In *Proceedings of ACL-08*

Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs

Zornitsa Kozareva
DLSI, University of Alicante
Campus de San Vicente
Alicante, Spain 03080
zkozareva@dlsi.ua.es

Ellen Riloff
School of Computing
University of Utah
Salt Lake City, UT 84112
riloff@cs.utah.edu

Eduard Hovy
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
hovy@isi.edu

Abstract

We present a novel approach to weakly supervised semantic class learning from the web, using a single powerful hyponym pattern combined with graph structures, which capture two properties associated with pattern-based extractions: *popularity* and *productivity*. Intuitively, a candidate is *popular* if it was discovered many times by other instances in the hyponym pattern. A candidate is *productive* if it frequently leads to the discovery of other instances. Together, these two measures capture not only frequency of occurrence, but also cross-checking that the candidate occurs both near the class name and near other class members. We developed two algorithms that begin with just a class name and one seed instance and then automatically generate a ranked list of new class instances. We conducted experiments on four semantic classes and consistently achieved high accuracies.

1 Introduction

Knowing the semantic classes of words (e.g., "trout" is a kind of FISH) can be extremely valuable for many natural language processing tasks. Although some semantic dictionaries do exist (e.g., WordNet (Miller, 1990)), they are rarely complete, especially for large open classes (e.g., classes of people and objects) and rapidly changing categories (e.g., computer technology). (Roark and Charniak, 1998) reported that 3 of every 5 terms generated by their semantic lexicon learner were not present in WordNet. Automatic semantic lexicon acquisition could be used to enhance existing resources such as WordNet, or to produce semantic lexicons for specialized categories or domains.

A variety of methods have been developed for automatic semantic class identification, under the rubrics of lexical acquisition, hyponym acquisition, semantic lexicon induction, semantic class learning, and web-based information extraction. Many of these approaches employ surface-level patterns to identify words and their associated semantic classes. However, such patterns tend to overgenerate (i.e., deliver incorrect results) and hence require additional filtering mechanisms.

To overcome this problem, we employed one single powerful *doubly-anchored* hyponym pattern to query the web and extract semantic class instances: CLASS_NAME *such as* CLASS_MEMBER *and* *.

We hypothesized that a doubly-anchored pattern, which includes both the class name and a class member, would achieve high accuracy because of its specificity. To address concerns about coverage, we embedded the search in a bootstrapping process. This method produced many correct instances, but despite the highly restrictive nature of the pattern, still produced many incorrect instances. This result led us to explore new ways to improve the accuracy of hyponym patterns without requiring additional training resources.

The main contribution of this work is a novel method for combining hyponym patterns with graph structures that capture two properties associated with pattern extraction: *popularity* and *productivity*. Intuitively, a candidate word (or phrase) is *popular* if it was discovered many times by other words (or

phrases) in a hyponym pattern. A candidate word is *productive* if it frequently leads to the discovery of other words. Together, these two measures capture not only frequency of occurrence, but also cross-checking that the word occurs both near the class name and near other class members.

We present two algorithms that use *hyponym pattern linkage graphs (HPLGs)* to represent popularity and productivity information. The first method uses a dynamically constructed HPLG to assess the popularity of each candidate and steer the bootstrapping process. This approach produces an efficient bootstrapping process that performs reasonably well, but it cannot take advantage of productivity information because of the dynamic nature of the process.

The second method is a two-step procedure that begins with an exhaustive pattern search that acquires popularity and productivity information about candidate instances. The candidates are then ranked based on properties of the HPLG. We conducted experiments with four semantic classes, achieving high accuracies and outperforming the results reported by others who have worked on the same classes.

2 Related Work

A substantial amount of research has been done in the area of semantic class learning, under a variety of different names and with a variety of different goals. Given the great deal of similar work in information extraction and ontology learning, we focus here only on techniques for weakly supervised or unsupervised semantic class (i.e., supertype-based) learning, since that is most related to the work in this paper.

Fully unsupervised semantic clustering (e.g., (Lin, 1998; Lin and Pantel, 2002; Davidov and Rappoport, 2006)) has the disadvantage that it may or may not produce the types and granularities of semantic classes desired by a user. Another related line of work is automated ontology construction, which aims to create lexical hierarchies based on semantic classes (e.g., (Caraballo, 1999; Cimiano and Volker, 2005; Mann, 2002)), and learning semantic relations such as meronymy (Berland and Charniak, 1999; Girju et al., 2003).

