Bootstrapped Learning of Semantic Classes from Positive and Negative Examples

Winston Lin WINSTON@CS.NYU.EDU
Roman Yangarber ROMAN@CS.NYU.EDU
Ralph Grishman GRISHMAN@CS.NYU.EDU
Courant Institute of Mathematical Sciences, New York University, 715 Broadway, 7th Floor, New York, NY 10003 USA

Abstract

We present an algorithm for unsupervised learning and semantic classification of names and terms. Given a small number of seed examples and an unlabeled training corpus, the algorithm learns patterns that identify more examples, in a bootstrapping cycle. Multiple classes are learned simultaneously, including *negative classes* that serve to provide negative examples for the target classes. We apply the algorithm to texts from several domains, in English and Chinese.

1. Introduction

Many approaches to various text-understanding tasks—such as Information Extraction, Question Answering, and Summarization—employ the identification and classification of names as an essential lower-level component. In broad terms, names are lexical items that are unlikely to be found in general-purpose dictionaries, and in text understanding it is important to be able to identify and categorize names correctly.

Name classification is a much-studied field; we mention some of the recent contributions in Section 2. The work presented here extends the bootstrapping algorithm of Yangarber, Lin, and Grishman (2002) to obtain wider applicability and better performance on multiple categories of names. It is tested on several different scenarios, different corpora and in two languages, English and Chinese.

1.1. Proper Names and Generalized Names

Much previous research focuses on the Named Entity (NE) task (Defense Advanced Research Projects Agency, 1995), which is largely restricted to proper names. In English text, most proper names are capitalized. The classic NE categories are *person, organization*, and *location*.

In some languages, such as Chinese or German, capitalization is either unavailable or unreliable as a cue for name detection. Thus, other textual features must be used to detect and identify names.

In English, some domains contain semantic classes whose members are not uniformly capitalized: organisms, diseases, proteins, chemical compounds, etc.; e.g., “fruit fly” vs. “Drosophila,” or “anthrax” vs. “Siberian Plague.” These *generalized names* (GNs) (Yangarber et al., 2002) are harder to identify than conventional proper names (PNs).

1.2. Learning from Positive and Negative Examples

Building on contributions described in Section 2, we use unlabeled training data to detect and classify names. Given a small number of seed examples, the algorithm learns patterns that identify more names and continues in a bootstrapping cycle.

Using this general approach, Thelen and Riloff (2002) and Yangarber et al. (2002) found that performance was improved when multiple semantic classes were learned simultaneously.[1] The work presented here confirms this effect. Essentially, a seed name serves as a *positive example* for its own class and a *negative example* for all other classes. Negative examples help steer the learner away from unreliable patterns.

[1]In another application of bootstrapping, Yangarber (2003) found a similar improvement from simultaneous learning of information extraction patterns for multiple scenarios.

For PNs in English, a few classes of interest may provide adequate negative examples for each other. E.g., Collins and Singer (1999) found that 91% of the proper names in their test set were names of persons, organizations, or locations. Thus, the precision of a learner for person names depends greatly on how well it avoids organizations and locations.

When capitalization cues are unavailable, the learner must consider a much larger space of possible names. E.g., any noun phrase could be a disease name in English or an organization name in Chinese. In such situations, we have found it valuable to learn *negative classes* that are not of interest in themselves but serve to provide negative examples for the target classes. In particular, including a "none of the above" class improved precision substantially.

Additional effort may be needed to prevent a target class from "creeping" into related concepts. E.g., a learner for disease names may acquire patterns such as "suffering from X" that are associated with both diseases and symptoms. If such ambiguous patterns lead the learner to misclassify enough symptoms as diseases, it will begin to acquire other patterns that are primarily associated with symptoms.

When likely areas of creep are anticipated, negative classes can help prevent the learner from acquiring ambiguous patterns. Another potentially useful strategy is to have a human reviewer reject misclassified names during the bootstrapping process. Human review can protect against unanticipated errors, but allow the learner to benefit from ambiguous patterns that would be blocked by an additional negative class. Section 4.3 describes a simulation designed to explore the effects of human review.

