A Context Pattern Induction Method for Named Entity Extraction

Partha Pratim Talukdar
CIS Department
University of Pennsylvania
Philadelphia, PA 19104
partha@cis.upenn.edu

Thorsten Brants
Google, Inc.
1600 Amphitheatre Pkwy.
Mountain View, CA 94043
brants@google.com

Mark Liberman Fernando Pereira
CIS Department
University of Pennsylvania
Philadelphia, PA 19104
{myl,pereira}@cis.upenn.edu

Abstract

We present a novel context pattern induction method for information extraction, specifically named entity extraction. Using this method, we extended several classes of seed entity lists into much larger high-precision lists. Using token membership in these extended lists as additional features, we improved the accuracy of a conditional random field-based named entity tagger. In contrast, features derived from the seed lists decreased extractor accuracy.

1 Introduction

Partial entity lists and massive amounts of unlabeled data are becoming available with the growth of the Web as well as the increased availability of specialized corpora and entity lists. For example, the primary public resource for biomedical research, MEDLINE, contains over 13 million entries and is growing at an accelerating rate. Combined with these large corpora, the recent availability of entity lists in those domains has opened up interesting opportunities and challenges. Such lists are never complete and suffer from sampling biases, but we would like to exploit them, in combination with large unlabeled corpora, to speed up the creation of information extraction systems for different domains and languages. In this paper, we concentrate on exploring utility of such resources for named entity extraction.

Currently available entity lists contain a small fraction of named entities, but there are orders of magnitude more present in the unlabeled data[1]. In this paper, we test the following hypotheses:

i. Starting with a few seed entities, it is possible to induce high-precision context patterns by exploiting entity context redundancy.

ii. New entity instances of the same category can be extracted from unlabeled data with the induced patterns to create high-precision extensions of the seed lists.

iii. Features derived from token membership in the extended lists improve the accuracy of learned named-entity taggers.

Previous approaches to context pattern induction were described by Riloff and Jones (1999), Agichtein and Gravano (2000), Thelen and Riloff (2002), Lin et al. (2003), and Etzioni et al. (2005), among others. The main advance in the present method is the combination of grammatical induction and statistical techniques to create high-precision patterns.

The paper is organized as follows. Section 2 describes our pattern induction algorithm. Section 3 shows how to extend seed sets with entities extracted by the patterns from unlabeled data. Section 4 gives experimental results, and Section 5 compares our method with previous work.

[1] For example, based on approximate matching, there is an overlap of only 22 organizations between the 2403 organizations present in CoNLL-2003 shared task training data and the Fortune-500 list.

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 141–148, New York City, June 2006.

2 Context Pattern Induction

The overall method for inducing entity context patterns and extending entity lists is as follows:

1. Let E = seed set, T = text corpus.
2. Find the contexts C of entities in E in the corpus T (Section 2.1).
3. Select *trigger words* from C (Section 2.2).
4. For each trigger word, induce a pattern automaton (Section 2.3).
5. Use induced patterns P to extract more entities E' (Section 3).
6. Rank P and E' (Section 3.1).
7. If needed, add high scoring entities in E' to E and return to step 2. Otherwise, terminate with patterns P and extended entity list $E \cup E'$ as results.

2.1 Extracting Context

Starting with the seed list, we first find occurrences of seed entities in the unlabeled data. For each such occurrence, we extract a fixed number W (context window size) of tokens immediately preceding and immediately following the matched entity. As we are only interested in modeling the context here, we replace all entity tokens by the single token `-ENT-`. This token now represents a *slot* in which an entity can occur. Examples of extracted entity contexts are shown in Table 1. In the work presented in this papers, seeds are entity instances (e.g. *Google* is a seed for organization category).

increased expression of `-ENT-` in vad mice
the expression of `-ENT-` mrna was greater
expression of the `-ENT-` gene in mouse

Table 1: Extracted contexts of known genes with $W = 3$.

The set of extracted contexts is denoted by C. The next step is to automatically induce high-precision patterns containing the token `-ENT-` from such extracted contexts.

