Automatic Set Instance Extraction using the Web

Richard C. Wang
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213 USA
rcwang@cs.cmu.edu

William W. Cohen
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213 USA
wcohen@cs.cmu.edu

ABSTRACT

An important and well-studied problem is the production of semantic lexicons from a large corpus. In this paper, we present a system named ASIE (Automatic Set Instance Extractor), which takes in the name of a semantic class as input (e.g., fruits) and automatically outputs its instances (e.g., orange, apple). ASIE is based on recent advances in *set expansion* - the problem of finding all instances of a set given a small number of "seed" instances. This approach effectively exploits web resources and can be easily adapted to different languages. In brief, our approach works by using simple language-dependent hyponym patterns to find a noisy set of initial seeds, and then using a state-of-the-art language-independent set expansion system to extend and rank these initial seeds. The proposed approach matches or outperforms prior systems on three English-language benchmarks. It also shows excellent performance on three dozen additional benchmark problems in English, Chinese and Japanese.

1. INTRODUCTION

An important and well-studied problem is the production of semantic lexicons for semantic classes of interest - i.e., generation of all instances of a set (e.g., apple, banana, orange, etc.) given a description of that set (e.g., fruits). This task is often addressed by analysis of large corpora [2, 4, 1, 8, 9].

In this paper, we evaluate a novel approach to this problem by using a system called ASIE[1] (Automatic Set Instance Extractor), which takes in a semantic class name as input (e.g., fruits) and automatically outputs its instances (e.g., apple, banana, orange, etc.). This system is based on recent advances in *set expansion* - the problem of finding all instances of a set given a small number of example (seed) instances. In recent work, we have developed a set expansion system called SEAL [11], which is a language-independent web-based system that performed extremely well on a large number of benchmark sets. Given three correct seeds, SEAL obtained average MAP scores in the high 90's for 36 benchmark problems, including a dozen test problems each for English, Chinese and Japanese. SEAL works well in part because it can very efficiently find and process *semi-structured web documents* containing instances of the set being expanded. Figure 1 shows some examples of SEAL's input and output.

		English	Chinese	Japanese
input	seed	Amazing Race	豬血糕	ドラえもん
		Survivor	臭豆腐	ハローキティ
output		Big Brother	蘿蔔糕	アンパンマン
		The Mole	蚵仔煎	シナモロール
		The Apprentice	肉圓	ウルトラマン
		Project Runway	滷肉飯	スヌーピー
		The Bachelor	春捲	くまのプーさん

Figure 1: Examples of SEAL's input and output. English entities are reality TV shows, Chinese entities are popular Taiwanese foods, and Japanese entities are famous cartoon characters.

In more recent work, SEAL has been extended to be robust to errors in its initial set of seeds [13] and to bootstrap performance by using only two initial seeds [12]. These extensions suggest the following approach to extract instances of sets from the Web. First, given a semantic class name (e.g., fruits), use simple language-dependent *hyponym patterns* (e.g., fruits such as *blah*) to find a large but noisy set of seed instances. Second, use the extended version of SEAL to expand and bootstrap this noisy set of seeds.

This approach has two advantages. First, it is nearly language-independent: all that is needed to support a new target language is a new set of hyponym patterns for that language. In this paper, we present experimental results for Chinese and Japanese, as well as English. Second, the system performs extremely well experimentally. In our evaluation we focus on the 36 benchmarks used in SEAL, which are relatively small closed sets (e.g., countries, constellations, NBA teams) that are fairly well-represented on the Web. Using these sets enables rigorous evaluation of both the recall and precision of the system, and we show that for these benchmarks, SEAL has excellent performance for both recall and precision. We also compare directly the recent work of Kozareva, Riloff, and Hovy [4] on three additional English-language benchmark problems (US states, countries, and common fish) and show comparable or better performance on each of these problems.

Section 2 of this paper describes our proposed approach that ASIE is based on. Section 3 presents the details of the experimental setting. Experimental results are presented in Section 4, and related work is discussed in Section 5. The

[1] `http://rcwang.com/asie`

WWW2009, April 20-24, 2009, Madrid, Spain.

paper concludes in Section 6 with a discussion of planned future work.

2. PROPOSED APPROACH

ASIE is composed of three main components: Noisy Instance Generator, Reranker, and Bootstrapper. The Noisy Instance Generator extracts a set of candidate instances given a semantic class name, and ranks the instances by using a simple ranking model. The Reranker re-ranks the instances using evidence from semi-structured web documents such that noisy (irrelevant) ones are ranked lower in the list. The Bootstrapper enhances the quality and completeness of the ranked list by using an unsupervised iterative technique. Note that the Reranker and Bootstrapper rely on SEAL to accomplish their goals. In this section, we describe the Generator, the Set Expander (SEAL), the Reranker, and finally, the Bootstrapper.