Our research focuses on semantic lexicon induction, which aims to generate lists of words that belong to a given semantic class (e.g., lists of FISH or VEHICLE words). Weakly supervised learning methods for semantic lexicon generation have utilized co-occurrence statistics (Riloff and Shepherd, 1997; Roark and Charniak, 1998), syntactic information (Tanev and Magnini, 2006; Pantel and Ravichandran, 2004; Phillips and Riloff, 2002), lexico-syntactic contextual patterns (e.g., *"resides in <location>"* or *"moved to <location>"*) (Riloff and Jones, 1999; Thelen and Riloff, 2002), and local and global contexts (Fleischman and Hovy, 2002). These methods have been evaluated only on fixed corpora[1], although (Pantel et al., 2004) demonstrated how to scale up their algorithms for the web.

Several techniques for semantic class induction have also been developed specifically for learning from the web. (Paşca, 2004) uses Hearst's patterns (Hearst, 1992) to learn semantic class instances and class groups by acquiring contexts around the pattern. Pasca also developed a second technique (Paşca, 2007b) that creates context vectors for a group of seed instances by searching web query logs, and uses them to learn similar instances.

The work most closely related to ours is Hearst's early work on hyponym learning (Hearst, 1992) and more recent work that has followed up on her idea. Hearst's system exploited patterns that explicitly identify a hyponym relation between a semantic class and a word (e.g., *"such authors as Shakespeare"*). We will refer to these as *hyponym patterns*. Pasca's previously mentioned system (Paşca, 2004) applies hyponym patterns to the web and acquires contexts around them. The KnowItAll system (Etzioni et al., 2005) also uses hyponym patterns to extract class instances from the web and then evaluates them further by computing mutual information scores based on web queries.

The work by (Widdows and Dorow, 2002) on lexical acquisition is similar to ours because they also use graph structures to learn semantic classes. However, their graph is based entirely on syntactic relations between words, while our graph captures the ability of instances to find each other in a hyponym pattern based on web querying, without any part-of-speech tagging or parsing.

[1] Meta-bootstrapping (Riloff and Jones, 1999) was evaluated on web pages, but used a precompiled corpus of downloaded web pages.

3 Semantic Class Learning with Hyponym Pattern Linkage Graphs

3.1 A Doubly-Anchored Hyponym Pattern

Our work was motivated by early research on hyponym learning (Hearst, 1992), which applied patterns to a corpus to associate words with semantic classes. Hearst's system exploited patterns that explicitly link a class name with a class member, such as *"X and other Ys"* and *"Ys such as X"*. Relying on surface-level patterns, however, is risky because incorrect items are frequently extracted due to polysemy, idiomatic expressions, parsing errors, etc.

Our work began with the simple idea of using an *extremely* specific pattern to extract semantic class members with high accuracy. Our expectation was that a very specific pattern would virtually eliminate the most common types of false hits that are caused by phenomena such as polysemy and idiomatic expressions. A concern, however, was that an extremely specific pattern would suffer from sparse data and not extract many new instances. By using the web as a corpus, we hoped that the pattern could extract at least a few instances for virtually any class, and then we could gain additional traction by bootstrapping these instances.

All of the work presented in this paper uses just one *doubly-anchored* pattern to identify candidate instances for a semantic class:

*<class_name> such as <class_member> and **

This pattern has two variables: the name of the semantic class to be learned (*class_name*) and a member of the semantic class (*class_member*). The asterisk (*) indicates the location of the extracted words. We describe this pattern as being *doubly-anchored* because it is instantiated with both the name of the semantic class as well as a class member.

For example, the pattern *"CARS such as FORD and *"* will extract automobiles, and the pattern *"PRESIDENTS such as FORD and *"* will extract presidents. The doubly-anchored nature of the pattern serves two purposes. First, it increases the likelihood of finding a true list construction for the class. Our system does not use part-of-speech tagging or parsing, so the pattern itself is the only guide for finding an appropriate linguistic context.