2. Related Work

Supervised algorithms for PN identification have been extensively studied (Bikel et al., 1997). Unsupervised algorithms (bootstrapping from seed examples and unlabeled data) were developed by Collins and Singer (1999) and Cucerzan and Yarowsky (1999, 2002).[2] Collins and Singer used a parsed corpus and classified PNs that appeared in certain syntactic contexts. Cucerzan and Yarowsky identified and classified PNs in seven languages, learning character-based contextual, internal, and morphological patterns. Their algorithm does not strictly require capitalization cues, but recall was much lower for the language without case distinctions (Hindi).

[2]Abney (2002) gives a theoretical analysis of bootstrapping and co-training algorithms.

A number of bootstrapping algorithms have been developed to learn semantic classes that may contain common nouns or GNs as well as PNs. Phillips and Riloff (2002) and others relied on structures such as appositives and compound nouns. Riloff and Jones (1999) and Thelen and Riloff (2002) learned contextual patterns that predict the semantic class of the subject, direct object, or prepositional phrase object. Strzalkowski and Wang (1996) and Yangarber et al. (2002) used windows of tokens to learn contextual and internal patterns without parsing. Yangarber et al. enabled discovery of GNs embedded in larger noun groups (such as *yellow fever* in "10 yellow fever cases").

3. Learning Algorithm

Our algorithm, Nomen, is adapted from Yangarber et al. (2002), with modifications in steps 4 (the confidence formula) and 6 (the balance requirement and the number of names acquired per iteration).

3.1. Pre-Processing

The training corpus is passed through a zoner, a sentence splitter, a tokenizer/lemmatizer, and a part-of-speech (POS) tagger. The zoner extracts the textual content, removing special segments such as e-mail headers or newspaper headlines. The tokenizer separates the text into words or lexical units. For English text, a lemmatizer converts the surface word forms to lemmas (base forms).

3.2. Bootstrapped Learning

0. Seeds: We initialize several learners to run simultaneously, one for each semantic class. The user must provide seed examples for each learner. E.g., in the domain of infectious diseases, we used the 10 most common names as seeds for the classes of diseases and locations.

The set of *accepted names* is initialized with the seeds, for each learner.

1. Tag Accepted Names: For each accepted name, mark each occurrence in the corpus with an enclosing pair of category tags, e.g., `<disease>` and `</disease>`.

2. Generate Patterns: For each tag T inserted in Step 1, we generate a *literal* pattern p using a context window of width w around the tag, e.g.,

$$p = [\ l_{-3}\ l_{-2}\ l_{-1}\ \texttt{<T>}\ l_{+1}\ l_{+2}\ l_{+3}\ ]$$

where $l_{\pm i}$ are the *context* of p—the lemmas of the surrounding words. Two literal patterns are generated for each name occurrence, one for the left tag, and one for the right.

The literal pattern p is then generalized by replacing some of the elements in the context window by wildcards. The generalized patterns form the set of *candidate* patterns. Note that each pattern matches on only one side of an instance, the left or the right.

3. Evaluate Patterns: For every learner, match each candidate pattern p against the training corpus. Wherever the context of p matches, p predicts the occurrence of the left or right boundary of a name. The learner then uses a noun group (NG) regular expression to determine the other, *partner* boundary.[3] Thus, if p predicts a left boundary, the regular expression determines the right boundary, and vice versa.

Next, the learner checks whether the NG between these boundaries has already been accepted as a name by any learner; the NG can be:

- *positive example*: already accepted as a name by the same learner;
- *negative example*: already accepted as a name by a different learner;
- *unknown*: not yet accepted as a name by any learner.

The *unknown* case is where a new candidate name may potentially be discovered.

4. Acquire Patterns: For each candidate pattern p, we compile three lists of *types* (distinct NGs) matched by p: the positive, negative, and unknown examples, or $pos(p)$, $neg(p)$, and $unk(p)$.

We then compute the pattern's *accuracy* and *confidence*:

$$\begin{aligned} acc(p) &= \frac{|pos(p)|}{|pos(p)| + |neg(p)|} \\ conf(p) &= \frac{|pos(p)| - |neg(p)|}{|pos(p)| + |neg(p)| + |unk(p)|} \end{aligned}$$

Patterns with accuracy below a threshold θ_{prec} are discarded. The remaining patterns are ranked by:

$$Score(p) = conf(p) \cdot \log |pos(p)| \qquad (1)$$

(Thus, to get a positive score, a pattern must have at least two distinct NGs as positive examples, and more positive than negative examples.) The n top-scoring patterns for each learner are added to its set of *accepted patterns.*

5. Apply Patterns: For each accepted pattern p, the noun groups in the set $unk(p)$ become *candidate names.*

6. Acquire Names: The learner scores each candidate name t, based on how many *different* accepted patterns match the instances of t, and how reliable these patterns are.