2.1 Noisy Instance Generator

Noisy Instance Generator extracts candidate instances using the methods presented in Hearst's early work [2]. Hearst exploited several patterns for identifying hyponymy relation (e.g., such author as Shakespeare) that many current state-of-the-art systems [4, 8, 1, 9] are using. However, unlike all of those systems, ASIE does not use any NLP tool (e.g., parts-of-speech tagger, parser) or rely on capitalization for extracting candidates (since we wanted ASIE to be as language-independent as possible).

The Generator first constructs a few queries of hyponym phrase by using a semantic class name and a set of predefined hyponym patterns. For every query, the Generator retrieves a hundred snippets from the search engine, and splits each snippet into multiple excerpts (a snippet often contains multiple excerpts from its web page). For each excerpt, the Generator extracts all chunks of characters that would then be used as candidate instances. Here, we define a *chunk* as a sequence of characters bounded by punctuation marks or the beginning and end of an excerpt. Lastly, the Generator ranks each candidate instance x based on its weight assigned by the simple ranking model presented below:

$$P(x|\mathbf{P}) = \frac{sf(x|\mathbf{S})}{|\mathbf{S}|} \times \frac{ef(x|\mathbf{E})}{|\mathbf{E}|} \times \frac{wcf(x|\mathbf{E})}{|\mathbf{C}|}$$

where $\mathbf{P}$ is the set of hyponym patterns, $\mathbf{S}$ is the set of snippets, and $\mathbf{E}$ is the set of excerpts. $sf(x|\mathbf{S})$ is the snippet frequency of x (i.e. the number of snippets containing x) and $ef(x|\mathbf{E})$ is the excerpt frequency of x. Furthermore, $wcf(x|\mathbf{E})$ is the weighted chunk frequency of x, which is defined as follows:

$$wcf(x|\mathbf{E}) = \sum_{e \in \mathbf{E}} \sum_{x \in e} \frac{1}{dist(x|e) + 1}$$

where $dist(x|e)$ is the number of characters between x and the hyponym phrase in excerpt e. This model weights every occurrence of x based on the assumption that chunks closer to a hyponym phrase are usually more important than those further away. We want to emphasize that we are not trying to present a state-of-the-art model for ranking candidates, since the Reranker will be *reranking* those candidates. Any other similar instance-ranking model can be used in this step.

English	Chinese	Japanese
<C> such as <I>	<C> 例如 <I>	<C> 例えば <I>
such <C> as <I>	<C> 比如 <I>	<C> たとえば <I>
<C> i.e. <I>	<C> 如 <I>	<C> と言えば <I>
<C> e.g. <I>	<C> 包含 <I>	<C> といえば <I>
<C> include <I>	<C> 包括 <I>	<I> 等の <C>
<C> including <I>	<I> 等 <C>	<I> などの <C>
<C> like <I>		<I> とかの <C>
<I> and other <C>		
<I> or other <C>		

Figure 2: Hyponym patterns in English, Chinese, and Japanese. In each pattern, <C> is a placeholder for the semantic class name and <I> is a placeholder for its instances.

Figure 2 shows the hyponym patterns we use for English, Chinese, and Japanese. There are two types of hyponym patterns: The first type are the ones that require the class name C to precede its instance I (e.g., C such as I), and the second type are those that require the class name to follow its instance (e.g., I and other C). In order to reduce irrelevant chunks, when excerpts were extracted, the Generator drops all characters preceding the hyponym phrase in excerpts that contain the first type. Similarly, it drops all characters following the hyponym phrase in excerpts containing the second type. Occasionally, for some semantic class name (e.g., "cmu buildings"), there are no web documents containing any of the hyponym phrases that were constructed using the name. In this case, the Generator turns to a *back-off* strategy which simply treats the semantic class name as the hyponym phrase and extracts/ranks all chunks co-occurring with the phrase in the excerpts.

2.2 Set Expander - SEAL

In this paper, we rely on a set expansion system named SEAL[2] [11], which stands for Set Expander for Any Language. The system accepts input element *seeds* of some target set S (e.g., fruits) and automatically finds other probable instances (e.g., apple, banana, orange) of S in web documents. As its name implies, SEAL is independent of document languages: both the written (e.g., English) and the markup language (e.g., HTML). SEAL is a research system that has shown good performance in previously published results [11, 13, 12]. Figure 1 shows some examples of SEAL's input and output.