Second, the doubly-anchored pattern virtually eliminates ambiguity because the *class_name* and *class_member* mutually disambiguate each other. For example, the word FORD could refer to an automobile or a person, but in the pattern *"CARS such as FORD and *"* it will almost certainly refer to an automobile. Similarly, the class "PRESIDENT" could refer to country presidents or corporate presidents, and "BUSH" could refer to a plant or a person. But in the pattern *"PRESIDENTS such as BUSH"*, both words will surely refer to country presidents.

Another advantage of the doubly-anchored pattern is that an ambiguous or underspecified class name will be constrained by the presence of the class member. For example, to generate a list of company presidents, someone might naively define the class name as PRESIDENTS. A singly-anchored pattern (e.g., *"PRESIDENTS such as *"*) might generate lists of other types of presidents (e.g., country presidents, university presidents, etc.). Because the doubly-anchored pattern also requires a class member (e.g., *"PRESIDENTS such as BILL GATES and *"*), it is likely to generate only the desired types of instances.

```
Members = {Seed};
P0= "Class such as Seed and *";
P = {P0};
iter = 0;
While ((iter < Max_Iters) and (P ≠ {}))
  iter++;
  For each Pi ∈ P
    Snippets = web_query(Pi);
    Candidates = extract_words(Snippets,Pi);
    Pnew = {};
    For each Candidatek ∈ Candidates
      If (Candidatek ∉ Members);
        Members = Members ∪ {Candidatek};
        Pk= "Class such as Candidatek and *";
        Pnew = Pnew ∪ { Pk };
  P = Pnew;
```

Figure 1: Reckless Bootstrapping

3.2 Reckless Bootstrapping

To evaluate the performance of the doubly-anchored pattern, we began by using the pattern to search the web and embedded this process in a simple bootstrapping loop, which is presented in Figure 1. As input, the user must provide the name of the desired

semantic class (*Class*) and a seed example (*Seed*), which are used to instantiate the pattern. On the first iteration, the pattern is given to Google as a web query, and new class members are extracted from the retrieved text snippets. We wanted the system to be as language-independent as possible, so we refrained from using any taggers or parsing tools. As a result, instances are extracted using only word boundaries and orthographic information. For proper name classes, we extract all capitalized words that immediately follow the pattern. For common noun classes, we extract just one word, if it is not capitalized. Examples are shown below, with the extracted items underlined:

countries such as China and Sri Lanka are ...
fishes such as trout and bass can ...

One limitation is that our system cannot learn multi-word instances of common noun categories, or proper names that include uncapitalized words (e.g., "United States of America"). These limitations could be easily overcome by incorporating a noun phrase (NP) chunker and extracting NPs.

Each new class member is then used as a seed instance in the bootstrapping loop. We implemented this process as breadth-first search, where each "ply" of the search process is the result of bootstrapping the class members learned during the previous iteration as seed instances for the next one. During each iteration, we issue a new web query and add the newly extracted class members to the queue for the next cycle. We run this bootstrapping process for a fixed number of iterations (search ply), or until no new class members are produced. We will refer to this process as *reckless bootstrapping* because there are no checks of any kind. Every term extracted by the pattern is assumed to be a class member.

3.2.1 Results

Table 1 shows the results for 4 iterations of reckless bootstrapping for four semantic categories: *U.S. states*, *countries*, *singers*, and *fish*. The first two categories are relatively small, closed sets (our gold standard contains 50 U.S. states and 194 countries). The *singers* and *fish* categories are much larger, open sets (see Section 4 for details).

Table 1 reveals that the doubly-anchored pattern achieves high accuracy during the first iteration, but quality deteriorates rapidly as bootstrapping progresses. Figure 2 shows the recall and precision curves for countries and states. High precision is achieved only with low levels of recall for countries. Our initial hypothesis was that such a specific pattern would be able to maintain high precision because non-class members would be unlikely to co-occur with the pattern. But we were surprised to find that many incorrect entries were generated for reasons such as broken expressions like "Merce -dez", misidentified list constructions (e.g., *"In countries such as China U.S. Policy is failing..."*), and incomplete proper names due to insufficient length of the retrieved text snippet.

