Set Expansion using Sibling Relations between Semantic Categories

Sho Takase Naoaki Okazaki Kentaro Inui
Graduate School of Information Sciences
Tohoku University, Japan
{takase, okazaki, inui}@ecei.tohoku.ac.jp

Abstract

Most set expansion algorithms assume to acquire new instances of different semantic categories independently even when we have seed instances of multiple semantic categories. However, in the setting of set expansion with multiple semantic categories, we might leverage other types of prior knowledge about semantic categories. In this paper, we present a method of set expansion when ontological information related to target semantic categories is available. More specifically, the proposed method makes use of sibling relations between semantic categories as an additional type of prior knowledge. We demonstrate the effectiveness of sibling relations in set expansion on the dataset in which instances and sibling relations are extracted from Wikipedia in a semi-automatic manner.

1 Introduction

Set expansion is the task of expanding a list of named entities from a few named entities (seed instances). For example, given a few instances of car vehicles “Prius”, “Lexus” and “Insight”, the task outputs new car instances such as “Corolla”, “Civic”, and “Fit”. Set expansion has many applications in NLP including named entity recognition (Collins and Singer, 1999), word sense disambiguation (Pantel and Lin, 2002), document categorization (Pantel et al., 2009), and query suggestion (Cao et al., 2008).

Set expansion is often implemented as bootstrapping algorithms (Hearst, 1992; Yarowsky, 1995; Abney, 2004; Pantel and Ravichandran, 2004; Pantel and Pennacchiotti, 2006). A bootstrapping algorithm iteratively acquires new instances of the target category using seed instances. First, a bootstrapping algorithm mines phrasal patterns that co-occur frequently with seed instances in a corpus. Given the words “Prius” and “Lexus” as seed instances of the car category, the algorithm finds patterns such as “Toyota produce X” and “X is a hybrid car” (X is a variable filled with a noun phrase). Next, the bootstrapping algorithm acquires instances that co-occur with patterns, i.e., noun phrases that appear frequently in the variable slots in the patterns. For example, the pattern “Toyota produce X” might retrieve vehicles manufactured by Toyota. Bootstrapping algorithms repeat these steps, expanding patterns using newly acquired instances.

However, bootstrapping algorithms often suffer from patterns that retrieve instances not only of the target category but also of other categories. For example, given the seed instances “Prius” and “Lexus”, a bootstrapping algorithm might choose the pattern “new type of X”, which might extract unrelated instances such as “iPhone” and “ThinkPad”. The *semantic drift* problem (Curran et al., 2007), the phenomenon by which a bootstrapping algorithm deviates from the target category, has persisted as the major impediment of bootstrapping algorithms.

Bootstrapping algorithms assume prior knowledge about a semantic category in the form of seed instances. Recently, researchers have been more interested in self-supervised learning of every semantic category in the world from massive text corpora, as exemplified by the *Machine Reading* project (Oren et al., 2006). In the setting of set expansion with multiple semantic categories, we might leverage prior knowledge of other types that were unexplored in previous studies. For example, a person cannot belong to both actor and actress categories simultaneously. Additionally, we know that two distinct categories of car and motorcycle products share similar properties (e.g., vehicle, gasoline-powered, overland), but have some crucial differences (e.g., with two or four wheels, with or without windows).

In this paper, we present a method of set expansion when ontological information related to target semantic categories is available. More specifically, the proposed method makes use of sibling relations between semantic categories as an additional type of prior knowledge. We demonstrate the effectiveness of sibling relations on the dataset (seed and test instances) extracted from Wikipedia.

This paper is organized as follows. Section 2 reviews the Espresso algorithm as the baseline algorithm of this study. The section also describes the problem of semantic drift and previous approaches to the problem. Section 3 presents the proposed method, which uses sibling relations of semantic categories as an additional source of prior knowledge. Section 4 demonstrates the effectiveness of the proposed method and discusses the experimental results. In section 5, we conclude this paper.

2 Related Work

2.1 Espresso algorithm

Pantel and Pennacchiotti (2006) proposed the Espresso algorithm, which fundamentally iterates two steps: candidate extraction and ranking. In candidate extraction, the algorithm collects patterns that are co-occurring with seed instances and instances acquired in the previous iteration. The algorithm also finds candidates of new instances using patterns extracted in the previous iteration.