In more detail, SEAL also contains three major components: the Fetcher, Extractor, and Ranker. The *Fetcher* is responsible for fetching web documents, and the URLs of the documents come from top results retrieved from the search engine using the concatenation of all seeds as the query. This ensures that every fetched web page contains all seeds. The *Extractor* automatically constructs "wrappers" (i.e. page-specific extraction rules) for each page that contains the seeds. Every wrapper comprises two character strings that specify the left and right contexts necessary for extracting candidate instances. These contextual strings are maximally-long contexts that bracket at least one occurrence of every seed string on a page. All other candidate instances bracketed by these contextual strings derived from a particular page are extracted from the same page.

[2] `http://rcwang.com/seal`

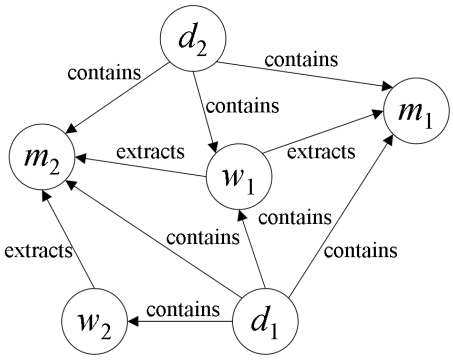

Figure 3: An example graph constructed by SEAL. Every edge from node x to y actually has an inverse relation edge from node y to x that is not shown here (e.g., m_1 *is extracted by* w_1).

In practice, the Extractor often extracts most of its candidate instances from semi-structured data pages found on the web. The character-level heuristics used to find "wrappers" often find candidate instances in parts of a web page that are not even easily visible to the end user (e.g., in pull-down menus of an HTML form).

After the candidates are extracted, the *Ranker* constructs a graph that models all the relations between documents, wrappers, and candidate instances. Figure 3 shows an example graph where each node d_i represents a document, w_i a wrapper, and m_i a candidate instance. The Ranker performs Random Walk with Restart [10] on this graph (where the initial "restart" set is the set of seeds) until all node weights converge, and then ranks nodes by their final score; thus nodes are weighted higher if they are connected by many short, low fan-out paths in the graph to many seed nodes. The final expanded set contains all candidate instance nodes, ranked by their weights in the graph.

2.3 Reranker

Previous work [13] illustrated that it is feasible to perform set expansion on noisy input seeds. The paper showed that the noisy output of any Question Answering system for list questions can be improved by using a noise-resistant version of SEAL (An example of a list question is "Who were the husbands of Heddy Lamar?"). Since the candidate instances obtained using Hearst's method are noisy, the Reranker ranks the candidates by applying the noise-resistant SEAL, which pushes irrelevant candidates lower in the ranked list by adding more relevant candidates and raising existing relevant candidates higher in the list.

There are three main differences between the original SEAL (described in previous section) and the noise-resistant SEAL. The first difference is the Fetcher. In the original SEAL, the Fetcher concatenates all seeds and sends them as one query to the search engine. However, when the seeds are noisy, the documents fetched are restricted by the irrelevant seeds, which decreases the chance of retrieving relevant documents. The Fetcher in noise-resistant SEAL overcomes this problem by sending a *two-seed* query for every possible pair of seeds to the search engine. Thus, if there are n input seeds, then the total number of queries sent would be $\binom{n}{2}$. This increases the chance of composing queries containing *only* relevant seeds; however, it also increases the number of hits to the search engine.

The second difference is the Extractor. In the original SEAL, the extractor requires the longest common contexts to bracket at least one instance of every seed per web document. However, when seeds are noisy, such common contexts usually do not exist or are too short to be useful. The Extractor in noise-resistant SEAL solves this problem by requiring the contexts to bracket at least one instance of *a minimum of two* seeds, rather than *every* seed. This increases the chance of finding longest common contexts that bracket *only* relevant seeds.

The third difference is the use of *hint words* when querying search engine. In the original SEAL, if the seeds are *George Washington*, *John Adams*, and *Thomas Jefferson*, then it would output a mixture of founding fathers and presidents of the United States. The noise-resistant SEAL allows hint words (e.g., founding fathers, presidents) to be appended to every query that is sent to the search engine. This increases the chance of retrieving documents that contain our desired set of answers.

Here, we want to make a distinction between *page seeds* and *wrapper seeds*. The first one is a set of seeds used as query for retrieving web documents from the search engine, and the latter one is for constructing wrappers from web documents. In previous published work [13], the same set of seeds (i.e. top four candidates) have been used for both page and wrapper seeds. However, we observed that the candidate instances extracted by the Noisy Instance Generator are extremely noisy; thus, we decide to increase the size of wrapper seeds to 15. We refrain from increasing the size of page seeds since it will significantly increase the number of hits to the search engine (we have limited quota per day).