Iter.	*countries*	*states*	*singers*	*fish*
1	.80	.79	.91	.76
2	.57	.21	.87	.64
3	.21	.18	.86	.54
4	.16	–	.83	.54

Table 1: *Reckless Bootstrapping Accuracies*

Incorporating a noun phrase chunker would eliminate some of these cases, but far from all of them. We concluded that even such a restrictive pattern is not sufficient for semantic class learning on its own.

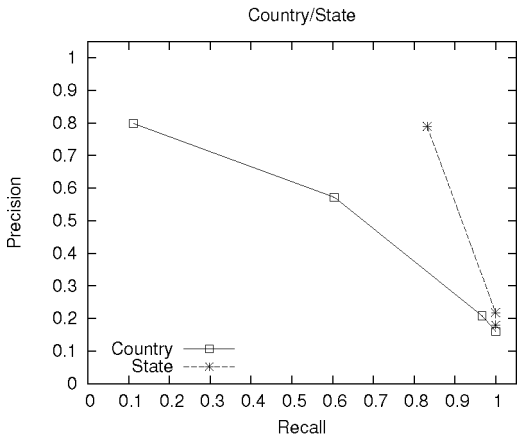

Figure 2: Recall/precision for reckless bootstrapping

In the next section, we present a new approach that creates a Hyponym Pattern Linkage Graph to steer bootstrapping and improve accuracy.

3.3 Using Dynamic Graphs to Steer Bootstrapping

Intuitively, we expect true class members to occur frequently in pattern contexts with other class mem-

bers. To operationalize this intuition, we create a *hyponym pattern linkage graph*, which represents the frequencies with which candidate instances generate each other in the pattern contexts.

We define a *hyponym pattern linkage graph (HPLG)* as a $G = (V, E)$, where each vertex $v \in V$ is a candidate instance and each edge $(u, v) \in E$ means that instance v was generated by instance u. The weight w of an edge is the frequency with which u generated v. For example, consider the following sentence, where the pattern is italicized and the extracted instance is underlined:

Countries such as China and Laos have been...

In the HPLG, an edge $e = (China, Laos)$ would be created because the pattern anchored by China extracted Laos as a new candidate instance. If this pattern extracted Laos from 15 different snippets, then the edge's weight would be 15. The in-degree of a node represents its *popularity*, i.e., the number of instance occurrences that generated it.

The graph is constructed dynamically as bootstrapping progresses. Initially, the seed is the only *trusted class member* and the only vertex in the graph. The bootstrapping process begins by instantiating the doubly-anchored pattern with the seed class member, issuing a web query to generate new candidate instances, and adding these new instances to the graph. A score is then assigned to every node in the graph, using one of several different metrics defined below. The highest-scoring unexplored node is then added to the set of *trusted class members*, and used as the seed for the next bootstrapping iteration.

We experimented with three scoring functions for selecting nodes. The **In-Degree (inD) score** for vertex v is the sum of the weights of all incoming edges (u, v), where u is a trusted class member. Intuitively, this captures the popularity of v among instances that have already been identified as good instances. The **Best Edge (BE) score** for vertex v is the maximum edge weight among the incoming edges (u, v), where u is a trusted class member.

The Key Player Problem (KPP) measure is used in social network analysis (Borgatti and Everett, 2006) to identify nodes whose removal would result in a residual network of minimum cohesion. A node receives a high value if it is highly connected and relatively close to most other nodes in the graph. The **KPP score** for vertex v is computed as:

$$KPP(v) = \frac{\sum_{u \in V} \frac{1}{d(u,v)}}{|V|-1}$$

where $d(u, v)$ is the shortest path between two vertices, where u is a trusted node. For tie-breaking, the distances are multiplied by the weight of the edge.

Note that all of these measures rely only on incoming edges because a node does not acquire outgoing edges until it has already been selected as a trusted class member and used to acquire new instances. In the next section, we describe a two-step process for creating graphs that can take advantage of both incoming and outgoing edges.

3.4 Re-Ranking with Precompiled Graphs

One way to try to confirm (or disconfirm) whether a candidate instance is a true class member is to see whether it can produce new candidate instances. If we instantiate our pattern with the candidate (i.e., "CLASS_NAME *such as* CANDIDATE *and* *") and successfully extract many new instances, then this is evidence that the candidate frequently occurs with the CLASS_NAME in list constructions. We will refer to the ability of a candidate to generate new instances as its *productivity*.

