entity's synonym, instead of entity synonym sets. Some studies propose to further cut-off the rank list into a set output (Ren and Cheng, 2015) or to build a synonym graph and then apply graph clustering techniques to derive synonym sets (Oliveira and Gomes, 2014; Ustalov et al., 2017b). However, they all operate directly on the entire input vocabulary which can be too extensive and noisy. Compared to them, our approach can leverage the semantic class information detected from set expansion to enhance the synonym set discovery process.

7 Conclusions

This paper shows entity set expansion and synonym discovery are two tightly coupled tasks and can mutually enhance each other. We present SynSetExpan, a novel framework jointly conducting two tasks, and SE2 dataset, the first large-scale synonym-enhanced set expansion dataset. Extensive experiments on SE2 and several other benchmark datasets demonstrate the effectiveness of SynSetExpan on both tasks. In the future, we plan to study how we can apply SynSetExpan at the entity mention level for conducting contextualized synonym discovery and set expansion.

Acknowledgements

Research was sponsored in part by US DARPA SocialSim Program No. W911NF-17-C0099, NSF IIS 16-18481, IIS 17-04532, and IIS 17-41317, and DTRA HDTRA11810026. Any opinions, findings 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. We thank anonymous reviewers for valuable and insightful feedback.

References

Marco Baroni and Sabrina Bisi. 2004. Using cooccurrence statistics and the web to discover synonyms in a technical language. In *LREC*.

Chandra Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. Tabel: Entity linking in web tables. In *ISWC*.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *TACL*.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *KDD*.

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

Hongliang Fei, Shulong Tan, and Ping Li. 2019. Hierarchical multi-task word embedding learning for synonym prediction. In *KDD*.

Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and James P. Callan. 2017. Dbpedia-entity v2: A test collection for entity search. In *SIGIR*.

Yeye He, Kaushik Chakrabarti, Tao Cheng, and Tomasz Tylenda. 2016. Automatic discovery of attribute synonyms using query logs and table corpora. In *WWW*.

Yeye He and Dong Xin. 2011. Seisa: set expansion by iterative similarity aggregation. In *WWW*.

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.

Guy Kushilevitz, Shaul Markovitch, and Yoav Goldberg. 2020. A two-stage masked lm method for term set expansion. In *ACL*.

Jonathan Mamou, Oren Pereg, Moshe Wasserblat, Ido Dagan, Yoav Goldberg, Alon Eirew, Yael Green, Shira Guskin, Peter Izsak, and Daniel Korat. 2018a. Term set expansion based on multi-context term embeddings: an end-to-end workflow. In *COLING*.

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

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016. The role of context types and dimensionality in learning word embeddings. In *HLT-NAACL*.

Yu Meng, Jiaxin Huang, Guangyuan Wang, Chao Zhang, Honglei Zhuang, Lance Kaplan, and Jiawei Han. 2019. Spherical text embedding. In *NeurIPS*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Hugo Gonalo Oliveira and Paulo Gomes. 2014. Eco and onto.pt: a flexible approach for creating a portuguese wordnet automatically. *Language Resources and Evaluation*, 48:373–393.

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

Meng Qu, Xiang Ren, and Jiawei Han. 2017. Automatic synonym discovery with knowledge bases. In *KDD*.

Xiang Ren and Tao Cheng. 2015. Synonym discovery for structured entities on heterogeneous graphs. In *WWW*.

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*.

Jiaming Shen, Ruiilang Lv, Xiang Ren, Michelle Vanni, Brian Sadler, and Jiawei Han. 2019. Mining entity synonyms with efficient neural set generation. In *AAAI*.

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.

Dmitry Ustalov, Mikhail Chernoskutov, Christian Biemann, and Alexander Panchenko. 2017a. Fighting with the sparsity of synonymy dictionaries for automatic synset induction. In *AIST*.

Dmitry Ustalov, Alexander Panchenko, and Christian Biemann. 2017b. Watset: Automatic induction of synsets from a graph of synonyms. In *ACL*.

Huazheng Wang, Bin Gao, Jiang Bian, Fei Tian, and Tie-Yan Liu. 2015. Solving verbal comprehension questions in iq test by knowledge-powered word embedding. *CoRR*, abs/1505.07909.