2.2 Trigger Word Selection

To induce patterns, we need to determine their starts. It is reasonable to assume that some tokens are more specific to particular entity classes than others. For example, in the examples shown above, *expression* can be one such word for gene names. Whenever one comes across such a token in text, the probability of finding an entity (of the corresponding entity class) in its vicinity is high. We call such starting tokens *trigger words*. Trigger words mark the beginning of a pattern. It is important to note that simply selecting the first token of extracted contexts may not be a good way to select trigger words. In such a scheme, we would have to vary W to search for useful pattern starts. Instead of that brute-force technique, we propose an automatic way of selecting trigger words. A good set of trigger words is very important for the quality of induced patterns. Ideally, we want a trigger word to satisfy the following:

- It is frequent in the set C of extracted contexts.
- It is specific to entities of interest and thereby to extracted contexts.

We use a term-weighting method to rank candidate trigger words from entity contexts. IDF (Inverse Document Frequency) was used in our experiments but any other suitable term-weighting scheme may work comparably. The IDF weight f_w for a word w occurring in a corpus is given by:

$$f_w = \log\left(\frac{N}{n_w}\right)$$

where N is the total number of documents in the corpus and n_w is the total number of documents containing w. Now, for each context segment $c \in C$, we select a *dominating word* d_c given by

$$d_c = \arg\max_{w \in c} f_w$$

There is exactly one dominating word for each $c \in C$. All dominating words for contexts in C form multiset M. Let m_w be the multiplicity of the dominating word w in M. We sort M by decreasing m_w and select the top n tokens from this list as potential trigger words.

Selection criteria based on dominating word frequency work better than criteria based on simple term weight because high term weight words may be rare in the extracted contexts, but would still be misleadingly selected for pattern induction. This can be avoided by using instead the frequency of dominating words within contexts, as we did here.

2.3 Automata Induction

Rather than using individual contexts directly, we summarize them into automata that contain the most significant regularities of the contexts sharing a given trigger word. This construction allows us to determine the relative importance of different context features using a variant of the forward-backward algorithm from HMMs.

2.3.1 Initial Induction

For each trigger word, we list the contexts starting with the word. For example, with *"expression"* as the trigger word, the contexts in Table 1 are reduced to those in Table 2. Since *"expression"* is a left-context trigger word, only one token to the right of `-ENT-` is retained. Here, the predictive context lies to the left of the slot `-ENT-` and a single token is retained on the right to mark the slot's right boundary. To model predictive right contexts, the token string can be reversed and the same techniques as here applied on the reversed string.[2]

expression of `-ENT-` *in*
expression of `-ENT-` *mrna*
expression of the `-ENT-` *gene*

Table 2: Context segments corresponding to trigger word *"expression"*.

Similar contexts are prepared for each trigger word. The context set for each trigger word is then summarized by a pattern automaton with transitions that match the trigger word and also the wildcard `-ENT-`. We expect such automata to model the position in context of the entity slot and help us extract more entities of the same class with high precision.

[2]Experiments reported in this paper use predictive left context only.

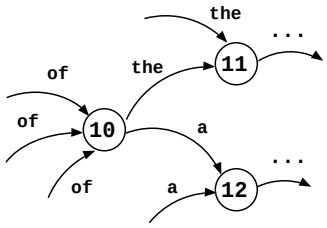

Figure 1: Fragment of a 1-reversible automaton

We use a simple form of grammar induction to learn the pattern automata. Grammar induction techniques have been previously explored for information extraction (IE) and related tasks. For instance, Freitag (1997) used grammatical inference to improve precision in IE tasks.

Context segments are short and typically do not involve recursive structures. Therefore, we chose to use 1-reversible automata to represent sets of contexts. An automaton A is *k-reversible* iff (1) A is deterministic and (2) A^r is deterministic with k tokens of lookahead, where A^r is the automaton obtained by reversing the transitions of A. Wrapper induction using *k-reversible* grammar is discussed by Chidlovskii (2000).

In the 1-reversible automaton induced for each trigger word, all transitions labeled by a given token go to the same state, which is identified with that token. Figure 1 shows a fragment of a 1-reversible automaton. Solan et al. (2005) describe a similar automaton construction, but they allow multiple transitions between states to distinguish among sentences.

Each transition $e = (v, w)$ in a 1-reversible automaton A corresponds to a bigram vw in the contexts used to create A. We thus assign each transition the probability

$$P(w|v) = \frac{C(v,w)}{\Sigma_{w'} C(v,w')}$$

where $C(v, w)$ is the number of occurrences of the bigram vw in contexts for W. With this construction, we ensure words will be credited in proportion to their frequency in contexts. The automaton may overgenerate, but that potentially helps generalization.