2.4 Bootstrapper

Bootstrapping is an (unsupervised) iterative process in which a system continuously consumes its own outputs to improve its own performance [1, 3, 5]. The advantage of bootstrapping is that, if successful, the performance of a system can be greatly improved with minimal supervision. Previous work [12] showed that it is feasible to bootstrap the results of set expansion to improve the quality of a list. The paper introduces an iterative version of SEAL, which expands a list in multiple iterations. In each iteration, iterative SEAL expands top-ranked candidates extracted in previous iterations and aggregates statistics by growing the graph. The Bootstrapper utilizes this iterative SEAL to further improve the quality of the list ranked by the Reranker.

The Bootstrapper requires only two initial seeds, which are used to trigger the first expansion of the iterative process above. For every i^{th} expansion, it expands $min(4, i+1)$ *unsupervised seeds* (i.e. highly ranked items obtained in previous iterations) while aggregating statistics from iteration to iteration. We set the maximum number of seeds to four because it has been shown to maximize the set expansion performance [1, 5, 12]. The pseudo-code for this strategy is presented below:

$used \leftarrow \emptyset$
for $i = 1$ to M **do**
  if $i = 1$ **then**
    $seeds \leftarrow select_2(E)$
  else
    $m = min(3, |used|)$
    $seeds \leftarrow select_m(used) \cup top_1(ranked\_list)$

end if
$used \leftarrow used \cup seeds$
$ranked\_list \leftarrow expand(seeds)$
end for

where M is the total number of iterations (inclusively), E is a set containing initial seeds, $select_n(S)$ randomly selects n different seeds from the set S, $top_1(ranked\_list)$ returns a new candidate that has the highest weight in $ranked\_list$, and $expand(seeds)$ expands the selected $seeds$ and produces a $ranked\_list$ of candidates.

3. EXPERIMENTAL SETTING

3.1 Datasets

We evaluate the quality of the extracted instances using an updated version[3] of the datasets presented in Wang and Cohen [11]. The original version consists of 36 manually constructed lists across three different languages: English, Chinese, and Japanese (12 lists per language). Each list contains all instances of a particular semantic class in a certain language, and each instance is represented by a set of synonyms (e.g., USA, America). The updated version includes more synonyms plus an additional English list of common fish names obtained from the Wikipedia. In this dataset of 37 lists, there are a total of 3637 instances and 6080 synonyms, with an average of 98 entities per class and 1.7 synonyms per instance. Figure 4 shows the datasets and their corresponding semantic class names used in our experiments.

3.2 Evaluation Metric

Since the output of ASIE is a ranked list of extracted instances, we choose mean average precision (MAP) as our evaluation metric. MAP is commonly used in the field of Information Retrieval for evaluating ranked lists because it is sensitive to the entire ranking and it contains both recall and precision-oriented aspects. The MAP for multiple ranked lists is simply the mean value of average precisions calculated separately for each ranked list. We define the average precision of a single ranked list as:

$$AvgPrec(L) = \frac{\sum_{r=1}^{|L|} \text{Prec}(r) \times \text{isFresh}(r)}{\text{Total \# of Correct Instances}}$$

where L is a ranked list of extracted instances, r is the rank ranging from 1 to $|L|$, Prec(r) is the precision at rank r, or the percentage of correct synonyms above rank r (inclusively). isFresh(r) is a binary function for ensuring that, if a list contains multiple synonyms of the same instance, we do not evaluate that instance more than once. More specifically, the function returns 1 if a) the synonym at r is correct, and b) it is the highest-ranked synonym of its instance in the list; it returns 0 otherwise.

3.3 Evaluation Method

In our experiments, we use Yahoo! as the search engine for querying web documents. For each semantic class in our dataset, the Noisy Instance Generator first produces a noisy list of candidate instances using the corresponding class name shown in Figure 4. The Reranker then re-ranks this noisy list *twice* to maximize the quality of the top-ranked candidates. For the first re-ranking, the Reranker retrieves one hundred web documents using the semantic class name as the only query to the search engine, and re-ranks the noisy list using those documents. For the second re-ranking, the Reranker sends a two-seed query to the search engine for every possible pair of top four candidates in the noisy list (thus, a total of six queries). For each query, the Reranker retrieves only ten web documents (thus, a total of 60 documents for all queries). In the second re-ranking, the semantic class name of the candidates were used as hint words to increase the chance of finding relevant documents. The Bootstrapper then conducts 20 iterations of bootstrapping using the top two candidates from the output of the Reranker as initial seeds. In each of the 20 iterations, the Bootstrapper retrieves, again, only ten web documents (thus, a total of 200 documents). Note that for each semantic class, ASIE utilizes a maximum of only 360 (100+60+200) web documents.