The previous bootstrapping algorithm uses a dynamically constructed graph that is constantly evolving as new nodes are selected and explored. Each node is scored based only on the set of instances that have been generated and identified as "trusted" at that point in the bootstrapping process. To use productivity information, we must adopt a different procedure because we need to know not only who generated each candidate, but also the complete set of instances that the candidate itself can generate.

We adopted a two-step process that can use both popularity and productivity information in a hyponym pattern linkage graph to assess the quality of candidate instances. First, we perform *reckless bootstrapping* for a *class_name* and *seed* until no new instances are generated. Second, we assign a score to each node in the graph using a scoring function that takes into account both the in-degree (popularity) and out-degree (productivity) of each node. We experimented with four different scoring functions, some of which were motivated by work on word

sense disambiguation to identify the most "important" node in a graph containing its possible senses (Navigli and Lapata, 2007).

The **Out-degree (outD) score** for vertex v is the weighted sum of v's outgoing edges, normalized by the number of other nodes in the graph.

$$outD(v) = \frac{\sum_{\forall(v,p)\in E} w(v,p)}{|V|-1}$$

This measure captures only productivity, while the next three measures consider both productivity and popularity. The **Total-degree (totD) score** for vertex v is the weighted sum of both incoming and outgoing edges, normalized by the number of other nodes in the graph. The **Betweenness (BT) score** (Freeman, 1979) considers a vertex to be important if it occurs on many shortest paths between other vertices.

$$BT(v) = \sum_{s,t\in V: s\neq v\neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is the number of shortest paths from s to t, and $\sigma_{st}(v)$ is the number of shortest paths from s to t that pass through vertex v. PageRank (Page et al., 1998) establishes the relative importance of a vertex v through an iterative Markov chain model. The **PageRank (PR) score** of a vertex v is determined on the basis of the nodes it is connected to.

$$PR(v) = \frac{(1-\alpha)}{|V|} + \alpha \sum_{u,v\in E} \frac{PR(u)}{outdegree(u)}$$

α is a damping factor that we set to 0.85. We discarded all instances that produced zero productivity links, meaning that they did not generate any other candidates when used in web queries.

States							
	Popularity			Prd	Pop&Prd		
N	BE	KPP	inD	outD	totD	BT	PR
25	1.0	1.0	1.0	1.0	1.0	.88	.88
50	.96	.98	.98	1.0	1.0	.86	.82
64	.77	.78	.77	.78	.78	.77	.67
Countries							
	Popularity			Prd	Pop&Prd		
N	BE	KPP	inD	outD	totD	BT	PR
50	.98	.97	.98	1.0	1.0	.98	.97
100	.96	.97	.94	1.0	.99	.97	.95
150	.90	.92	.91	1.0	.95	.94	.92
200	.83	.81	.83	.90	.87	.82	.80
300	.60	.59	.61	.61	.62	.56	.60
323	.57	.55	.57	.57	.58	.52	.57
Singers							
	Popularity			Prd	Pop&Prd		
N	BE	KPP	inD	outD	totD	BT	PR
10	.92	.96	.92	1.0	1.0	1.0	1.0
25	.89	.90	.91	1.0	1.0	1.0	.99
50	.92	.85	.92	.97	.98	.95	.97
75	.89	.83	.91	.96	.95	.93	.95
100	.86	.81	.89	.96	.93	.94	.94
150	.86	.79	.88	.95	.92	.93	.87
180	.86	.80	.87	.91	.91	.91	.88
Fish							
	Popularity			Prd	Pop&Prd		
N	BE	KPP	inD	outD	totD	BT	PR
10	.90	.90	.90	1.0	1.0	.90	.70
25	.80	.88	.76	1.0	.96	.96	.72
50	.82	.80	.78	1.0	.94	.88	.66
75	.72	.69	.72	.93	.87	.79	.64
100	.63	.68	.66	.84	.80	.74	.62
116	.60	.65	.66	.80	.78	.71	.59

