What Is This, Anyway: Automatic Hypernym Discovery

Alan Ritter and **Stephen Soderland** and **Oren Etzioni**
Turing Center
Department of Computer Science and Engineering
University of Washington
Box 352350
Seattle, WA 98195, USA
{aritter,soderlan,etzioni}@cs.washington.edu

Abstract

Can a system that "learns from reading" figure out on it's own the semantic classes of arbitrary noun phrases? This is essential for text understanding, given the limited coverage of proper nouns in lexical resources such as WordNet. Previous methods that use lexical patterns to discover hypernyms suffer from limited precision and recall.

We present methods based on lexical patterns that find hypernyms of arbitrary noun phrases with high precision. This more than doubles the recall of proper noun hypernyms provided by WordNet at a modest cost to precision. We also present a novel method using a Hidden Markov Model (HMM) to extend recall further.

Introduction and Motivation

A goal of "Machine Reading" is an automatic system that extracts information from text and supports a wide range of inferencing capabilities (Etzioni, Banko, and Cafarella 2006). To enable such inferencing, an Information Extraction (IE) system must go beyond representing extracted entities as text strings and *ontologize* the text strings (Pennacchiotti and Pantel 2006; Soderland and Mandhani 2007).

This involves semantically typing the text strings, grounding the string in real world entities where possible, and mapping the string to a concept in a formal taxonomy. In many cases, particularly for proper nouns, the text string is not found in an existing taxonomy and must be added on the fly as a child of an existing concept.

This paper focuses on just one step in this ontologizing process: finding hypernyms for an arbitrary noun phrase (NP). This is a necessary first step in semantically typing the NP and mapping it to a node in a taxonomy. The problem that we address here is as follows. Given an NP e, find a set of NPs c_i such that each c_i is a hypernym of e in some sense as judged by a human.

$$\textit{hypernymOf}(e) = \{c_1, c_2, ...c_k\}$$

Note that we define hypernymy as a relation between surface forms and not "synsets", as is done in WordNet.

Consider the example of an NP extracted from the Web, "Borland Delphi". This term is not found in WordNet, but lexical patterns in a large corpus suggest that Borland Delphi may have the following hypernyms:

hypernymOf(Borland Delphi)
= *development environment*
= *software development system*
= *software tool*
= *language*
= *technology*

The rest of this paper is organized as follows. We discuss previous attempts at solving this problem. The manually engineered WordNet thesaurus (Miller et al. 1990) has good coverage of common nouns, but contains relatively few entries for Proper Nouns. Several researchers have used methods based on lexical patterns to discover hypernyms (Hearst 1992; Roark and Charniak 1998; Caraballo 1999; Snow, Jurafsky, and Ng 2005), although each of these methods suffers from limited precision and recall. We then present a series of methods for finding hypernyms of arbitrary NPs: HYPERNYMFINDER$_{freq}$ which is based on the frequency of Hearst pattern matches and HYPERNYMFINDER$_{svm}$ that uses a Support Vector Machine (SVM) classifier to incorporate additional features. Each of these systems learns to find hypernyms from lexical patterns and corpus statistics. Since this is an inherently error-prone process, each version of HYPERNYMFINDER assigns a probability to each c_i it finds. Finally, we present HYPERNYMFINDER$_{hmm}$, which uses an HMM language model to extend the recall of HYPERNYMFINDER to cover NPs that do not match any of the lexical patterns used by the previous versions.

Previous Work on Hypernym Discovery

The WordNet thesaurus (Miller et al. 1990) is a high-precision lexical resource that has served as a baseline for hypernym discovery since the 1990's. On a test set of 953 Noun phrases randomly selected from our Web corpus, we found that only 17% of the proper nouns and 64% of the common nouns were covered by WordNet. Information is often spotty even for NPs that are found in WordNet. For example, WordNet has an entry for "Mel Gibson" as an instance of the class "actor", a subclass of "human", but does not indicate that he is a "man". We would need to use other

knowledge sources or extractions from a corpus to find that Mr. Gibson is also an instance of "movie star", "Australian", or "Catholic".