In the ranking step, the algorithm finds the top N candidates of patterns and instances based on their scores. The espresso algorithm defines score $r_\pi(p)$ for candidate pattern p and score $r_\iota(i)$ for the candidate instance i as

$$r_\pi(p) = \frac{1}{|I|} \sum_{i \in I} \frac{\mathrm{pmi}(i,p)}{\max \mathrm{pmi}} r_\iota(i), \tag{1}$$

$$r_\iota(i) = \frac{1}{|P|} \sum_{p \in P} \frac{\mathrm{pmi}(i,p)}{\max \mathrm{pmi}} r_\pi(p), \tag{2}$$

$$\mathrm{pmi}(i,p) = \log_2 \frac{|i,p|}{|i,*||*,p|}. \tag{3}$$

In these equations, P and I are sets of patterns and instances of each category. $|P|$ and $|I|$ are the numbers of patterns and instances in the sets. $|i,*|$ and $|*,p|$ are the frequencies of instance i and pattern p in a given corpus. $|i,p|$ presents the frequency by which instance i co-occurs with pattern p. Also, max pmi is the maximum of pmi values in all instances and patterns.

First, the Espresso algorithm extracts patterns that co-occur with seed instances. Next, the algorithm ranks the patterns based on their score calculated using equation (1) and acquires the top N patterns. The more a pattern co-occurs with reliable instances, the higher the score the pattern obtains. In this way, the algorithm acquires patterns that correspond to the target semantic category.

2.2 Semantic Drift

The major obstacle of bootstrapping algorithms is semantic drift (Curran et al., 2007). Semantic drift is the phenomenon by which a bootstrapping algorithm deviates from the target categories. For example, one can consider the car category, which includes “Prius” and “Lexus” as seed instances. The Espresso algorithm might extract patterns that co-occur with many categories such as “new type of X” and “performance of X” after some iterations. These generic patterns might gather unrelated instances such as “iPhone” and “ThinkPad”. These instances obscure the characteristics of the target semantic category. Therefore, the patterns extracted in the next iteration might not represent at the semantic category of the seed instances.

Semantic drift is also caused by polysemous words. For example, to expand the set of motor vehicle manufacturers using seed instances “Saturn” and “Subaru”, a bootstrapping algorithm might find instances representing the star category (e.g., “Jupiter” and “Uranus”). This is because “Saturn” and “Subaru” are polysemous words, belonging not only to motor vehicle manufacture but also to astronomical objects: planets and stars.

2.3 Approaches to semantic drift

Many researchers have presented various approaches to reduce the effects of semantic drift. The approaches range from refinement of the seed set (Vyas et al., 2009), applying classifier (Bellare et al., 2007; Sadamitsu et al., 2011; Pennacchiotti and Pantel, 2011), using human judges (Vyas and Pantel, 2009), to using relationships between semantic categories (Curran et al., 2007; Carlson et al., 2010).

Vyas et al. (2009) investigated the influence of seed instances on bootstrapping algorithms. They reported that seed instances selected by human who are not specialists sometimes yield worse results than those selected randomly. They proposed a method that refines seed sets generated by humans to improve the set expansion performance.

Bellare et al. (2007) proposed a method using a classifier instead of scoring functions in the ranking step of bootstrapping algorithms. The classifier approach can use multiple features to select instances. Sadamitsu et al. (2011) extended the method of Bellare et al. (2007) to use topic information estimated using Latent Dirichlet Allocation (LDA). They use not only contexts but also topic information as features of the classifier. Pennacchiotti and Pantel (2011) proposed a method for the automatic construction of training data for the classification approach. However, these researchers did not target set expansion for multiple semantic categories.

Vyas and Pantel (2009) proposed an algorithm that finds and removes the causes of semantic drift. The algorithm employs a human judge to prevent semantic drift in the iterative process of a bootstrapping algorithm. When a human judge detects an incorrect instance, the algorithm removes the patterns that acquired the incorrect instance. The algorithm also removes instances having a similar context vector to that of the incorrect instance to avoid a similar error. Although they used human judges, they ignored ontological information such as relations between categories.

Curran et al. (2007) proposed Mutual Exclusion Bootstrapping, which incorporates exclusiveness constraint between categories into the bootstrapping algorithm. Mutual Exclusion Bootstrapping uses the restriction that an instance and a pattern must belong to only one category. Instances or patterns appearing in multiple categories are ambiguous. Therefore, they are likely to cause semantic drift. By removing ambiguous instances and patterns, Curran et al. (2007) achieved high precision.

Carlson et al. (2010) proposed the Coupled Pattern Learner (CPL) algorithm which also uses mutual exclusiveness. The CPL algorithm acquires entity instances (e.g., instances of the car category) and relation instances (e.g., *CEO-of-Company (Larry Page, Google)* and *Company-acquired-Company (Google, Youtube)*) simultaneously. To acquire those instances, the algorithm requires knowledge about exclusiveness constraint between categories and links between categories (e.g., an instance of the CEO category must be CEO of some instance of the company category). However the algorithm uses only the exclusiveness constraint as prior knowledge related to multiple semantic categories.