Let M_t be the set of accepted patterns which match any of the instances of t. We require that M_t have sufficient

- mass[4]: $|M_t| \geq 2$,
- balance[5]: M_t contains at least one pattern predicting the left boundary of t and one pattern predicting the right boundary.

Compute $Rank(t)$ based on the quality of M_t:

$$Rank(t) = 1 - \prod_{p \in M_t} \Big(1 - conf(p)\Big) \qquad (2)$$

This formula favors candidates matched by more patterns or more reliable patterns.

Each learner acquires the top-scoring k percent (here, 5%) of the candidate names, up to a maximum of m names (here, 5 names), on each iteration.[6]

7. Repeat: from Step 1, until no more names can be learned.

4. Experiments

We used NOMEN to discover names in a specialized corpus (in English) and two general news corpora (in English and Chinese).

The specialized corpus was drawn from the ProMED mailing list, a global forum for public health professionals reporting outbreaks of infectious diseases. The corpus contains 600,000 words from 1,400 postings.

[3]This is why POS tagging is needed. We use simple NG regular expression heuristics, similar to those used in terminology discovery, e.g., `[Adj* Noun+]` (Frantzi, Ananiadou, & Mima, 2000).

[4]Because of the mass requirement, the algorithm is less likely to learn a name that appears only once in the corpus.

[5]The balance requirement is waived if at least two patterns in M_t are *contextual* (matching lemmas outside the name boundaries). The NG heuristic to find the partner boundary is less reliable when the pattern is *internal* (matching the initial or final lemmas of the name).

[6]If k% of all candidates is less than one (and greater than zero), then the learner acquires the single top-scoring name.

Appropriate evaluation strategies may depend on the algorithm's intended use. If the goal is to build a lexicon of names of a given class, a type-based evaluation may be more appropriate, measuring how many of the *distinct* names in the corpus were correctly discovered, and how accurate the list of discovered names is.

On the other hand, if the algorithm is to be used as a name tagger, one may be more interested in measuring how many of the *instances* of names in the corpus are identified correctly. Instance-based evaluation was used in the Message Understanding Conferences (MUCs) and in recent work on unsupervised name tagging (Collins & Singer, 1999; Cucerzan & Yarowsky, 1999, 2002).

Yangarber et al. (2002) discussed the differences between type- and instance-based evaluation in an earlier experiment on the ProMED corpus.

4.1. Learning Disease and Location Names

The experiment shown in Figure 1 focuses on building lexicons of disease and location names from the ProMED corpus. For measuring the type-based performance we first compiled two reference lists of names for each class: a recall list and a precision list.

First, a *manual list* of disease and location names was compiled from multiple databases. NOMEN's recall is judged by how well it finds these names, if they occur in the corpus. The manual list was filtered through the corpus: names that appear in the corpus two or more times were placed in the *recall list*. We focused on recall for names that occur two or more times because the algorithm is inherently discouraged from learning names that appear only once in the corpus (cf. the *mass* criterion in Section 3.2, step 6).

To measure precision we constructed the *precision list*, which contains the manual list and an automatically generated set of disease acronyms.[7]

The recall list contains 322 disease names and 641 location names; the precision list contains 3,867 disease names and 2,404 location names.

Figure 1 shows NOMEN's performance, with recall judged against the recall list and precision against the precision list.[8]

[7]Because this list is incomplete, precision is understated: NOMEN is penalized for learning some correct names that are not in the precision list.

[8]For completeness, we also computed recall against *all* names in the manual list that appear in the corpus, *one* or more times. The final recall was 61% for diseases and 60% for locations, compared with the 74% for both classes shown in Figure 1.