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*.

Zhen Wang, Xiang An Yue, Soheil Moosavinasab, Yungui Huang, Simon Lin, and Huan Sun. 2019. Surfcon: Synonym discovery on privacy-aware clinical data. In *KDD*.

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

Mohamed Yahya, Steven Euijong Whang, Rahul Gupta, and Alon Y. Halevy. 2014. Renoun: Fact extraction for nominal attributes. In *EMNLP*.

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

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 Allan. 2019b. Efficient corpus-based set expansion with lexico-syntactic features and distributed representations. In *SIGIR*.

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

Shuo Zhang and Krisztian Balog. 2018. On-the-fly table generation. In *SIGIR*.

Yunyi Zhang, Jiaming Shen, Jingbo Shang, and Jiawei Han. 2020. Empower entity set expansion via language model probing. In *ACL*.

Wanzheng Zhu, Hongyu Gong, Jiaming Shen, Chao Zhang, Jingbo Shang, S. Bhat, and J. Han. 2020. Fuse: Multi-faceted set expansion by coherent clustering of skip-grams. In *ECMLPKDD*.

A Entity Pair Features

Feature Description	Example
IsPrefix	(Florida, FL) $\rightarrow$ 1
IsInitial	(North Carolina, NC) $\rightarrow$ 1
Edit distance	(North Carolina, Texas) $\rightarrow$ 13
Jaro-Winkler similarity	(Arizona, Texas) $\rightarrow$ 0.4476
Characters in common	(Lone Star State, Texas) $\rightarrow$ 2
Tokens in common	(North Carolina, South Carolina) $\rightarrow$ 1
Difference in #tokens	(Land of Lincoln, Illinois) $\rightarrow$ |3-1| = 2
Initial edit distance	(North Carolina, State of North Carolina) $\rightarrow$ 2
Longest token edit distance	(North Dakota, North Carolina) $\rightarrow$ 5
Cosine similarity of embedding	(Texas, Lone Star State) $\rightarrow$ 0.9
Transformed cosine similarities	(Texas, Lone Star State) $\rightarrow [\frac{1}{0.9}, \sqrt{0.9}, (0.9)^2]$
Multiplication of two entities' PCA-reduced embedding	(Illinois, Land of Lincoln) $\rightarrow$ [0.006, 0.072, -0.008, 0.074, $\cdots$, -0.004]

Table 7: All entity pair features used in our synonym discovery model.

B SynSetExpan Framework Complexity

From the Algorithm 1 in the main text, we can see our SynSetExpan framework costs $O(T \times (1 + K) \times |S| + |\mathcal{V}|)$ for each iteration, where T is the ensemble times (usually 50), K is the negative sampling size (usually 10-20), S is the currently expanded set (usually of size < 100), and $|\mathcal{V}|$ is the vocabulary size. Although such complexity looks expensive, we can significantly reduce the practical running time in two ways. First, we can learn T separate classifiers in set expansion model in parallel. Second, we can aggregate all words in the vocabulary into one batch and apply synonym discovery model for inference in one run. We report the practical running time for each component in the below experiments.

C Existing ESE Datasets

An ideal set expansion benchmark dataset should contain four parts: a corpus, a vocabulary, a set of complete semantic classes, and a collection of seed queries for each semantic class. One of the earliest corpus-based set expansion work (Pantel et al., 2009) uses "*List of*" pages in Wikipedia to construct 50 semantic classes and applies random sampling to construct 30 queries for each class. Although those classes and queries are still available today, we have no access to its underlying corpus and vocabulary and thus cannot easily reproduce their results. Similarly, SEISA (He and Xin, 2011) and EgoSet (Rong et al., 2016) also release their constructed semantic classes and seed queries but hold the corpus and vocabulary. On the other side, SetExpander (Mamou et al., 2018b) and CaSE (Yu et al., 2019b) clearly describe their corpus and vocabulary but do not release their classes/queries. To the best of our knowledge, SetExpan (Shen et al., 2017) is the only public dataset consisting of all four essential components. However, it only contains 65 queries from 13 classes and has no synonym information. Below Table 8 compares our proposed SE2 with all existing datasets and we can see that our new dataset contains all four key parts for a set expansion benchmark dataset, as well as additional synonym information.