Curran et al. (2007) and Carlson et al. (2010) use not only seed instances but also exclusiveness constraint between semantic categories as a prior knowledge. However, we have more prior knowledge about semantic categories at hand. For example, we can obtain ontological information between semantic categories easily from existing resources such as Wikipedia. Ontological information provides sibling relations between semantic categories, i.e., categories that should have common properties. In this study, we explore the usefulness of sibling relations between semantic categories in set expansion.

At last, we mention the relationship between this study and ontology learning. Ontology learning (Maedche and Staab, 2001; Navigli et al., 2003) is the task of constructing hierarchical structure of ontology by extracting terms and ontological relations between terms. In contrast, this study utilizes an existing resource as a hierarchical structure of ontology, and expands the list of instances of semantic categories on the hierarchical structure.

3 Proposed method

3.1 Filtering with patterns of sibling categories

In this section, we present the method using sibling relations between semantic categories as a prior knowledge. We gather categories that are siblings as a *sibling group*. For example, car and motorcycle categories belong to the same sibling group. We ex-

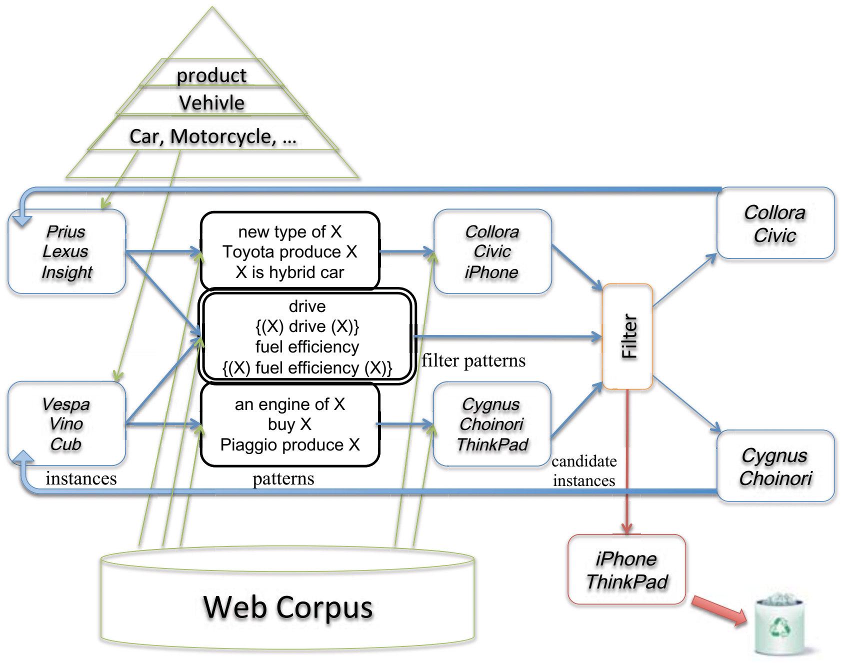

Figure 2: Set expansion using sibling relations.

Algorithm 1 The proposed method.

```
Input: $C$: a set of categories, $S_1, S_2, ..., S_T$: sibling groups (subset
    of C), $I_c$: seed instances of each $c \in C$, $L$: the number of itera-
    tions
Output: $I_c$: instances for each $c \in C$
 1: for $j = 1, 2, ..., T$ do
 2:     $F_{S_j} \leftarrow$ pattern-extraction($S_j$)
 3: end for
 4: for $l = 1, 2, 3..., L$ do
 5:     for $j = 1, 2, ..., T$ do
 6:         $I$ = []
 7:         for all $c \in S_j$ do
 8:             $R \leftarrow$ ESPRESSO($I_c$)
 9:             $R' \leftarrow$ FILTER($R, F_{S_j}$)
10:             $I$ += $R'$
11:         end for
12:         for $(i, c, s)$ in $I$ in descending order of score do
13:             if $|I_c| \leq N * l$ and $i \notin I_{c'}$ for all $c' \in S \setminus c$ then
14:                 insert $i$ into $I_c$
15:             end if
16:         end for
17:     end for
18: end for
19: function FILTER($R, F$)
20:     $R'$ = []
21:     for $(i, c, s)$ in R do
22:         if $i$ co-occur with $\forall f \in F$ then
23:             insert $(i, c, s)$ into $R'$
24:         end if
25:     end for
26:     return $R'$
27: end function
```

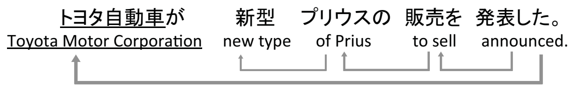

Figure 1: Example of syntactic dependencies.

pect that instances including the same sibling group hold common properties. We assume the common property to be represented by patterns of the sibling group. These patterns, which can check the common property of the sibling group, are denoted as *filter patterns*. Our proposed method obtains filter patterns using the sibling group and ascertains whether an instance co-occurs with the filter patterns.