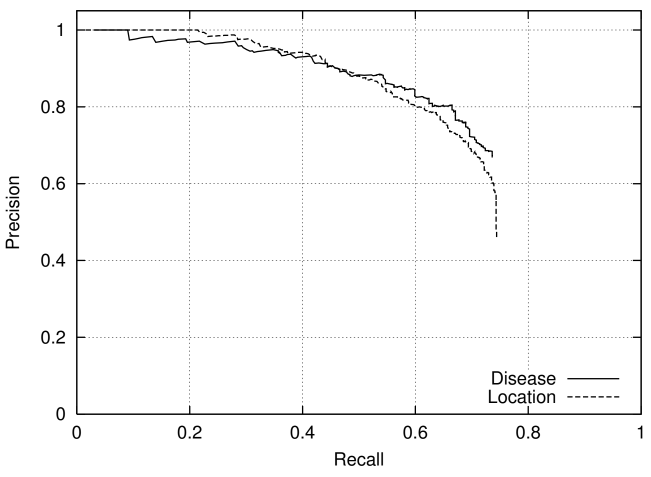

Figure 1. Disease and location names: type-based evaluation.

The parameters in these experiments are: 10 seeds per category, context window width $w = 3$, pattern accuracy threshold $\theta_{prec} = 0.5$, and $n = m = 5$ for the maximum number of patterns and names learned per iteration. Six negative classes were learned, as explained in Section 4.2.

4.2. Competing Categories

There are three reasons why competition between semantic categories can improve bootstrapped learning of names and patterns:

- As observed by Thelen and Riloff (2002), a category is less likely to expand beyond its true territory if it cannot acquire names that have already been claimed by other categories.
- The accepted names in each category serve as negative examples for all other categories. Learners avoid acquiring patterns with too many negative examples (Section 3.2, step 4).
- Conversely, accepted patterns can provide negative evidence against other categories' candidate names. We have experimented with formulas that consider such negative evidence and have not yet found a consistent performance gain, but more exploration is warranted.[9]

Figure 2 shows the effect of competition on the disease name learner. When running unopposed (*"Disease only"*), the learner quickly picks up non-specific

[9]Thelen and Riloff do not use negative examples to score patterns but do find a small improvement from using negative evidence to score words or names.

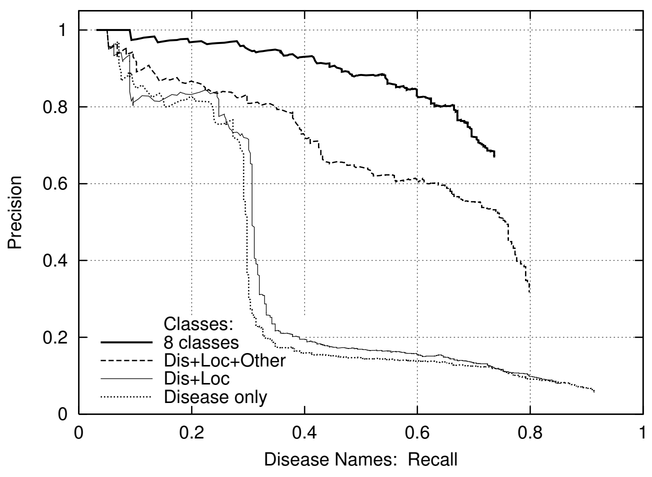

Figure 2. Effect of competing categories on the disease name learner.

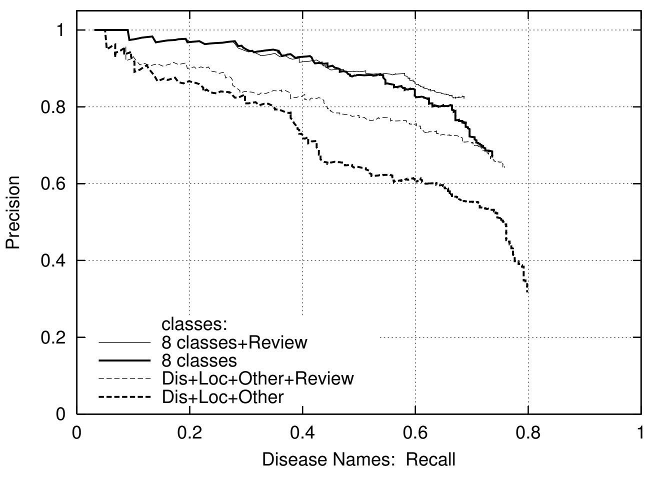

Figure 3. Simulation of learning with review.

patterns that may match many diseases but also match irrelevant NGs. This leads it to acquire a very large number of names, at very low precision. Adding the location learner (*"Dis+Loc"*) helps the disease learner only slightly.