Table 2: *Accuracies for each semantic class*

4 Experimental evaluation

4.1 Data

We evaluated our algorithms on four semantic categories: *U.S. states*, *countries*, *singers*, and *fish*. The *states* and *countries* categories are relatively small, closed sets: our gold standards consist of 50 U.S. states and 194 countries (based on a list found on Wikipedia). The *singers* and *fish* categories are much larger, open classes. As our gold standard for *fish*, we used a list of common fish names found on Wikipedia.[2] All the singer names generated by our algorithms were manually reviewed for correctness. We evaluated performance in terms of accuracy (the percentage of instances that were correct).[3]

4.2 Performance

Table 2 shows the accuracy results of the two algorithms that use hyponym pattern linkage graphs. We display results for the top-ranked N candidates, for all instances that have a productivity value $>$ zero.[4] The Popularity columns show results for the

[2]We also counted as correct plural versions of items found on the list. The total size of our fish list is 1102.

[3]We never generated duplicates so the instances are distinct.

[4]Obviously, this cutoff is not available to the popularity-based bootstrapping algorithm, but here we are just comparing the top N results for both algorithms.

bootstrapping algorithm described in Section 3.3, using three different scoring functions. The results for the ranking algorithm described in Section 3.4 are shown in the Productivity (Prd) and Popularity&Productivity (Pop&Prd) columns. For the *states*, *countries*, and *singers* categories, we randomly selected 5 different initial seeds and then averaged the results. For the *fish* category we ran each algorithm using just the seed *"salmon"*.

The popularity-based metrics produced good accuracies on the *states*, *countries*, and *singers* categories under all 3 scoring functions. For *fish*, KPP performed better than the others.

The *Out-degree (outD)* scoring function, which uses only Productivity information, obtained the best results across all 4 categories. OutD achieved 100% accuracy for the first 50 states and fish, 100% accuracy for the top 150 countries, and 97% accuracy for the top 50 singers. The three scoring metrics that use both popularity and productivity also performed well, but productivity information by itself seems to perform better in some cases.

It can be difficult to compare the results of different semantic class learners because there is no standard set of benchmark categories, so researchers report results for different classes. For the state and country categories, however, we can compare our results with that of other web-based semantic class learners such as Pasca (Paşca, 2007a) and the KnowItAll system (Etzioni et al., 2005). For the U.S. states category, our system achieved 100% recall and 100% precision for the first 50 items generated, and KnowItAll performed similarly achieving 98% recall with 100% precision. Pasca did not evaluate his system on states.

For the *countries* category, our system achieved 100% precision for the first 150 generated instances (77% recall). (Paşca, 2007a) reports results of 100% precision for the first 25 instances generated, and 82% precision for the first 150 instances generated. The KnowItAll system (Etzioni et al., 2005) achieved 97% precision with 58% recall, and 79% precision with 87% recall.[5] To the best of our knowledge, other researchers have not reported results for the singer and fish categories.

[5](Etzioni et al., 2005) do not report exactly how many countries were in their gold standard.

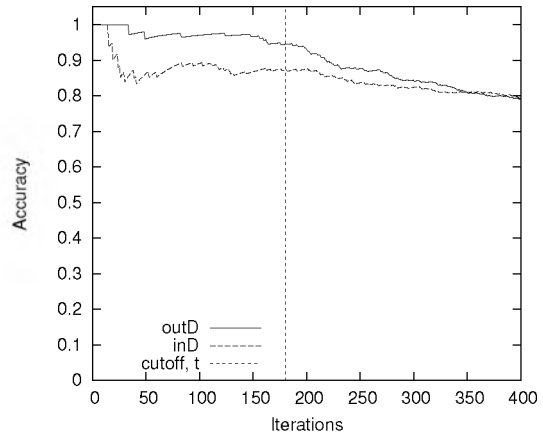

Figure 3: Learning curve for Placido Domingo

Figure 3 shows the learning curve for both algorithms using their best scoring functions on the *singer* category with *Placido Domingo* as the initial seed. In total, 400 candidate words were generated. The *Out-degree* scoring function ranked the candidates well. Figure 3 also includes a vertical line indicating where the candidate list was cut (at 180 instances) based on the zero productivity cutoff.