Figure 2 presents examples of car and motorcycle categories included in the same sibling group. The method detects "drive" and "fuel efficiency" as filter patterns. Note that filter patterns are unconstrained by the difference of parent–child in the dependency tree. In the previous studies on bootstrapping, if the method obtains "new type of X" as the pattern of car category, then the method does not have a mechanism to reject incorrect instances such as "iPhone". In contrast, the proposed method ascertains whether each candidate instance co-occurs with the filter patterns before the final decision of acquiring instances. The method approves instances that co-occur with the filter patterns (e.g., "Corolla").

The detail of the method is described in Algorithm 1. The method is given a set of target categories C, sibling groups $(S_1, S_2, ..., S_T)$, seed instances I_c of each category $c \in C$ and the number of iterations L. Each sibling group is a subset of C, and disjoint from each other. The method chooses filter patterns F_{S_j} of each sibling group S_j from lines 1 to 3. In line 8, the method extracts instances of each category c of the sibling group S_j using the function ESPRESSO. ESPRESSO requires an instance set I_c of category c. ESPRESSO returns R, the list of the tuples each of which consists of instance i, category c and score s (i.e., (i, c, s)), using the Espresso algorithm described in Section 2.1.

In the experiments, using a Japanese large-scale corpus, we employ a phrase-like unit (*bunsetsu*) having dependency with an instance as a pattern. Figure 1 shows an example of Japanese sentence and its English translation[1]. Consider the instance "Toyota Motor Corporation" in the sentence shown in Figure 1. The algorithm extracts the pattern:

- X⟵ 発表した (X⟵announced)

In line 9, the method checks whether each candidate instance i in R has a common property of a sibling group using FILTER function (lines 19 to 27). FILTER examines that each i in R co-occurs with a filter pattern f in F. The function returns the list of instances and their scores which co-occur with the filter patterns. In short, this function filters out an instance which lacks the common property of the sibling group captured by the filter patterns F.

The method uses the exclusiveness constraint between categories of the sibling group to prevent drift within the group. If a pattern or an instance appears in multiple categories of the sibling group, then the method decides a single category that suits the best to the pattern or the instance. The method makes this decision based on a ranking. For example, consider a pattern "muffler of X", in which a pattern appeared

[1]The words in the English sentence are ordered as they appear in the Japanese sentence. For this reason, the word order and dependency edges in the English sentence look strange, but these are correct in the Japanese original sentence.

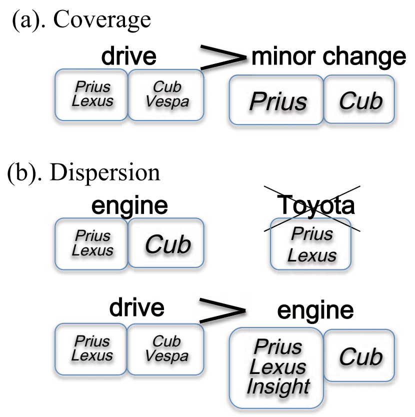

Figure 3: Two desirable properties for filter patterns.

in car and motorcycle categories. If the pattern is ranked 13th in the car category and fourth in the motorcycle category, then the pattern belongs only to the motorcycle category. In algorithm 1, function Espresso incorporates an exclusiveness constraint for patterns. The exclusiveness constraint for instances is implemented from lines 12 to 16.

The method acquires top N instances in order of score s while applying the exclusiveness constraint from lines 12 to 16. After the method secures new instances of each category, the method proceeds to the next iteration.

3.2 Acquisition of filter patterns

As described above, using filter patterns, our proposed method checks whether an instance has a common property of the sibling group. We describe how to extract and score the filter patterns.

Acquisition of filter patterns has two phases: candidate extraction and ranking. In candidate extraction, our method collects patterns that co-occur with seed instances of the sibling group. For example, given a sibling group consisting of car and motorcycle categories, the method finds patterns co-occurring with seed instances of car or motorcycle categories.