Some of the earliest research in automatically discovering hypernyms from text was by Hearst (1992). She presented the manually discovered lexical patterns listed in Figure 1 that detect hypernyms. These patterns are fairly good evidence that the entity E is a subclass or member of C. For the remainder of this paper, we will refer to these patterns as the *Hearst patterns*.

E is a C
E [,] and other C
E [,] or other C
C [,] such as E
C [,] including E
C [,] especially E
such C as E

Figure 1: The "Hearst patterns": Lexical patterns that predict that entity E belongs to class C.

The Hearst patterns suffer from limited recall – not all pairs $<E,C>$ will be found in such patterns. Also, due to the local nature of the patterns, they inevitably make errors. A sentence with

"... urban birds in cities such as pigeons ..."

matches the pattern "C such as E" with C bound to city and E bound to pigeon, leading to city as a hypernym of pigeon.

We made use of the Hearst patterns in our IE systems, KNOWITALL (Etzioni et al. 2004a; 2004b; 2005) and TEXTRUNNER (Banko et al. 2007). We found that we needed to refine the rules somewhat. We increased recall by replacing the NP E with a list of NPs for most patterns. For example, the pattern "C [,] such as NList" would identify k NPs with hypernym C when $NList$ is a list of k simple NPs.

We also found that some patterns only have high precision when the class name is identified as a plural noun by a statistical part of speech tagger.

Correct match on plural class name: "Developing countries such as China and India have emerged ..." ⇒ *hypernymOf*(China) = country ⇒ *hypernymOf*(India) = country
Error from singular class name: "I like to listen to country such as Garth Brooks." ⇒ *hypernymOf*(Garth Brooks) = country

The accuracy of proposed hypernyms is correlated with extraction frequency, the number of distinct sentences that match a Hearst pattern. We estimated that, on a corpus of 117 million Web pages, hypernyms with frequency 1 had accuracy of 30% for common nouns and 49% for proper nouns. With a frequency $\geq$ 40, the accuracy increased to 86% (at recall less than 0.03) for common nouns and accuracy of 87% (at recall 0.01) for proper nouns.

Snow, Jurafsky, and Ng (2005) induced a large number of weak patterns to detect hypernyms, using WordNet to automatically tag pairs of NPs from a corpus of 6 million newswire sentences. They then built a classifier using nearly 70,000 of these patterns and the frequency that each pattern occurs for a given pair of noun phrases in their corpus. Most of the patterns gave very weak evidence of a hypernym relation. The Hearst patterns, which were among those induced, were among the highest precision patterns.

Snow *et al.* found that their classifier outperformed a baseline of simply looking for a single occurrence of a Hearst pattern and also outperformed a WordNet classifier. They evaluated their classifier on a set of 5,300 hand-tagged pairs of NPs taken from random paragraphs in their newswire corpus. Their recall-precision graph indicates precision 0.85 at recall 0.10 and precision 0.25 at recall of 0.30 for their hypernym classifier. A variant of their classifier that included evidence from coordinate terms (terms with common ancestor classes) also had precision 0.85 at recall 0.10 but increased precision to 0.35 at recall 0.30.

More recently, (McNamee et al. 2008) applied the techniques described in (Snow, Jurafsky, and Ng 2005) to detecting hypernyms of Named Entities (i.e. Proper Nouns) in order to improve performance of Question Answering systems. Their corpus consisted of about 16 million sentences which was larger than that used by Snow et. al, but still an order of magnitude smaller than the corpus used in this work. They obtained 53% mean average precision on automatically detected hyponyms of 75 categories. Their automatically discovered hyponyms resulted in a 9% boost in performance on a TREC Question Answering data set.

Exploiting Lexico-syntactic Patterns

Our goal is an efficient, practical HYPERNYMFINDER that takes an arbitrary NP e and returns a ranked list of up to k terms that are hypernyms of e for any sense of e. We first present HYPERNYMFINDER$_{freq}$, a classifier based on the frequency of Hearst pattern matches.

Then we present HYPERNYMFINDER$_{svm}$ that combines a variety of features using an SVM classifier to distinguish the correct hypernyms from among matches to the Hearst patterns. Later we explore methods that extend recall beyond hypernyms found by the Hearst patterns.