[3]Available at `http://rcwang.com/papers/dataset/seal-data2.zip`

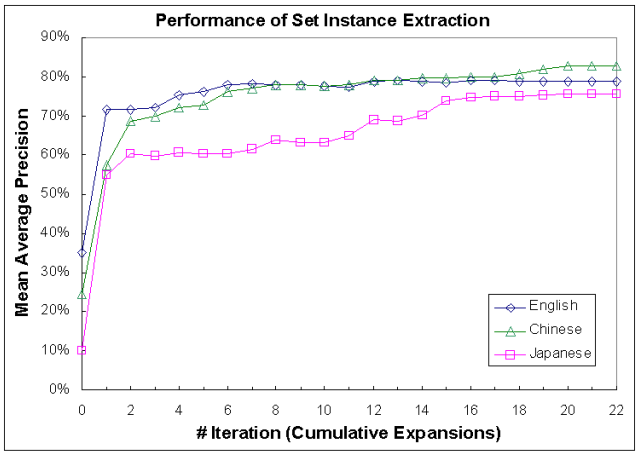

Figure 5: MAP of extracted instances in English, Chinese, and Japanese. Note that the MAP at iteration 0 is the performance of Noisy Instance Generator, at 1 and 2 are the performance of the Reranker, and the rest of the 20 iterations are the performance of the Bootstrapper.

4. EXPERIMENTAL RESULTS

Figure 5 shows the MAP of the set instances extracted by our system ASIE. The result shown at the initial iteration is the performance of the Noisy Instance Generator, followed by the performance of the Reranker (for two iterations), which is then followed by the performance of the Bootstrapper (for 20 iterations). As illustrated, the Reranker substantially improves the quality of the initial noisy list of candidate instances, and the Bootstrapper then improves the quality of this list even more.

Table 1 shows a more detailed view of this graph. As shown, the Reranker improves the performance of the Generator from 35% to 72% for English, 24% to 69% for Chinese, and 10% to 60% for Japanese. The Bootstrapper then further improves the performance of the Reranker from 72% to

English		Chinese		Japanese	
Dataset	Class Name	Dataset	Class Name	Dataset	Class Name
classic-disney	classic disney movies	classic-disney	迪士尼動畫	classic-disney	ディズニー映画
constellations	constellations	constellations	星座	constellations	星座
countries	countries	countries	國家	countries	国
mlb-teams	mlb teams	mlb-teams	mlb 球隊	mlb-teams	mlb チーム
nba-teams	nba teams	nba-teams	nba 球隊	nba-teams	nba チーム
nfl-teams	nfl teams	nfl-teams	nfl 球隊	nfl-teams	nfl チーム
popular-car-makers	car makers	popular-car-makers	車廠	popular-car-makers	自動車メーカー
us-presidents	us presidents	us-presidents	美國總統	us-presidents	アメリカ大統領
us-states	us states	us-states	州	us-states	アメリカの州
cmu-buildings	cmu buildings	china-dynasties	朝代	japan-emperors	天皇
common-diseases	common diseases	china-provinces	省	japan-prime-ministers	総理大臣
periodic-comets	comets	taiwan-cities	縣市	japan-provinces	県
common-fish	common fish				

Figure 4: Semantic class names used in our experiments.

English	NIG	RR	BS	Chinese	NIG	RR	BS	Japanese	NIG	RR	BS
classic-disney	20.9	74.1	84.6	classic-disney	10.3	72.2	85.3	classic-disney	19.2	36.1	70.0
constellations	30.9	98.5	100.0	constellations	11.3	81.5	90.2	constellations	18.4	26.2	100.0
countries	58.7	95.4	97.9	countries	32.3	41.3	77.7	countries	18.0	89.8	95.7
mlb-teams	63.3	100.0	100.0	mlb-teams*	4.9	98.8	100.0	mlb-teams*	0.3	6.2	8.0
nba-teams	53.1	100.0	100.0	nba-teams	19.0	62.6	96.8	nba-teams*	1.0	0.0	0.1
nfl-teams	54.6	100.0	100.0	nfl-teams*	0.0	0.0	0.1	nfl-teams*	0.9	63.7	73.7
popular-car-makers	37.6	95.6	97.4	popular-car-makers	12.4	43.3	96.4	popular-car-makers	21.4	64.1	86.8
us-presidents	40.3	100.0	100.0	us-presidents	17.8	77.4	91.5	us-presidents	7.8	88.5	89.9
us-states	79.5	96.0	100.0	us-states	68.8	98.9	100.0	us-states	4.8	99.8	100.0
cmu-buildings*	0.0	0.0	0.0	china-dynasties	8.3	53.5	57.2	japan-emperors	3.0	48.9	93.0
common-diseases	11.8	33.4	42.0	china-provinces	48.8	96.6	100.0	japan-prime-ministers	10.3	99.7	99.8
periodic-comets	1.3	2.0	0.7	taiwan-cities	57.7	99.7	100.0	japan-provinces	15.4	100.0	100.0
common-fish	4.0	37.2	93.4								
Average	35.1	71.7	78.2	Average	24.3	68.8	82.9	Average	10.0	60.3	76.4