Filter patterns do not acquire instances of one sibling group but examine whether the instances have a common property of the sibling group. It is therefore unnecessary that filter patterns are of strict form to locate entities. For filter patterns, we disregard the difference of parent–child in the dependency tree.Please refer to *filter patterns (top 3)* column of Table 2 as an example of filter patterns.

After extracting candidates, the method selects the most suitable filter patterns in the candidates. The method selects filter patterns based on the two factors: *Coverage* and *Dispersion*. *Coverage* is the number of instances with which a filter pattern co-occurs. *Dispersion* is the degree of scattering categories in which the pattern appears. Figure 3 shows example of suitable filter patterns based on these factors. In Figure 3, a caption in bold font presents a filter pattern, and a caption in italic font presents an instance supported by the corresponding filter pattern.

The filter pattern is expected to cover all correct instances of the sibling group. Therefore, a pattern co-occurring with many seed instances is desirable. For example, in Figure 3 (a), the pattern "drive" is more suitable than "minor change" because "drive" supports more correct instances. This factor, *Coverage*, is modeled by recall. Recall of the pattern f of the sibling group S_j is calculated using equation (4).

$$Recall(S_j, f) = \frac{\sum_{c \in S_j} \sum_{i \in I_c} cooccur(f, i)}{\sum_{c \in S_j} |I_c|} \quad (4)$$

$$cooccur(f, i) = \begin{cases} 1 & \text{if } i \text{ co-occurs with } f \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

I_c is the set of seed instances of category c. $|I_c|$ is the number of seed instances of category c. $cooccur(f, i)$ indicates whether the seed instance i co-occurs with the pattern f. $\sum_{i \in I_c} cooccur(f, i)$ is the number of seed instances co-occurring with pattern f.

A filter pattern co-occurring with specific instances of the sibling group is inappropriate because filter patterns must ascertain whether candidate instances have a common property among categories. Therefore the method applies the restriction that the patterns must appear in two or more categories of the sibling group [2]. In Figure 3 (b), the pattern "engine" co-occurs with seed instances of both car and motorcycle categories but "Toyota" appears only in the car category. The method removes the latter in this example. Furthermore, we should respect a pattern which co-occurs with seed instances in each category in a sibling group equally. For example, Figure 3 (b) shows that the filter pattern "drive" is more suitable than "engine" because "drive" co-occurs with seed instances of car and motorcycle categories equally. This factor, *Dispersion*, is modeled by entropy. Entropy of the pattern f of the sibling group S_j is calculated using equation (6).

$$Entropy(S_j, f) = -\sum_{c \in S_j} P_c(f) \log_{|C|} P_c(f) \quad (6)$$

$$P_c(f) = \frac{\sum_{i \in I_c} cooccur(f, i)}{\sum_{c \in S_j} \sum_{i \in I_c} cooccur(f, i)} \quad (7)$$

$|C|$ is the number of categories in which the pattern f appears. If the pattern f co-occurs with seed instances of each category equally, then $Entropy(S_j, f)$ obtains the highest score.

To prioritize patterns with consideration of *Coverage* and *Dispersion*, we score the pattern f:

$$Score(S_j, f) = Entropy(S_j, f) * Recall(S_j, f) \quad (8)$$

Calculating $Score(S_j, f)$ for candidate pattern f of each sibling group S_j, the method acquires the top 15 patterns of each sibling group. We presume that a sibling group is exclusive to the other sibling groups. Therefore if a pattern is considered as candidates in multiple sibling groups, then the method decides that the pattern belongs to only the sibling group in which the pattern appears most frequently.

4 Experiments

4.1 Experimental setting

We evaluated the effect of sibling relations as a prior knowledge for set expansion. We compare the Espresso algorithm (Pantel and Pennacchiotti, 2006), the Espresso algorithm with exclusiveness constraint between categories (Espresso + exclusiveness constraint), and the Espresso algorithm with exclusiveness constraint and sibling relations (the proposed method). Each method was configured to extract 15 patterns and instances at every iteration. Because set expansion is the task to obtain unknown

[2] We found that the number of candidate patterns was too small when we adopted the restriction that filter patterns must appear in all categories of the sibling group. Therefore, we introduced the restriction that filter patterns must appear in two or more categories of the sibling group. Furthermore, we measure *Coverage* to obtain filter patterns that appear many categories of the sibling group.