Dataset	Corpus	Vocab	Classes	Queries	Synonyms
Pantel *et al.* (Pantel et al., 2009)	×	×	✓	✓	×
SEISA (He and Xin, 2011)	×	×	✓	✓	×
EgoSet (Rong et al., 2016)	×	×	✓	✓	×
SetExpander (Mamou et al., 2018b)	✓	✓	×	×	×
CaSE (Yu et al., 2019b)	✓	✓	×	×	×
SetExpan (Shen et al., 2017)	✓	✓	✓	✓	×
SE2	✓	✓	✓	✓	✓

Table 8: Comparison of ESE datasets.

D SE2 Dataset Construction Details

We construct our dataset in four stages: (1) Corpus and vocabulary selection, (2) Semantic class selection, (3) Query generation and class enrichment, and (4) Synonym set curation.

Corpus and vocabulary selection. An ideal corpus for set expansion task should contain a diverse set of semantic classes and enough context information for methods to discover those sets. Based on these two criteria, we select Wikipedia 20171201 dump as our evaluation corpus. This corpus is also used in previous studies (Mamou et al., 2018b,a) and contains 1.9 billion tokens of raw size 14GB. Next, we extract all noun phrases with frequency above 10 and filter out those noun phrases that start with either a stopword (e.g., "*a/an*" and "*the*") or a non-word character (e.g., "*(*", and "-"). The remaining 1.47 million noun phrases consist of our vocabulary.

Semantic class selection. To select a diverse set of semantic classes, we first use simple string matching to align our corpus and vocabulary with two benchmark datasets designed for tasks closely related to Set Expansion: (1) *DBpedia-Entity v2* (Hasibi et al., 2017) for Entity Search (particularly entity list search), and (2) *WikiTable* (Bhagavatula et al., 2015; Zhang and Balog, 2018) for Entity Linking in Wikipedia Table. Then, we retain all semantic classes with at least 10 entities and obtain totally 60 classes covering 6 different types (e.g., Person, Location, Organization, etc). Table 9 shows some examples. Such generated classes have high precision but low recall in the sense that some correct entities are not included. In the following stage, we enlarge each semantic class and increase its coverage using crowdsourcing.

Query generation and class enrichment. For each semantic class, we generate 5 queries for each of four query sizes: 2, 3, 4, 5, which results in 20 queries per class and 1200 queries in total. Furthermore, we want those queries to cover both popular and long-tail entities. To achieve this goal, we first sort all entities based on their frequencies within each class. Then, we generate each subgroup of 5 queries (of the same size $M \in \{2, 3, 4, 5\}$) as follows: we select 1 query consisting of the M most frequent entities, 2 queries of entities in frequency quantile top-10%, and 2 queries of entities in frequency quantile [top-10%, top-30%].

After generating queries, we run all baseline methods to retrieval their top 100 results and aggregate all results to a set of 17,400 $\langle \text{class}, \text{entity} \rangle$ pairs. Next, we employ crowdworkers to check all those pairs on Amazon Mechanical Turk. Crowdworkers are required to have a 95% HIT acceptance rate, a minimum of 1000 HITs, and be located in the United States or Canada. Workers are asked to view one semantic class and six candidate entities, and to select all entities that belong to the given class. On average, workers spend 40 seconds on each task and are paid \$0.1, which is equivalent to a \$9 hourly payment. All $\langle \text{class}, \text{entity} \rangle$ pairs are labeled by three workers independently and the inter-annotator agreement is 0.8204, measured by Fleiss's Kappa (k). Finally, we enrich each semantic class C_j by adding the entity e_i whose corresponding pair $\langle C_j, e_i \rangle$ is labeled "True" by at least two workers.