2.3.2 Pruning

The initially induced automata need to be pruned to remove transitions with weak evidence so as to increase match precision.

The simplest pruning method is to set a count threshold c below which transitions are removed. However, this is a poor method. Consider state 10 in the automaton of Figure 2, with $c = 20$. Transitions $(10, 11)$ and $(10, 12)$ will be pruned. $C(10, 12) \ll c$ but $C(10, 11)$ just falls short of c. However, from the transition counts, it looks like the sequence *"the -ENT-"* is very common. In such a case, it is not desirable to prune $(10, 11)$. Using a local threshold may lead to overpruning.

We would like instead to keep transitions that are used in relatively many probable paths through the automaton. The probability of path p is $P(p) = \prod_{(v,w)\in p} P(w|v)$. Then the posterior probability of edge (v, w) is

$$P(v, w) = \frac{\sum_{(v,w)\in p} P(p)}{\sum_p P(p)} \quad ,$$

which can be efficiently computed by the forward-backward algorithm (Rabiner, 1989). We can now remove transitions leaving state v whose posterior probability is lower than $p_v = k(\max_w P(v, w))$, where $0 < k \leq 1$ controls the degree of pruning, with higher k forcing more pruning. All induced and pruned automata are trimmed to remove unreachable states.

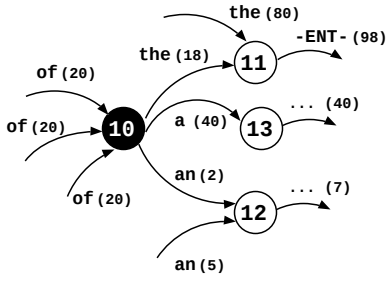

Figure 2: Automaton to be pruned at state 10. Transition counts are shown in parenthesis.

3 Automata as Extractor

Each automaton induced using the method described in Sections 2.3-2.3.2 represents high-precision patterns that start with a given trigger word. By scanning unlabeled data using these patterns, we can extract text segments which can be substituted for the slot token `-ENT-`. For example, assume that the induced pattern is *"analyst at -ENT- and"* and that the scanned text is *"He is an analyst at the University of California and ..."*. By scanning this text using the pattern mentioned above, we can figure out that the text *"the University of California"* can substitute for "`-ENT-`". This extracted segment is a candidate extracted entity. We now need to decide whether we should retain all tokens inside a candidate extraction or purge some tokens, such as *"the"* in the example.

One way to handle this problem is to build a language model of content tokens and retain only the maximum likelihood token sequence. However, in the current work, the following heuristic which worked well in practice is used. Each token in the extracted text segment is labeled either *keep* (`K`) or *droppable* (`D`). By default, a token is labeled `K`. A token is labeled `D` if it satisfies one of the droppable criteria. In the experiments reported in this paper, droppable criteria were whether the token is present in a stopword list, whether it is non-capitalized, or whether it is a number.

Once tokens in a candidate extraction are labeled using the above heuristic, the longest token sequence corresponding to the regular expression $\mathtt{K}[\mathtt{D}\ \mathtt{K}]*\mathtt{K}$ is retained and is considered a final extraction. If there is only one `K` token, that token is retained as the final extraction. In the example above, the tokens are labeled *"the/D University/K of/D California/K"*, and the extracted entity will be *"University of California"*.

To handle run-away extractions, we can set a domain-dependent hard limit on the number of tokens which can be matched with "`-ENT-`". This stems from the intuition that useful extractions are not very long. For example, it is rare that a person name longer than five tokens.

3.1 Ranking Patterns and Entities

Using the method described above, patterns and the entities extracted by them from unlabeled data are paired. But both patterns and extractions vary in quality, so we need a method for ranking both. Hence, we need to rank both patterns and entities. This is difficult given that there we have no nega-

tive labeled data. Seed entities are the only positive instances that are available.

Related previous work tried to address this problem. Agichtein and Gravano (2000) seek to extract relations, so their pattern evaluation strategy considers one of the attributes of an extracted tuple as a key. They judge the tuple as a positive or a negative match for the pattern depending on whether there are other extracted values associated with the same key. Unfortunately, this method is not applicable to entity extraction.