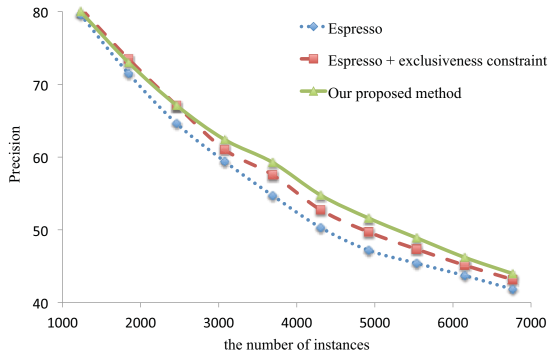

Figure 4: Precision of each method in accordance with the number of acquired instances.

instances, it is difficult to measure recall. Therefore, we compare the precision of each method when each method acquires fixed quantities of instances. We asked three human annotators to judge the correctness of acquired instances. We conducted the experiments in Japanese. The results described herein have been translated into English for presentation.

Table 2 reports all categories used for the experiments. Each category belongs to only one sibling group. Each sibling group consists of two or more categories. We prepared sibling groups by manually based on Wikipedia. Each category starts with 15 seed instances extracted from Wikipedia in a semi-automatic manner (Sumida et al., 2008). Because the automatic method yields incorrect seed instances, we removed errors manually.

We used 110 million Japanese web pages from which patterns and instances are extracted. We parsed sentences in the web pages using KNP, a Japanese dependency parser (Kurohashi et al., 1994). To reduce the computational time for the Espresso algorithm, we removed patterns and instances occurring fewer than three times.

4.2 Results

Figure 4 shows the precision of each method in accordance with the number of acquired instances. The dotted line depicts the precision curve of the Espresso algorithm. Espresso + exclusiveness constraint (depicted in dash line) improved the precision of extracted instances by 2.4 percents (with 4305 instances) and 1.3 percents (with 6765 instances). The proposed method (in solid line) with exclusive and sibling relations outperformed other baselines. In particular, the proposed method improved the precision from Espresso by 4.4 percents (with 4305 instances) and 2.1 percents (with 6765 instances). This result demonstrates that prior knowledge about sibling relations improves the performance of set expansion.

Table 1 shows the top 15 instances with high scores of Shinto shrine and temple categories, which belong to the same sibling group, acquired by each method in the 5th iteration. In Table 1, we divided instances into correct or incorrect ones. In Table 1, Espresso and Espresso + exclusiveness constraint obtained many incorrect instances, but each method acquired different instances. In Espresso, some instances (e.g., “Hachiman Shrine” and “Dazaifu Tenman-gu”) were identified as instances of both Shinto shrine and temple categories. In contrast, Espresso + exclusiveness constraint prohibits an instance from belonging to multiple categories and tries to choose the best category for the instance. This example suggests that mutual exclusivity mitigates semantic drift. However, in the temple category, Espresso + exclusiveness constraint obtains many unrelated instances such as “Fukuroya Soy Sauce Shop” and “Adashinomayu Village”. The proposed method removed these incorrect instances with knowledge about commonality of the sibling group. These results suggest that sibling relations are useful additional knowledge for set expansion.

Table 2 describes the precision of acquired instances of each category when each method has finished the 5th iteration. Table 2 also describes the improvement ratio of precision of the proposed method against Espresso. In Table 2, each line divides categories into a sibling group. Additionally, Table 2 exhibits the top three filter patterns used in each sibling group, along with their scores. Table 2 shows that the proposed method and Espresso + exclusiveness constraint improved the precision from Espresso in many categories. This fact indicates that sibling relation and exclusivity between categories improve the set expansion accuracy. However, in some categories, knowledge about sibling relations is ineffectual. We classify the possible causes of these failures into two types.

Table 1: Top 15 instances of the Shinto shrine and temple categories obtained by each method.