To provide more guiding evidence, we introduce a *negative learner* that acquires noun phrases belonging to neither of the target classes. As negative seeds, we use the 10 most frequent NGs in the corpus.[10] The corresponding curve, *"Dis+Loc+Other,"* shows a substantial improvement in performance.

Still better performance is obtained when we split the negative class into six competing categories: symptom, animal, human, institution, time, and other. The result is shown in both Figure 1 (the *"Disease"* curve) and Figure 2 (the *"8 classes"* curve). The final recall and precision are around 70%.

4.3. Learning with Review

The algorithm as presented so far is unsupervised, except for the information in the seed examples. The next question we explore is: how can we help the algorithm if we add a human reviewer who would check each new name discovered on the current iteration, and if the name is not of the correct class, tell Nomen to reject it.

We designed an experiment to simulate human review. On each iteration, the disease and location name learners submit for review the names they would normally acquire. The simulated reviewer rejects submissions that do not match the category's precision list. If a name is rejected, the learner does not acquire it on the present iteration or at any later time.

Figure 3 shows the simulated effect of review on the disease name learner. The curve labeled *"Dis+Loc+Other"* is the same as in Figure 2. *"Dis+Loc+Other+Review"* shows that with three learners and review, performance improves to about the level of having eight competing learners.[11] One way to view this finding is that the five additional learners are able to compensate for the absence of a human in the loop. Note, though, that specifying the additional categories and seeds took some thought and time.

Adding review on top of the eight classes yields only a slight improvement in precision and reduces recall somewhat. However, the simulation may understate the benefits of review. The simulated reviewer rejects some correct names because they are not in the precision list. This may prevent the learners from acquiring useful patterns associated with those names. Also, a real reviewer could suggest the correct category for incorrect submissions, instead of just rejecting them.

4.4. Name Tagging in English

For the experiments on general news corpora, we performed instance-based evaluation (cf. the beginning of Section 4), using the MUC scoring software (Defense Advanced Research Projects Agency, 1995).

Name instances are marked with SGML tags in the

[10]We exclude disease and location names and related generic NGs ("illness," "area"). The negative seeds are: case, health, day, people, year, patient, death, number, report, farm.

[11]The precision measure is the percentage of submissions from the learner that are accepted by the reviewer.

answer key and in NOMEN's response. Each name instance receives two scores, for *type* and *text*. The type score measures whether the category of the name is correct in the response. The text score measures whether the boundaries of the name were correctly identified. E.g., if the answer key contains `<person>Herbert G. Wells</person>` but the program tags only `Wells` as a *person*, then it will get 1 for the type score and 0 for the text. The overall score is the average of type and text.

We made some modifications to the standard MUC scoring scheme. In MUC scoring, if the program tags `Herbert G. Wells` as a *location*, it will get 0 for type but 1 for text. Our scoring is more conservative: the text score is 0 unless both text and type are correct.[12]

The training corpus for this experiment consists of 3,800 documents (3 million words) from the New York Times News Service, 1996. These include the 200 annotated documents (150,000 words) of the MUC-7 dry-run and formal training corpora, which we use as our test set. The annotations are invisible to NOMEN.

We ran learners for names of persons, organizations, and locations. The NG regular expression (Section 3.2, step 3) is `[Adj* Noun+]` with all words capitalized.[13] Name boundary determination for left-boundary contextual patterns is handled differently: The learner first finds the maximal NG, regardless of case. It then searches for a *capitalized* NG inside the maximal NG, with the same right boundary. E.g., when the pattern "said *X*" is applied to the text "said FAA spokeswoman Mary Culver," it extracts *Mary Culver*, not *FAA*.

For location seed names, we use six countries, three states, and two cities. For persons, we use "Bill Clinton", "Bob Dole", "Newt Gingrich", and all capitalized NGs immediately preceded by "Dr." anywhere in the corpus. For organizations, we use "White House", "Congress", and all capitalized NGs that end in "Corp."

Figure 4 shows the text and type curves for the instance-based evaluation on the three categories. In general, the curves achieve high recall with very slow decline in precision. Note the curious reversal in text precision for person names. The learner initially tends to acquire first or last names *separately*, which causes a sharp drop in text precision. However, toward the end, it proceeds to learn enough *full* names to recover from the initial precision loss.