Synonym set curation. To construct synonym sets in each semantic class, we first run all baseline methods to generate a candidate pool of possible synonymous pairs. Then, we enlarge this pool to include all term pairs that form an inflection[8]. After that, we automatically treat those terms that can be mapped to the same entity in WikiData[9] as positive pairs and manually label the remaining 7,625 pairs. The inter-annotator agreement is 0.8431. Note here we do not use Amazon MTurk because labeling synonym pairs are much simpler than labeling entity class membership and also has less ambiguity. Here, we avoid using YAGO KB in order to prevent possible data leakage problem. Next, we construct a synonym graph where each node is a term and each edge connects two synonymous terms. Finally, we extract all connected components in this synonym graph and treat them as synonym sets.

E SE2 Dataset Analysis

We analyze some properties of SE2 dataset from the following aspects: (1) semantic class size, (2) set expansion difficulty of each class, and (3) synonym discovery difficulty of each class.

Semantic class size. The 60 semantic classes in our SE2 dataset consist on average 145 entities (with a minimum of 16 and a maximum of 864) for a total of 8697 entities. After we grouping these entities into synonym sets, these 60 classes consist of on average 118 synsets (with a minimum of 14 and a maximum of 800) for totally 7090 synsets. The average synset size is 1.258 and the maximum size of one synset is 11.

Set expansion difficulty of each class. We define the set expansion difficulty of each semantic class as follows:

$$\text{Set-Expansion-Difficulty}(C) = \frac{1}{|C|} \sum_{e \in C} \frac{|C - \text{Top}k(e)|}{|C|}, \tag{4}$$

where Top$k(e)$ represents the set of k most similar entities to entity e in the vocabulary. We set k to be 10,000 in this study. Intuitively, this metric calculates the average portion of entities in class C that cannot be easily found by another entity in the same class. As shown in Table 2, the most difficult classes are those LOC classes[10] and the easiest ones are FAC classes.

Synonym discovery difficulty of each class. We continue to measure the difficulty of finding synonym pairs in each class. Specifically, we calculate two metrics: (1) *Lexical difficulty* defined as the average Jaro-Winkler distance[11] between the surface names of two synonyms, and (2) *Semantic difficulty* defined as the average cosine distance between two synonymous entities' embeddings. Table 2 lists the results. We find PRODUCT classes have the largest lexical difficulty and LOC classes have the largest semantic difficulty.

[8]We check word inflection using: https://github.com/jazzband/inflect.

[9]https://www.wikidata.org/wiki/Wikidata:Main_Page

[10]We exclude MISC type because by its definition classes of this type will be very random.

[11]We use Jaro-Winkler distance instead of other edit distances because it is symmetric, normalized to range from 0 to 1, and is widely used in previous synonym literature.

Class ID	Class Name	Class Type (Class Description)	Entities with Synsets
WikiTable-21	U.S. states	LOC (Locations)	[{"*Texas*", "*TX*", "*Lone Star State*"}, {"*Arizona*", "*AZ*"}, {"*California*", "*CA*", "*Golden State*"}, ...]
SemSearch-LS-3	Astronauts who landed on the Moon	PERSON (People)	[{"*Eugene Andrew Cernan*", "*Gene Cernan*"}, {"*Pete Conrad*"}, {"*Neil A. Armstrong*", "*Neil Armstrong*"}, ...]
Enriched-1	Apple Products	PRODUCT (Objects, vehicles, ...)	[{"*MacBook Pro*", "*MBP*"}, { "*iTouch*", "*iPod Touch*"}, ...]
Enriched-3	Volcanoes in USA	LOC (Non-GPE locations)	[{"*Yellowstone*"}, {"*Mount Rainier*", "*Tahoma*", "*Tacoma*"}, {"*Mount Hood*", "*Mt. Hood*", "*Wy'east*"}, ...]
WikiTable-27	Airports in British Isles	FAC (Facilities)	[{"*Ringway Airport*", "*Manchester Airport*" }, {"*RAF Exeter*", "*Exeter International Airport*"}, ...]
Enriched-4	NBA Teams	ORG (Organizations)	[{"*Washington Bullets*", "*Washington Wizards*" }, {"*Los Angeles Lakers*", "*L.A. Lakers*", "*Lakers*"}, ...]
INEX-XER-147	Chemical elements that are named after people	MISC (Miscellaneous classes)	[{"*Gadolinium*"}, {"*Seaborgium*", "*Element 106*"}, {"*Einsteinium*", "*Es99*"}, ...]