Table 1: Performance of set instance extraction for each dataset measured in MAP (%). NIG is the Noisy Instance Generator, RR is the Reranker, and BS is the Bootstrapper. The candidate instances for datasets marked with * were extracted using the *back-off* strategy since no web documents containing their hyponym phrases could be found.

78% for English, 69% to 83% for Chinese, and 60% to 76% for Japanese. In addition, the simple *back-off* strategy seems to be effective as well. There are six datasets (marked with *) of which their hyponym phrases return zero web documents. For those datasets, ASIE automatically uses the *back-off* strategy described in Section 2.1. Considering only those six datasets, the Reranker, on average, improves the performance of the Generator from 1.2% to 28.1% and the Bootstrapper then improves it to 30.3%.

Table 2 shows a comparison of our extraction performance to that of Kozareva *et al* [4]. The size of their datasets for US states, countries, and fish are 50, 194, and 1102 respectively; whereas, in our datasets, they are 50, 200, and 1120 respectively. The results show that the quality of our extracted instances is comparable to theirs. Note that the input to their system is a semantic class name *plus* one seed instance; whereas the input to ASIE is only the class name. In terms of system runtime, for each semantic class, Kozareva *et al* reported that their extraction process usually finished overnight; whereas ASIE usually finished within 3 minutes.

We also compare our results to Pasca [7] as shown in Table 3. There are 197 countries in his evaluation dataset, and the input to his system is a set of seed named entities. As illustrated, ASIE outperforms his system in terms of extracting country names using the semantic class name "countries" as the only input to our system.

5. RELATED WORK

There has been a significant amount of research done in the area of semantic class learning (aka lexical acquisition, lexicon induction, hyponym extraction, or open-domain information extraction). However, to the best of our knowledge, there is not a system that can perform set instance extraction in multiple languages given only the *name* of the set.

Pantel *et al* [8] presented an algorithm for automatically inducing names for semantic classes and for finding their instances by using concept signatures. Pasca [9] presented a method for acquiring named entities in arbitrary categories using lexico-syntatic extraction patterns. Etzioni *et al* [1] presented the KnowItAll system that also utilizes hyponym patterns to extract class instances from the Web. Note that all the systems mentioned require the use of parts-of-speech tagger and/or parser.

Hearst [2] presented an approach that utilizes hyponym patterns for extracting candidate instances given the name of a semantic set. The approach presented in Section 2.1 is based on this work, except that we extended it to other lan-

N	Kozareva	ASIE
US States		
25	100% (25)	100% (25)
50	100% (50)	100% (50)
64	78% (50)	78% (50)
Countries		
50	100% (50)	100% (50)
100	100% (100)	99% (99)
150	100% (150)	99% (149)
200	90% (180)	93% (186)
300	61% (183)	66% (198)
323	57% (184)	62% (200)
Common Fish		
10	100% (10)	100% (10)
25	100% (25)	100% (25)
50	100% (50)	100% (50)
75	93% (70)	97% (73)
100	84% (84)	97% (97)
116	80% (93)	97% (113)

Table 2: Set instance extraction performance comparison. At various rank N, we report the precision (and the number of correct instances above N).

N	Pasca	ASIE
Countries		
25	100% (25)	100% (25)
50	98% (49)	98% (49)
100	95% (95)	99% (99)
150	82% (123)	93% (140)
250	60% (150)	66% (165)

Table 3: Set instance extraction performance comparison. At various rank N, we report the precision (and the number of correct instances above N).

guages: Chinese and Japanese. Kozareva *et al* [4] illustrated an approach that uses a single hyponym pattern combined with graph structures to learn semantic class from the Web. As mentioned earlier, their system requires the name of the semantic set and an additional seed instance. Pasca [7, 6] illustrated an set expansion approach that extracts instances from Web search queries given a set of input seed instances, which is similar in flavor to our system SEAL but different from the task addressed in this paper: the user provides no seeds, but instead provides the *name* of the set being expanded.