The pattern evaluation mechanism used here is similar in spirit to those of Etzioni et al. (2005) and Lin et al. (2003). With seeds for multiple classes available, we consider seed instances of one class as negative instances for the other classes. A pattern is penalized if it extracts entities which belong to the seed lists of the other classes. Let $\mathrm{pos}(p)$ and $\mathrm{neg}(p)$ be respectively the number of distinct positive and negative seeds extracted by pattern p. In contrast to previous work mentioned above, we do not combine $\mathrm{pos}(p)$ and $\mathrm{neg}(p)$ to calculate a single accuracy value. Instead, we discard all patterns p with positive $\mathrm{neg}(p)$ value, as well as patterns whose total positive seed (distinct) extraction count is less than certain threshold η_{pattern}. This scoring is very conservative. There are several motivations for such a conservative scoring. First, we are more interested in precision than recall. We believe that with massive corpora, large number of entity instances can be extracted anyway. High accuracy extractions allow us to reliably (without any human evaluation) use extracted entities in subsequent tasks successfully (see Section 4.3). Second, in the absence of sophisticated pattern evaluation schemes (which we are investigating — Section 6), we feel it is best to heavily penalize any pattern that extracts even a single negative instance.

Let G be the set of patterns which are retained by the filtering scheme described above. Also, let $I(e, p)$ be an indicator function which takes value 1 when entity *e* is extracted by pattern *p* and 0 otherwise. The score of e, $S(e)$, is given by

$$S(e) = \Sigma_{p \in G} I(e, p)$$

This whole process can be iterated by including extracted entities whose score is greater than or equal to a certain threshold η_{entity} to the seed list.

4 Experimental Results

For the experiments described below, we used 18 billion tokens (31 million documents) of news data as the source of unlabeled data. We experimented with 500 and 1000 trigger words. The results presented were obtained after a single iteration of the Context Pattern Induction algorithm (Section 2).

4.1 English LOC, ORG and PER

For this experiment, we used as seed sets subsets of the entity lists provided with CoNLL-2003 shared task data.[3] Only multi-token entries were included in the seed lists of respective categories (location (LOC), person (PER) & organization (ORG) in this case). This was done to partially avoid incorrect context extraction. For example, if the seed entity is *"California"*, then the same string present in *"University of California"* can be incorrectly considered as an instance of LOC. A stoplist was used for dropping tokens from candidate extractions, as described in Section 3. Examples of top ranking induced patterns and extracted entities are shown in Table 9. Seed list sizes and experimental results are shown in Table 3. The precision numbers shown in Table 3 were obtained by manually evaluating 100 randomly selected instances from each of the extended lists.

Category	Seed Size	Patterns Used	Extended Size	Precision
LOC	379	29	3001	70%
ORG	1597	276	33369	85%
PER	3616	265	86265	88%

Table 3: Results of LOC, ORG & PER entity list extension experiment with η_{pattern} = 10 set manually.

The overlap[4] between the induced ORG list and the Fortune-500 list has 357 organization names, which is significantly higher than the seed list overlap of 22 (see Section 1). This shows that we have been able to improve coverage considerably.

4.2 Watch Brand Name

A total of 17 watch brand names were used as seeds. In addition to the pattern scoring scheme

[3] A few locally available entities in each category were also added. These seeds are available upon request from the authors.

[4] Using same matching criteria as in Section 1.

of Section 3.1, only patterns containing sequence *"watch"* were finally retained. Entities extracted with $\eta_{\text{entity}} = 2$ are shown in Table 5. Extraction precision is 85.7%.

Corum, Longines, Lorus, Movado, Accutron, Audemars Piguet, Cartier, Chopard, Franck Muller, IWC, Jaeger-LeCoultre, A. Lange & Sohne, Patek Philippe, Rolex, Ulysse, Nardin, Vacheron Constantin

Table 4: Watch brand name seeds.

Rolex	Fossil	Swatch
Cartier	Tag Heuer	Super Bowl
Swiss	Chanel	SPOT
Movado	Tiffany	Sekonda
Seiko	TechnoMarine	Rolexes
Gucci	Franck Muller	Harry Winston
Patek Philippe	Versace	Hampton Spirit
Piaget	Raymond Weil	Girard Perregaux
Omega	Guess	Frank Mueller
Citizen	Croton	David Yurman
Armani	Audemars Piguet	Chopard
DVD	DVDs	Chinese
Breitling	Montres Rolex	Armitron
Tourneau	CD	NFL

Table 5: Extended list of watch brand names after single iteration of pattern induction algorithm.