Table 9: Example Semantic Classes in **SE2** Dataset.

Class Type	ESE	ESD (Lexical)	ESE (Semantic)
Location	0.3789	0.2132	0.6599
Person	0.2322	0.2874	0.5526
Product	0.0848	0.3922	0.4811
Facility	0.0744	0.2345	0.4466
Organization	0.1555	0.2566	0.4935
Misc	0.4282	0.2743	0.5715

Table 10: Difficulty of each semantic class for entity set expansion (ESE) and entity synonym discovery (ESD).

F Entity Set Expansion Experiments

F.1 Implementation Details and Hyper-parameter Choices

For Wiki and APR datasets, we directly report each baseline method's performance obtained in the CG-Expan paper (Zhang et al., 2020). For our proposed SE2 dataset, we tune each method's hyper-parameters on 6 semantic classes (one for each class type) and use tuned parameters for all the other classes. The implementation details and specific hyper-parameter choices are discussed below:

1. **EgoSet**: There is no open-source code for EgoSet and thus we implement it on our own. We use each entity's 250 most relevant skip-grams to calculate entity-entity similarity.

2. **SetExpan**[12]: We run SetExpan for 10 iterations and add 10 entities into the set in each iteration. We set ensemble time to be 90 and use the default values for all other hyper-parameters.

3. **SetExpander**[13]: We directly download the pre-trained vectors (as they are trained on the same corpus as ours) and filter out those words that do not exist in our vocabulary.

4. **MCTS**[14]: In each iteration, we perform 1000 MCTS simulations and select 10 patterns to add 10 entities.

5. **CaSE**[15]: We use its CaSE-BERT version where a BERT-base-uncased model is used to calculate entity representations.

6. **CGExpan**[16]: We use BERT-base-uncased as its underlying Language Model for generating class names. We run CGExpan for 5 iterations and each iteration finds 5 candidate classes and adds 10 most confident entities into the currently expanded set.

7. **SynSetExpan**: We set the ensemble times $T = 50$, the negative sampling ratio (in set expansion model) $K = 10$, the maximum iteration number *max_iter*= 6, the number of fine-tuning trees $H = 10$, and the negative sampling ratio (in synonym discovery model) $N = 10$. For other (less important) hyper-parameters, we directly discuss their values in the paper and SynSetExpan is robust to those hyper-parameters.

F.2 Hyper-parameter Sensitivity Analysis

Within our SynSetExpan framework, there are two important hyper-parameters in the set expansion model: the ensemble times T and negative sampling ratio K. Figure 5 shows the hyper-parameter sensitivity analysis. We see that the model performance first increases when the ensemble times T increases from 1 to 10 and then becomes stable when T further increases. A similar trend is also witnessed on the negative sampling ratio K. Overall, we can say that SynSetMine is insensitive to

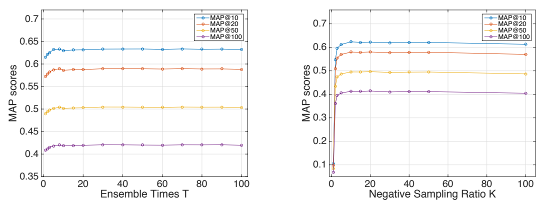
[12]`https://github.com/mickeystroller/SetExpan`

[13]`http://nlp_architect.nervanasys.com/term_set_expansion.html`

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

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

[16]`https://github.com/yzhan238/CGExpan`



(a) Ensemble Times T (b) Negative Ratio K

Figure 5: Sensitivity analysis of hyper-parameters T and K in SynSetExpan for the set expansion task.

these two hyper-parameters as long as their values are larger than 10.

F.3 Efficiency Analysis

We test the efficiency of our SynSetExpan framework (with T = 50 and K = 10) on a single server with 20 CPU threads. For each query, the first iteration of SynSetExpan on average takes 7.5 seconds, the first three iterations need 27 seconds, and the first six iterations consume 56 seconds. Later iterations take longer time because there are more entities in the already expanded set of that iteration. In comparison, one iteration of EgoSet takes 86 seconds, six iterations of SetExpan need 188 seconds, and CGExpan takes 174 seconds for five iterations on a 1080Ti GPU. This result shows the efficiency of SynSetExpan.