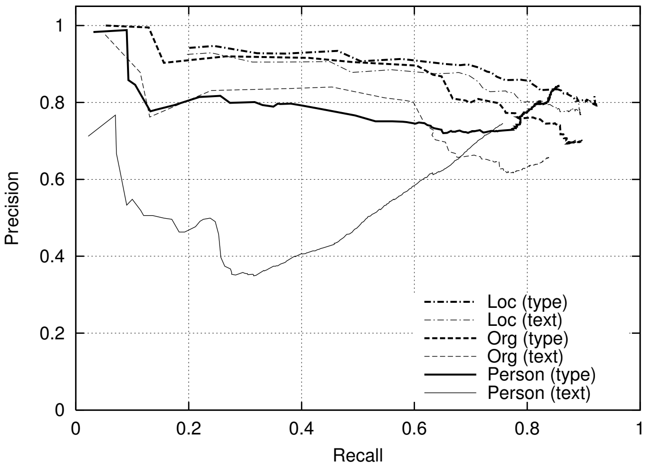

Figure 4. Proper names in English: instance-based evaluation.

The "English" columns of Table 1 summarize recall and precision before and after bootstrapping. Final type recall is in the 86–92% range, with reasonable precision (above 70%).

4.5. Name Tagging in Chinese

We have also begun to use NOMEN to learn proper names in Chinese text. The training corpus consists of 700,000 words from the Beijing University Institute of Computational Linguistics corpus (articles from the People's Daily, January 1998).

The Beijing annotators segmented the text into words, attached part-of-speech tags, and labeled the person, organization, and location names. We use the named entity labels for evaluation. In our initial development environment we rely on the annotators' word segmentation and POS tags, which were influenced by their knowledge of names. In future research we plan to use automated segmentation and POS tags, which will be messier but more realistic.

Because of some omissions in the NE annotations (e.g., single-word abbreviations were not labeled as NEs), we asked two colleagues to revise the NE tags for 41 documents (10,000 words), which we use as our test set.

Because case distinctions do not exist in Chinese,

[12]The original MUC scheme computes category-specific recall and precision for type but not for text; misclassifying a person as a location hurts type recall and precision for persons (not locations). In our scheme, the same error hurts recall for persons and precision for locations, on both type and text.

[13]Sentence-initial capitals are treated as lowercase if the non-sentence-initial occurrences of the bigram or token in the corpus are mostly lowercase. Hyphenated words are assigned the case of the first token and the POS tag of the last.

Table 1. Proper name tagging: instance-based evaluation.

		English		**Chinese**			
				–Negative		**+Negative**	
		Rec	Prec	Rec	Prec	Rec	Prec
Initial (seeds)							
Person	type	3.1	98.3	17.8	100.0	*same*	
	text	*2.2*	*71.2*	*17.8*	*100.0*		
Org	type	5.3	100.0	25.4	75.9		
	text	*5.2*	*97.6*	*20.8*	*62.1*		
Location	type	20.1	94.2	39.8	87.1		
	text	*19.7*	*92.5*	*39.4*	*86.4*		
Final							
Person	type	85.7	84.4	63.1	75.1	67.1	83.9
	text	*75.7*	*74.5*	*47.1*	*56.1*	*51.6*	*64.4*
Org	type	89.6	70.2	54.9	16.5	48.0	54.6
	text	*83.9*	*65.7*	*37.6*	*11.3*	*35.8*	*40.8*
Location	type	91.8	82.2	77.6	61.9	86.3	67.6
	text	*89.1*	*79.7*	*65.8*	*52.5*	*79.2*	*62.0*

the patterns learned may identify common nouns and phrases as well as proper names in the target categories. For applications such as information extraction, finding generic terms of a given class may be as important as finding PNs. However, generics are not labeled in the test set, so our evaluation penalizes Nomen for tagging generic person, organization, and location phrases.

In the absence of case distinctions, a proper name tagger can avoid tagging common nouns that appear in a basic dictionary. (This approach does not work for multiword phrases, since many organization names are composed of common nouns.) To approximate a dictionary check for common nouns, we prevented the learners from acquiring words that the annotators had ever POS-tagged as nouns and never labeled as NEs. (The NE annotations were otherwise invisible to the learners.)