This experiment is interesting for several reasons. First, it shows that the method presented in this paper is effective even with small number of seed instances. From this we conclude that the unambiguous nature of seed instances is much more important than the size of the seed list. Second, no negative information was used during pattern ranking in this experiment. This suggests that for relatively unambiguous categories, it is possible to successfully rank patterns using positive instances only.

4.3 Extended Lists as Features in a Tagger

Supervised models normally outperform unsupervised models in extraction tasks. The downside of supervised learning is expensive training data. On the other hand, massive amounts of unlabeled data are readily available. The goal of semi-supervised learning to combine the best of both worlds. Recent research have shown that improvements in supervised taggers are possible by including features derived from unlabeled data (Miller et al., 2004; Liang, 2005; Ando and Zhang, 2005). Similarly, automatically generated entity lists can be used as additional features in a supervised tagger.

System	F1 (Precision, Recall)
Florian et al. (2003), best single, no list	89.94 (91.37, 88.56)
Zhang and Johnson (2003), no list	90.26 (91.00, 89.53)
CRF baseline, no list	89.52 (90.39, 88.66)

Table 6: Baseline comparison on 4 categories (LOC, ORG, PER, MISC) on Test-a dataset.

For this experiment, we started with a conditional random field (CRF) (Lafferty et al., 2001) tagger with a competitive baseline (Table 6). The baseline tagger was trained[5] on the full CoNLL-2003 shared task data. We experimented with the LOC, ORG and PER lists that were automatically generated in Section 4.1. In Table 7, we show the accuracy of the tagger for the entity types for which we had induced lists. The test conditions are just baseline features with no list membership, baseline plus seed list membership features, and baseline plus induced list membership features. For completeness, we also show in Table 8 accuracy on the full CoNLL task (four entity types) without lists, with seed list only, and with the three induced lists. The seed lists (Section 4.1) were prepared from training data itself and hence with increasing training data size, the model overfitted as it became completely reliant on these seed lists. From Tables 7 & 8 we see that incorporation of token membership in the extended lists as additional membership features led to improvements across categories and at all sizes of training data. This also shows that the extended lists are of good quality, since the tagger is able to extract useful evidence from them.

Relatively small sizes of training data pose interesting learning situation and is the case with practical applications. It is encouraging to observe that the list features lead to significant improvements in such cases. Also, as can be seen from Table 7 & 8, these lists are effective even with mature taggers trained on large amounts of labeled data.

[5]Standard orthographic information, such as character n-grams, capitalization, tokens in immediate context, chunk tags, and POS were used as features.

Training Data (Tokens)	Test-a			Test-b		
	No List	Seed List	Unsup. List	No List	Seed List	Unsup. List
9268	68.16	70.91	**72.82**	60.30	63.83	**65.56**
23385	78.36	79.21	**81.36**	71.44	72.16	**75.32**
46816	82.08	80.79	**83.84**	76.44	75.36	**79.64**
92921	85.34	83.03	**87.18**	81.32	78.56	**83.05**
203621	89.71	84.50	**91.01**	84.03	78.07	**85.70**

Table 7: CRF tagger F-measure on LOC, ORG, PER extraction.

Training Data (Tokens)	Test-a			Test-b		
	No List	Seed List	Unsup. List	No List	Seed List	Unsup. List
9229	68.27	70.93	**72.26**	61.03	64.52	**65.60**
204657	89.52	84.30	**90.48**	83.17	77.20	**84.52**

Table 8: CRF tagger F-measure on LOC, ORG, PER and MISC extraction.

5 Related Work

The method presented in this paper is similar in many respects to some of the previous work on context pattern induction (Riloff and Jones, 1999; Agichtein and Gravano, 2000; Lin et al., 2003; Etzioni et al., 2005), but there are important differences. Agichtein and Gravano (2000) focus on relation extraction while we are interested in entity extraction. Moreover, Agichtein and Gravano (2000) depend on an entity tagger to initially tag unlabeled data whereas we do not have such requirement. The pattern learning methods of Riloff and Jones (1999) and the generic extraction patterns of Etzioni et al. (2005) use language-specific information (for example, chunks). In contrast, the method presented here is language independent. For instance, the English pattern induction system presented here was applied on German data without any change. Also, in the current method, induced automata compactly represent all induced patterns. The patterns induced by Riloff and Jones (1999) extract NPs and that determines the number of tokens to include in a single extraction. We avoid using such language dependent chunk information as the patterns in our case include right[6] boundary tokens thus explicitly specifying the slot in which an entity can occur. Another interesting deviation here from previous work on context pattern induction is the fact that on top of extending seed lists at high precision, we have successfully included membership in these automatically generated lexicons as features in a high quality named entity tagger improving its performance.