G Synonym Discovery Experiments

G.1 PubMed Dataset Details

Besides using our SE2 dataset, we also evaluate SynSetExpan for synonym discovery task on the public PubMed dataset which consists of a corpus of 1.5 million paper abstracts in biomedical domain, a vocabulary of 357,991 terms, and a collection of 203,648 synonym pairs (10,486 positive pairs and 193,162 negative pairs). All terms involved in synonym pairs are linked to one entity in UMLS knowledge base[17] and we group these terms into 10 semantic classes based on their linked entities' types.

G.2 Implementation Details and Hyper-parameter Choices

All compared synonym discovery methods are tested using the same distant supervision data (c.f. Section 3 in the main text) and hyper-parameter values are obtained using 5-fold cross validation. We discuss the implementation details and hyper-parameter choices of each compared synonym discovery methods below:

1. **SVM**[18]: We use the RBF kernel and set regularization parameter λ to be 0.3.
2. **XGBoost**[19]: We set the maximum tree depth to be 5, $\gamma = 0.1$, $\eta = 0.1$, subsample ratio to be 0.5, and use the default values for all other hyper-parameters.
3. **SynSetMine**[20]: We use two hidden layers (of dimension 250, 500) for its internal set encoder. We learn the model using the "mix sampling" strategy.
4. **DPE**[21]: We set the embedding dimension as 300, $\lambda = 0.3$, and use the default values for all other hyper-parameters.
5. **SynSetExpan**: We use the same hyper-parameter values as XGBoost to obtain the class-agonistic synonym discovery model. During the fine-tuning stage, we fit 10 additional trees in each iteration. For other (less important) hyper-parameters, we directly discuss their values in the paper and SynSetExpan is robust to those hyper-parameters.

G.3 Hyper-parameter Sensitivity Analysis

We study how sensitive SynSetExpan is to the choices of two fine-tuning hyper-parameters in its synonym discovery module: (1) the number of additional fitted trees H, and (2) the negative sampling ratio N in constructing pseudo-labeled dataset for fine-tuning. Results are shown in Figure 6. First, we find that our model is insensitive to the negative sampling ratio N in terms of all three metrics. Second, we notice that the model performance first increases as H increases until it reaches about 15 and then starts to decrease when we further increase H. Although SynSetExpan is somewhat sensitive to the hyper-parameter H, we find that a wide range of H choices are better than $H = 0$ which essentially disables the model fine-tuning.

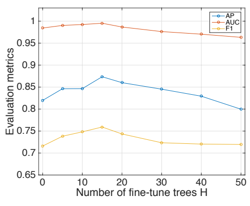
[17]https://uts.nlm.nih.gov/home.html

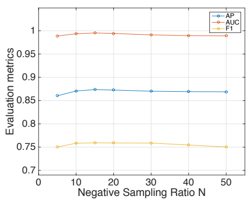
[18]https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

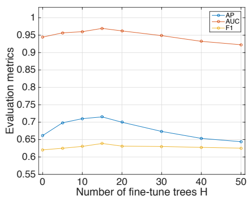
[19]https://github.com/dmlc/xgboost

[20]https://github.com/mickeystroller/SynSetMine-pytorch

[21]https://github.com/mnqu/DPE


(a) SE2 Dataset-H


(b) SE2 Dataset-N


(c) PubMed Dataset-H

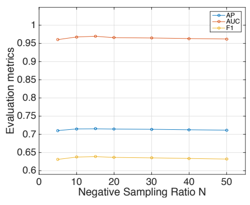

(d) PubMed Dataset-N

Figure 6: Sensitivity analysis of hyper-parameters H and N in SynSetExpan framework for the synonym discovery task.

G.4 Efficiency Analysis

By linking SE2 Dataset with YAGO KB, we can obtain 260 thousand synonym pairs based on which training a class-agnostic synonym discovery model takes 15 minutes. Then, each iteration of SynSetExpan generates on average 5000 pseudo-labeled synonym pairs and fitting 10 additional trees needs about 0.75 seconds. After training, our synonym discovery model can predict 4000 term pairs per second.