We ran a negative learner in addition to the name learners. Each learner was given 10 seeds.[14] The pattern accuracy threshold is $\theta_{prec} = 0.9$, and the NG regular expression is `[Noun+]`.[15]

Recall and precision are shown in the "Chinese: +Negative" columns of Table 1. For the type scores, we also computed an adjusted measure (based on manual review) that does not penalize generic phrases in the correct category. The final adjusted precision scores (not shown in the table) are 87% for persons, 76% for organizations, and 72% for locations.

The columns labeled "Chinese: –Negative" show the effect of omitting the negative learner. Type precision for the organization learner plunges from 55% to 17%.

[14]Negative seeds: Shiwuda spirit, economic development, spiritual civilization construction, bi-coastal relations, cooperative relations, economic installations, financial crisis, national relations, Party-style cultivation of clean government, struggle against corruption.

[15]With a left-boundary contextual pattern, the NG must not be followed by the possessive/attributive particle *de*.

5. Conclusion

We discussed an algorithm for unsupervised learning of semantic classes. The algorithm shows promise due to its applicability in several different settings and good performance levels. It requires very limited linguistic information, which enhances portability. A central idea is combining multiple learners in parallel, including a negative learner. The learners provide negative evidence to each other, which improves precision.

Acknowledgements

This research is supported by the Defense Advanced Research Projects Agency as part of the Translingual Information Detection, Extraction and Summarization (TIDES) program, under Grant N66001-001-1-8917 from the Space and Naval Warfare Systems Center San Diego, and by the National Science Foundation under Grant IIS-0081962. Shubin Zhao and Heng Ji gave valuable suggestions and help for our work with the Beijing corpus. Two anonymous reviewers provided helpful comments on an earlier version.

References

Abney, S. (2002). Bootstrapping. *Proceedings of the Fortieth Annual Meeting of the Association for Computational Linguistics*. San Francisco: Morgan Kaufmann.

Bikel, D., Miller, S., Schwartz, R., & Weischedel, R. (1997). Nymble: a high-performance learning name-finder. *Proceedings of the Fifth Applied Natural Language Processing Conference*. San Francisco: Morgan Kaufmann.

Collins, M., & Singer, Y. (1999). Unsupervised models for named entity classification. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*. New Brunswick, NJ: Association for Computational Linguistics.

Cucerzan, S., & Yarowsky, D. (1999). Language independent named entity recognition combining morphological and contextual evidence. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*. New Brunswick, NJ: Association for Computational Linguistics.

Cucerzan, S., & Yarowsky, D. (2002). Language independent NER using a unified model of internal and contextual evidence. *Proceedings of the Sixth Conference on Natural Language Learning*. San Francisco: Morgan Kaufmann.

Defense Advanced Research Projects Agency (1995). *Proceedings of the Sixth Message Understanding Conference*. San Francisco: Morgan Kaufmann.

Frantzi, K., Ananiadou, S., & Mima, H. (2000). Automatic recognition of multi-word terms: the C-value/NC-value method. *International Journal on Digital Libraries*, *3*, 115–130.

Phillips, W., & Riloff, E. (2002). Exploiting strong syntactic heuristics and co-training to learn semantic lexicons. *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. New Brunswick, NJ: Association for Computational Linguistics.

Riloff, E., & Jones, R. (1999). Learning dictionaries for information extraction by multi-level bootstrapping. *Proceedings of the Sixteenth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Strzalkowski, T., & Wang, J. (1996). A self-learning universal concept spotter. *Proceedings of the Sixteenth International Conference on Computational Linguistics*. San Francisco: Morgan Kaufmann.

Thelen, M., & Riloff, E. (2002). A bootstrapping method for learning semantic lexicons using extraction pattern contexts. *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. New Brunswick, NJ: Association for Computational Linguistics.

Yangarber, R. (2003). Counter-training in discovery of semantic patterns. *Proceedings of the Forty-First Annual Meeting of the Association for Computational Linguistics*. San Francisco: Morgan Kaufmann.

Yangarber, R., Lin, W., & Grishman, R. (2002). Unsupervised learning of generalized names. *Proceedings of the Nineteenth International Conference on Computational Linguistics*. San Francisco: Morgan Kaufmann.