6 Conclusion

We have presented a novel language-independent context pattern induction method. Starting with a few seed examples, the method induces in an unsupervised way context patterns and extends the seed list by extracting more instances of the same category at fairly high precision from unlabeled data. We were able to improve a CRF-based high quality named entity tagger by using membership in these automatically generated lists as additional features.

Pattern and entity ranking methods need further investigation. Thorough comparison with previously proposed methods also needs to be carried out. Also, it will be interesting to see whether the features generated in this paper complement some of the other methods (Miller et al., 2004; Liang, 2005; Ando and Zhang, 2005) that also generate features from unlabeled data.

7 Acknowledgements

We thank the three anonymous reviewers as well as Wojciech Skut, Vrishali Wagle, Louis Monier, and Peter Norvig for valuable suggestions. This work is supported in part by NSF grant EIA-0205448.

[6] In case of predictive left context.

Induced LOC Patterns
troops in -ENT-to
Cup qualifier against -ENT-in
southern -ENT-town
war - torn -ENT-.
countries including -ENT-.
Bangladesh and -ENT-,
England in -ENT-in
west of -ENT-and
plane crashed in -ENT-.
Cup qualifier against -ENT-,

Extracted LOC Entities
US
United States
Japan
South Africa
China
Pakistan
France
Mexico
Israel
Pacific

Induced PER Patterns
compatriot -ENT-.
compatriot -ENT-in
Rep. -ENT-,
Actor -ENT-is
Sir -ENT-,
Actor -ENT-,
Tiger Woods , -ENT-and
movie starring -ENT-.
compatriot -ENT-and
movie starring -ENT-and

Extracted PER Entities
Tiger Woods
Andre Agassi
Lleyton Hewitt
Ernie Els
Serena Williams
Andy Roddick
Retief Goosen
Vijay Singh
Jennifer Capriati
Roger Federer

Induced ORG Patterns
analyst at -ENT-.
companies such as -ENT-.
analyst with -ENT-in
series against the -ENT-tonight
Today 's Schaeffer 's Option Activity Watch features -ENT-(
Cardinals and -ENT-,
sweep of the -ENT-with
joint venture with -ENT-(
rivals -ENT-Inc.
Friday night 's game against -ENT-.

Extracted ORG Entities
Boston Red Sox
St. Louis Cardinals
Chicago Cubs
Florida Marlins
Montreal Expos
San Francisco Giants
Red Sox
Cleveland Indians
Chicago White Sox
Atlanta Braves

Table 9: Top ranking LOC, PER, ORG induced pattern and extracted entity examples.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM International Conference on Digital Libraries*.

Rie Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of ACL-2005*. Ann Arbor, USA.

Boris Chidlovskii. 2000. Wrapper generation by k-reversible grammar induction. *ECAI Workshop on Machine Learning for Information Extraction*.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web - an experimental study. *Artificial Intelligence Journal*.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of CoNLL-2003*.

Dayne Freitag. 1997. Using grammatical inference to improve precision in information extraction. In *ICML-97 Workshop on Automata Induction, Grammatical Inference, and Language Acquisition, Nashville*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML 2001*.

Percy Liang. 2005. Semi-supervised learning for natural language. *MEng. Thesis, MIT*.

Winston Lin, Roman Yangarber, and Ralph Grishman. 2003. Bootstrapped learning of semantic classes from positive and negative examples. In *Proceedings of ICML-2003 Workshop on The Continuum from Labeled to Unlabeled Data*.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of HLT-NAACL 2004*.

L. R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *Proc. of IEEE, 77, 257–286*.

Ellen Riloff and Rosie Jones. 1999. Learning Dictionaries for Information Extraction by Multi-level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*.

Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. 2005. Unsupervised learning of natural languages. In *Proceedings of National Academy of Sciences. 102:11629-11634*.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of EMNLP 2002*.

Tong Zhang and David Johnson. 2003. A robust risk minimization based named entity recognition system. In *Proceedings of CoNLL-2003*.