One observation is that the rankings do a good job of identifying borderline cases, which typically are ranked just below most correct instances but just above the obviously bad entries. For example, for *states*, the 50 U.S. states are ranked first, followed by 14 more entries (in order):

> Russia, Ukraine, Uzbekistan, Azerbaijan, Moldova, Tajikistan, Armenia, Chicago, Boston, Atlanta, Detroit, Philadelphia, Tampa, Moldavia

The first 7 entries are all former states of the Soviet Union. In retrospect, we realized that we should have searched for "U.S. states" instead of just "states". This example illustrates the power of the doubly-anchored hyponym pattern to correctly identify our intended semantic class by disambiguating our class name based on the seed class member.

The algorithms also seem to be robust with respect to initial seed choice. For the *states*, *countries*, and *singers* categories, we ran experiments with 5 different initial seeds, which were randomly selected. The 5 country seeds represented a diverse set of nations, some of which are rarely mentioned in the news: *Brazil*, *France*, *Guinea-Bissau*, *Uganda*,

and *Zimbabwe*. All of these seeds obtained $\geq 92\%$ recall with $\geq 90\%$ precision.

4.3 Error Analysis

We examined the incorrect instances produced by our algorithms and found that most of them fell into five categories.

Type 1 errors were caused by incorrect proper name extraction. For example, in the sentence *"states such as Georgia and English speaking countries like Canada..."*, *"English"* was extracted as a state. These errors resulted from complex noun phrases and conjunctions, as well as unusual syntactic constructions. An NP chunker might prevent some of these cases, but we suspect that many of them would have been misparsed regardless.

Type 2 errors were caused by instances that formerly belonged to the semantic class (e.g., *Serbia-Montenegro* and *Czechoslovakia* are no longer countries). In this error type, we also include borderline cases that could arguably belong to the semantic class (e.g., Wales as a country).

Type 3 errors were spelling variants (e.g., *Kyrgystan* vs. *Kyrgyzhstan*) and name variants (e.g., *Beyonce* vs. *Beyonce Knowles*). Officially, every entity has one official spelling and one complete name, but in practice there are often variations that may occur nearly as frequently as the official name. For example, it is most common to refer to the singer *Beyonce* by just her first name.

Type 4 errors were caused by sentences that were just flat out wrong in their factual assertions. For example, some sentences referred to *"North America"* as a country.

Type 5 errors were caused by broken expressions found in the retrieved snippets (e.g. *Michi -gan*). These errors may be fixable by cleaning up the web pages or applying heuristics to prevent or recognize partial words.

It is worth noting that incorrect instances of Types 2 and 3 may not be problematic to encounter in a dictionary or ontology. Name variants and former class members may in fact be useful to have.

5 Conclusions

Combining hyponym patterns with pattern linkage graphs is an effective way to produce a highly accurate semantic class learner that requires truly minimal supervision: just the class name and one class member as a seed. Our results consistently produced high accuracy and for the *states* and *countries* categories produced very high recall.

The *singers* and *fish* categories, which are much larger open classes, also achieved high accuracy and generated many instances, but the resulting lists are far from complete. Even on the web, the doubly-anchored hyponym pattern eventually ran out of steam and could not produce more instances. However, all of our experiments were conducted using just a single hyponym pattern. Other researchers have successfully used sets of hyponym patterns (e.g., (Hearst, 1992; Etzioni et al., 2005; Paşca, 2004)), and multiple patterns could be used with our algorithms as well. Incorporating additional hyponym patterns will almost certainly improve coverage, and could potentially improve the quality of the graphs as well.

Our popularity-based algorithm was very effective and is practical to use. Our best-performing algorithm, however, was the 2-step process that begins with an exhaustive search (reckless bootstrapping) and then ranks the candidates using the *Outdegree* scoring function, which represents productivity. The first step is expensive, however, because it exhaustively applies the pattern to the web until no more extractions are found. In our evaluation, we ran this process on a single PC and it usually finished overnight, and we were able to learn a substantial number of new class instances. If more hyponym patterns are used, then this could get considerably more expensive, but the process could be easily parallelized to perform queries across a cluster of machines. With access to a cluster of ordinary PCs, this technique could be used to automatically create extremely large, high-quality semantic lexicons, for virtually any categories, without external training resources.