		Top 15 instances of each method		
category	correct/incorrect	Espresso	Espresso + exclusiveness constraint	The proposed method
Shinto shrine	correct	Hachiman Shrine, Dazaifu Tenman-gu, Meiji Shrine, Tenman-gu, Tsurugaoka Hachiman-gu, Ise Grand Shrine, Yasaka Shrine, Kasuga Shrine, Izummo Shrine, Yasukuni Shrine, Kanda Shrine, Shinto Shrine	Dazaifu Tenman-gu, Meiji Shrine, Hachiman Shrine, Ise Grand Shrine, Tenman-gu, Izumo Shrine, Tsurugaoka Hachiman-gu, Kasuga Shrine, Yasaka Shrine, Yasukuni Shrine, Kanda Shrine, Atago Shrine, Shinto Shrine	Meiji Shrine, Ise Grand Shrine, Dazaifu Tenman-gu, Hachiman-gu, Tsurugaoka Hachiman-gu, Izumo Shrine, Yasaka Shrine, Kasuga Shrine, Tenman-gu, Yasukuni Shrine, Itsukushima Shrine, Kanda Shrine, Atago Shrine, Shinto Shrine
	incorrect	Senso-ji, Narita Mountain, Temple	Narita Mountain, Senso-ji	Narita Mountain
temple	correct	Senso-ji, Zenko-ji, Narita Mountain, Temple	Jio-ji, Tokurin-an	Nanzen-ji, Daitoku-ji, Chion-in, Myoshin-ji, Rokuharamitsu-ji, Shokoku-ji, Jojako-ji, Sekizanzen-in, Raige-in, Konzo-ji, Temple, Temple
	incorrect	Shinto Shrine, Hachiman Shrine, Dazaifu Tenman-gu, Tenman-gu, Tsurugaoka Hachiman-gu, Meiji shrine, Yasaka Shrine, Kasuga Shrine, Ise Grand Shrine, Izumo Shrine, Yasukuni Shrine	Konoshimanimasu Amaterumitama Shrine, Kohata Shrine, Kyoto Prefectural Insho-Domoto Museum of Fine Arts, Adashinomayu Village, Uji-Kanbayashi Musium, Fukuroya Soy Sauce Shop, Kyoto Orthodox Church, Konjyakunishimura, Ichiharaheibei Shop, Kungyokudo, Catholic Miyazu Parish, Ise Bay Tour Boat, Lake Biwa Canal Memorial	Shimogamo Shrine, Imamiya Shrine, Hirano Shrine

1. Low score of filter patterns

2. High precision in the baseline

In cause 1, precision drops in categories (e.g., motor vehicle manufacture, medical supplies manufacture, art museum, and theater categories) of some sibling groups. For example, in motor vehicle manufacture and medical supplies manufacture categories, improvement ratios are -14.44 percent and -2.22 percent, respectively. In this sibling group, the highest score of filter patterns is as low as 0.1837. Recall that scores of filter patterns are computed for a small number of seed instances in a sibling group. Low scores of filter patterns imply that the proposed method could not find patterns with of *Coverage* and *Dispersion*. We suspect that the lack of commonality of categories in the sibling group (e.g., motor vehicle manufacture and medical supplies manufacture) does not fit well to the assumption of using filter patterns. Therefore, investigating the impact of choosing sibling groups would be an interesting future direction of this research.

The film director and the comedian categories are affected by cause 2. In cause 2, although semantic drift does not occur in Espresso, the proposed method filtered out some positive instances. In other words, the proposed method removed correct instances more than incorrect ones. The proposed method forces instances to co-occur with filter patterns but the constraint may be too strong. Moreover, because filter patterns are determined only by seed instances, filter patterns may not cover the common property of newly-acquired instances as the method iterates the bootstrapping process. In order to remedy this effect, it might be necessary to update filter patterns in the middle of iterations.

5 Conclusion

In this paper, we demonstrated that sibling relations between semantic categories provide useful knowledge for set expansion. We proposed the method that uses sibling relations as prior knowledge. In the experiments, we reported that the proposed method gained 4.4 percent improvements (with 4305 instances) from the baseline Espresso algorithm.

However, as explained in Section 4.2, the proposed method suffers from some side effects. We suspect that the causes of the side effects derive from the design of sibling groups and the constant use of filter patterns. Addressing these issues is left for future work. We also plan to extend this approach for extracting relation instances where each relation has semantic constraints on arguments (entity instances) and where pairwise relations (e.g., *is-president-of* and *is-citizen-of*) also have hierarchy (e.g., entailment and causality relations).

Table 2: Precision for each category and each method after five iterations.