In our own prior work, we used a noise-resistant version of SEAL to improve a Question Answering system for list questions [13]. Here, the input to SEAL was a set of noisy seeds provided by a Question Answering system. We have also separately explored the issue of using SEAL for set expansion in an iterative, bootstrapping manner, using a small number of clean seeds [12].

6. CONCLUSIONS

The results in Table 2 show that ASIE, a SEAL-based set instance extraction system, performs comparably to the method of Kozareva *et al* on two problems: US states and countries, with slightly higher precision being obtained on the larger "countries" problems. The SEAL-based system also obtains much better performance on the "common fish" task, obtaining a precision of 97% at rank 116, versus 80% for the Kozareva *et al* system. We note that the Kozareva *et al* system also has somewhat more information available than our system, as it assumes the user provides a class name and a single initial seed.

ASIE is also quite efficient, requiring only a few minutes of computation per problem and a small number of queries to an appropriate search engine API. Perhaps most importantly, the system we describe also obtains comparable results on 36 additional benchmark problems in three languages (Chinese and Japanese as well as English). The ability to construct semantic lexicons in diverse languages has obvious applications in machine translation. At the end of this paper, we have included some real example inputs and outputs of ASIE in Table 4 for English and in Figure 6 for Chinese and Japanese.

In further work we plan to explore other aspects of this language-independence. For instance, we wish to explore whether lexicons can be constructed using only the "back-off method" for hyponym extraction (in which the initial set of "noisy seeds" are just "chunks" retrieved from a query consisting the semantic class name itself). The advantage of this more restrictive approach is that it is completely language independent. We also wish to explore whether performance can be improved by simultaneously finding class instances in multiple languages (e.g., Chinese and English) while simultaneously learning translations between the extracted instances.

7. ACKNOWLEDGMENTS

This work was supported in part by the Google Research Awards program.

8. REFERENCES

[1] O. Etzioni, M. J. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, 165(1):91–134, 2005.

[2] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *In Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545, 1992.

[3] Z. Kozareva. Bootstrapping named entity recognition with automatically generated gazetteer lists. In *EACL*. The Association for Computer Linguistics, 2006.

[4] Z. Kozareva, E. Riloff, and E. Hovy. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL-08: HLT*, pages 1048–1056, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[5] D. Nadeau, P. D. Turney, and S. Matwin. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In L. Lamontagne and M. Marchand, editors, *Canadian Conference on AI*, volume 4013 of *Lecture Notes in Computer Science*, pages 266–277. Springer, 2006.

[6] M. Paşca. Organizing and searching the world wide web of facts – step two: harnessing the wisdom of the crowds. In *WWW '07: Proceedings of the 16th*

international conference on World Wide Web, pages 101–110, New York, NY, USA, 2007. ACM.

[7] M. Paşca. Weakly-supervised discovery of named entities using web search queries. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 683–690, New York, NY, USA, 2007. ACM.

[8] P. Pantel and D. Ravichandran. Automatically labeling semantic classes. In D. M. Susan Dumais and S. Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 321–328, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

[9] M. Pasca. Acquisition of categorized named entities for web search. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 137–145, New York, NY, USA, 2004. ACM.

[10] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *ICDM*, pages 613–622. IEEE Computer Society, 2006.

[11] R. C. Wang and W. W. Cohen. Language-independent set expansion of named entities using the web. In *ICDM*, pages 342–350. IEEE Computer Society, 2007.

[12] R. C. Wang and W. W. Cohen. Iterative set expansion of named entities using the web. To appear in the Proceedings of 8th IEEE International Conference on Data Mining, December 2008.

[13] R. C. Wang, N. Schlaefer, W. W. Cohen, and E. Nyberg. Automatic set expansion for list question answering. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 947–954, Honolulu, Hawaii, October 2008. Association for Computational Linguistics.