Acknowledgments

This research was supported in part by the Department of Homeland Security under ONR Grants N00014-07-1-014 and N0014-07-1-0152, the European Union Sixth Framework project QALLME FP6 IST-033860, and the Spanish Ministry of Science and Technology TEXT-MESS TIN2006-15265-C06-01.

References

M. Berland and E. Charniak. 1999. Finding Parts in Very Large Corpora. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics.*

S. Borgatti and M. Everett. 2006. A graph-theoretic perspective on centrality. *Social Networks*, 28(4).

S. Caraballo. 1999. Automatic Acquisition of a Hypernym-Labeled Noun Hierarchy from Text. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 120–126.

P. Cimiano and J. Volker. 2005. Towards large-scale, open-domain and ontology-based named entity classification. In *Proc. of Recent Advances in Natural Language Processing*, pages 166–172.

D. Davidov and A. Rappoport. 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *Proc. of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL.*

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence*, 165(1):91–134, June.

M.B. Fleischman and E.H. Hovy. 2002. Fine grained classification of named entities. In *Proc. of the 19th International Conference on Computational Linguistics*, pages 1–7.

C. Freeman. 1979. Centrality in social networks: Conceptual clarification. *Social Networks*, 1:215–239.

R. Girju, A. Badulescu, and D. Moldovan. 2003. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proc. of Conference of HLT / North American Chapter of the Association for Computational Linguistics.*

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

D. Lin and P. Pantel. 2002. Concept discovery from text. In *Proc. of the 19th International Conference on Computational linguistics*, pages 1–7.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *Proc. of the 17th international conference on Computational linguistics*, pages 768–774.

G. Mann. 2002. Fine-grained proper noun ontologies for question answering. In *Proc. of the 19th International Conference on Computational Linguistics*, pages 1–7.

G. Miller. 1990. Wordnet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4).

R. Navigli and M. Lapata. 2007. Graph connectivity measures for unsupervised word sense disambiguation. In *Proc. of the 20th International Joint Conference on Artificial Intelligence*, pages 1683–1688.

M. Paşca. 2004. Acquisition of categorized named entities for web search. In *Proc. of the Thirteenth ACM International Conference on Information and Knowledge Management*, pages 137–145.

M. Paşca. 2007a. Organizing and searching the world wide web of facts – step two: harnessing the wisdom of the crowds. In *Proc. of the 16th International Conference on World Wide Web*, pages 101–110.

M. Paşca. 2007b. Weakly-supervised discovery of named entities using web search queries. In *Proc. of the sixteenth ACM conference on Conference on information and knowledge management*, pages 683–690.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project.

P. Pantel and D. Ravichandran. 2004. Automatically labeling semantic classes. In *Proc. of Conference of HLT / North American Chapter of the Association for Computational Linguistics*, pages 321–328.

P. Pantel, D. Ravichandran, and E. Hovy. 2004. Towards terascale knowledge acquisition. In *Proc. of the 20th international conference on Computational Linguistics*, page 771.

W. Phillips and E. Riloff. 2002. Exploiting Strong Syntactic Heuristics and Co-Training to Learn Semantic Lexicons. In *Proc. of the 2002 Conference on Empirical Methods in Natural Language Processing.*

E. Riloff and R. Jones. 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proc. of the Sixteenth National Conference on Artificial Intelligence.*

E. Riloff and J. Shepherd. 1997. A Corpus-Based Approach for Building Semantic Lexicons. In *Proc. of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124.

B. Roark and E. Charniak. 1998. Noun-phrase Co-occurrence Statistics for Semi-automatic Semantic Lexicon Construction. In *Proc. of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 1110–1116.

H. Tanev and B. Magnini. 2006. Weakly supervised approaches for ontology population. In *Proc. of 11st Conference of the European Chapter of the Association for Computational Linguistics.*

M. Thelen and E. Riloff. 2002. A Bootstrapping Method for Learning Semantic Lexicons Using Extraction Pattern Contexts. In *Proc. of the 2002 Conference on Empirical Methods in Natural Language Processing.*

D. Widdows and B. Dorow. 2002. A graph model for unsupervised lexical acquisition. In *Proc. of the 19th International Conference on Computational Linguistics*, pages 1–7.