category	Espresso (%)	Espresso + exclusiveness constraint(%)	The proposed method(%)	improvement ratio(%)	filter patterns (top 3)	pattern score
Shinto shrine	72.22	73.33	75.56	3.33	precicnct	0.9658
					plum	0.5946
temple	14.44	37.78	63.33	48.89	*hatsumode*	0.5266
Japanese city	97.78	97.78	100.00	2.22	live	1.0000
American city	28.89	34.44	36.67	7.78	go	1.0000
Chinese city	37.78	58.89	60.00	22.22	leave	0.9319
infection	26.67	47.78	47.78	21.11	illness	1.0000
					treatment	0.9658
mental illness	34.44	46.67	46.67	12.22	symptom	0.9658
film director	97.78	97.78	74.44	-23.33	original work	0.6235
cartoonist	87.78	86.67	91.11	3.33	masterpiece	0.5832
novelist	95.56	93.33	94.44	-1.11	best work	0.3167
car	95.56	95.56	95.56	0.00	drive	0.7572
					own car	0.5510
motorcycle	83.33	83.33	93.33	10.00	you	0.5409
board game	24.44	23.33	22.22	-2.22	play	0.9092
computer game	98.89	98.89	98.89	0.00	game	0.8651
card game	63.33	61.11	61.11	-2.22	enjoy	0.4434
motor vehicle manufacture	62.22	62.22	47.78	-14.44	stock quote	0.1837
					Takeda Pharmaceutical	0.1333
medical supplies manufacture	5.56	5.56	3.33	-2.22	meeting	0.1333
Asian country	34.44	33.33	33.33	-1.11	Japan	1.0000
African country	44.44	45.56	45.56	1.11	nation	1.0000
European country	44.44	48.89	48.89	4.44	speak	1.0000
art museum	35.56	37.78	17.78	-17.78	outward appearance	0.1618
					front yard	0.1203
theater	50.00	50.00	24.44	-25.56	close	0.1082
island	66.67	66.67	61.11	-5.56	flow	0.5757
mountain	96.67	96.67	92.22	-4.44	fish	0.4412
river	95.56	95.56	95.56	0.00	sea	0.3749
radio station	20.00	61.11	61.11	41.11	program	0.9658
					broadcasting	0.8630
TV station	68.89	67.78	67.78	-1.11	announcer	0.8630
station	98.89	100.00	100.00	1.11	get off	0.6989
					near	0.6042
airport	37.78	37.78	44.44	6.67	Haneda Airpoat	0.5090
chemical element	20.00	20.00	20.00	0.00	contain	1.0000
					quantity	0.9299
chemical combination	41.11	41.11	41.11	0.00	component	0.8518
lake	31.11	21.11	21.11	-10.00	beside	0.8518
					command	0.7146
pool	2.22	0.00	0.00	-2.22	cape	0.6618
actor	95.56	94.44	94.44	-1.11	picture	0.8920
					movie	0.8518
comedian	98.89	98.89	97.78	-1.11	perform	0.8324
bacterium	37.78	45.56	40.00	2.22	bacterium	0.4585
					bacillus	0.4460
virus	32.22	28.89	14.44	-17.78	microbe	0.4183
news paper	48.89	48.89	53.33	4.44	publish	0.8920
					article	0.7635
magazine	44.44	44.44	96.67	52.22	print	0.5698
publisher	32.22	23.33	97.78	65.56	publisher	0.6473
					familiar	0.2014
record company	38.89	47.78	48.89	10.00	manufacture	0.1656

Acknowledgments

This research was partly supported by JST, PRESTO. This research was partly supported by JSPS KAKENHI Grant Numbers 23240018 and 23700159.

References

Steven Abney. 2004. Understanding the yarowsky algorithm. *Comput. Linguist.*, 30(3):365–395.

Kedar Bellare, Partha Pratim Talukdar, Giridhar Kumaran, O Pereira, Mark Liberman, Andrew Mccallum, and Mark Dredze. 2007. Lightlysupervised attribute extraction for web search. In *Proceedings of Machine Learning for Web Search Workshop, NIPS 2007*.

Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 875–883.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *Pacific Association for Computational Linguistics*, pages 172–180.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545.

Sadao Kurohashi, Sadao Kurohashi, and Makoto Nagao. 1994. Kn parser : Japanese dependency/case structure analyzer. In *In Proceedings of the Workshop on Sharable Natural Language Resources*, pages 48–55.

Alexander Maedche and Steffen Staab. 2001. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79.

Roberto Navigli, Paola Velardi, Universit Roma, and La Sapienza. 2003. Ontology learning and its application to automated terminology translation. *IEEE Intelligent Systems*, 18(1):22–31.

Etzioni Oren, Banko Michele, and Cafarella Michael J. 2006. Machine reading. In *Proceedings of The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference*.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 113–120.

Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *Proceedings of Human Language Technology/North American chapter of the Association for Computational Linguistics*, pages 321–328.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 938–947.

Marco Pennacchiotti and Patrick Pantel. 2011. Automatically building training examples for entity extraction. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 163–171.

Kugatsu Sadamitsu, Kuniko Saito, Kenji Imamura, and Genichiro Kikui. 2011. Entity set expansion using topic information. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers*, pages 726–731.

Asuka Sumida, Naoki Yoshinaga, and Kentaro Torisawa. 2008. Boosting precision and recall of hyponymy relation acquisition from hierarchical layouts in wikipedia. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*.

Vishnu Vyas and Patrick Pantel. 2009. Semi-automatic entity set refinement. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 290–298.

Vishnu Vyas, Patrick Pantel, and Eric Crestan. 2009. Helping editors choose better seed sets for entity set expansion. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 225–234.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196.