Simpsons Characters	Time Travel Movies	Nobel Prize Winners	Tropical Fruits	Natural Disasters
marge simpson	the time machine	nelson mandela	mango	earthquakes
krusty the clown	back to the future	woodrow wilson	papaya	tornadoes
bart simpson	time bandits	albert schweitzer	pineapple	volcanoes
lisa simpson	somewhere in time	mother teresa	jackfruit	hurricanes
homer simpson	peggy sue got married	elie wiesel	mangosteen	floods
ralph wiggum	time after time	mikhail gorbachev	guava	tsunamis
ned flanders	millennium	jimmy carter	coconut	landslides
maggie simpson	frequency	desmond tutu	rambutan	wildfires
comic book guy	the final countdown	linus pauling	durian	avalanches
waylon smithers	timeline	theodore roosevelt	banana	tsunami
kent brockman	donnie darko	willy brandt	lychee	blizzards
apu nahasapeemapetilon	the terminator	aung san suu kyi	longan	fires
carl carlson	the philadelphia experiment	amnesty international	pomegranate	earthquake
chief wiggum	terminator	elihu root	breadfruit	storms
milhouse van houten	planet of the apes	yitzhak rabin	tamarind	weather
nelson muntz	lost in space	menachem begin	soursop	volcano
groundskeeper willie	groundhog day	henry kissinger	avocado	hurricane
barney gumble	kate and leopold	shimon peres	jujube	lightning
montgomery burns	flight of the navigator	cordell hull	pomelo	tornado
sideshow bob	timecop	shirin ebadi	watermelon	droughts
lenny leonard	butterfly effect	frank b. kellogg	langsat	flooding
hans moleman	happy accidents	andrei sakharov	passionfruit	forest fires
moe szyslak	army of darkness	fridtjof nansen	tangerine	volcanic eruptions

Table 4: Real examples of ASIE's inputs and outputs in English.

夜市	台南小吃	觀光景點	ドラマ	アニメ	宮崎駿の映画
武聖夜市	棺材板	日月潭	プロポーズ大作戦	おおきく振りかぶって	風の谷のナウシカ
大東夜市	擔仔麵	中正紀念堂	ホテリアー	銀魂	もののけ姫
花園夜市	鱔魚意麵	龍山寺	ガリレオ	瀬戸の花嫁	千と千尋の神隠し
小北夜市	蚵仔煎	阿里山	山田太郎ものがたり	ながされて藍蘭島	魔女の宅急便
永華夜市	碗粿	清境農場	ホタルノヒカリ	怪物王女	となりのトトロ
成功夜市	肉圓	故宮博物院	ファースト・キス	天元突破グレンラガン	天空の城ラピュタ
大統一夜市	米糕	忠烈祠	働きマン	名探偵コナン	紅の豚
大林夜市	蝦捲	故宮	山おんな壁おんな	エル・カザド	ハウルの動く城
光華夜市	周氏蝦捲	太魯閣	ハケンの品格	結界師	耳をすませば
國泰夜市	肉粽	旅遊	ハタチの恋人	かみちゃまかりん	おもひでぽろぽろ
興中夜市	虱目魚粥	總統府	暴れん坊ママ	電脳コイル	火垂るの墓
民雄夜市	鴨肉羹	墾丁	受験の神様	ぼくらの	平成狸合戦ぽんぽこ
瑞豐夜市	魚丸湯	澎湖	歌姫	ゲゲゲの鬼太郎	未来少年コナン
新興夜市	蝦仁肉圓	宜蘭	有閑倶楽部	ケロロ軍曹	パンダコパンダ
青年夜市	意麵	九份	ライフ	さよなら絶望先生	崖の上のポニョ
家樂福夜市	東山鴨頭	花蓮	モップガール	ヒロイック・エイジ	ナウシカ
朴子夜市	鹹粥	南投	女帝	スカイガールズ	猫の恩返し
大昌夜市	魚羹	士林官邸	華麗なる一族	桃華月憚	ルパン三世
忠孝路夜市	豆花	九族文化村	肩ごしの恋人	もえたん	スタジオジブリ
文化路夜市	鼎邊銼	野柳	わたしたちの教科書	ひぐらしのなく頃に解	母をたずねて三千里
六合夜市	安平豆花	陽明山	ハチミツとクローバー	精霊の守り人	ゲド戦記
花園街夜市	北平烤鴨	小人國	ライアーゲーム	アイシールド21	アルプスの少女ハイジ
南雅夜市	肉包	陽明山國家公園	ヒミツの花園	史上最強の弟子ケンイチ	三鷹の森ジブリ美術館
城隍廟夜市	菴母鴨	士林夜市	オトユの子育て	ムシウタ	ラピュタ
開元夜市	鵝肉	合歡山	花嫁とパパ	ぱてまよ	王と鳥
公館夜市	綠豆湯	墾丁國家公園	薔薇のない花屋	ドージンワーク	ハウル
中華路夜市	鼎邊趖	淡水	パパとムスメの7日間	風のスティグマ	トトロ
六甲夜市	浮水魚羹	澄清湖	地獄の沙汰もヨメ次第	大江戸ロケット	千と千尋
隆田夜市	肉丸	桃園	帰ってきた時効警察	鋼鉄三国志	徳間書店
民族路夜市	八寶冰	台北101	冬のソナタ	モノノ怪	海がきこえる

Figure 6: Real examples of ASIE's inputs and outputs in Chinese and Japanese.