Frequency-based Classifiers

As a first step in building each of the following versions of HYPERNYMFINDER, we ran TEXTRUNNER on corpus of 117 million Web pages and recorded candidate hypernym pairs that matched any of the Hearst patterns. For each of these pairs of noun phrases $<e, c>$ we counted the frequency of matches from distinct sentences for each Hearst pattern.

Although the precision of a candidate hypernym generally correlates with the frequency of pattern matches, we found that even very high frequency candidates are often errors. Simply setting a threshold on frequency is not sufficient to obtain high precision.

The reason for this is the local nature of the Hearst patterns, which means that they can be fooled by certain sentence structures that occur with high frequency. Here are some examples of high frequency matches on Hearst patterns that produce incorrect hypernyms.

A.	"... all around the world including Australia, ..."
	⇒ *hypernymOf*(Australia) = world
B.	"... information about this hotel such as rates, ..."
	⇒ *hypernymOf*(rates) = hotel
C.	"... first prize is a trip to ..."
	⇒ *hypernymOf*(prize) = trip

Figure 2: Certain sentence structures that cause errors match Hearst patterns with high frequency. Fortunately, most of these sentence structures match only "right" patterns (as in A and B) or only "left" patterns (as in C). High precision can be obtained by filtering out hypernyms that do not have at least one right and one left pattern.

Most of these high frequency errors occur only with "right" patterns (where e comes to the right of c) or only with "left" patterns (where e comes to the left of c). Precision is improved by requiring at least one match on both right and left pattern.

Figure 3 compares the recall-precision curve for two methods of using Hearst pattern frequencies to find hypernyms. Both methods rank the proposed hypernyms by the number of distinct sentences that match a Hearst pattern. The baseline frequency method simply ranks by total pattern matches. The filtered frequency method (labeled as "1 left and 1 right") assigns low probability to any proposed hypernyms that are not found in at least one right pattern and at least one left pattern.

For this and each of the other experiments, we used a test set of 953 well-formed simple NPs e that were randomly selected from TEXTRUNNER extractions from a corpus of 117 million Web pages. We had each method output up to $k = 5$ proposed hypernyms for each e. We hand-tagged the proposed hypernyms as correct, incorrect, or vague.

We compute recall in these experiments as the percentage of NPs in the test set that are "covered" by each method – those NPs for which at least one correct hypernym is found. Precision is the total number of correct hypernyms divided by the total number of guessed hypernyms (up to a maximum of 5 for each e). Note that this differs from the method used to compute Precision and recall used by (Snow, Jurafsky, and Ng 2005). Their method computed precision and recall over candidate hypernym/hyponym word pairs. In contrast we are more concerned with whether or not we can find a good hypernym for a given noun phrase.

The precision-recall curve for the baseline frequency method has recall 0.11 at precision 0.90 after an initial dip below 0.90 at recall 0.02. The filtered frequency method avoids this initial dip in precision and reaches recall 0.23 at precision 0.90, more than twice the recall of the baseline method. Each method reaches recall of 0.66 at precision 0.64.

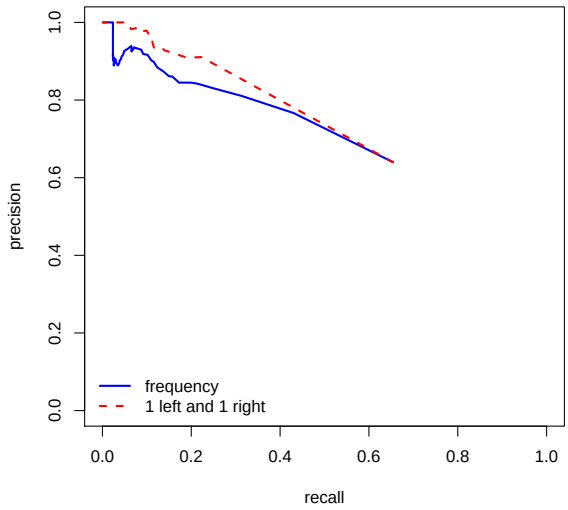

Figure 3: Ranking hypernyms by the frequency of Hearst pattern matches covers 66% of the NPs in our test set. Filtering these hypernyms to require at least one "left" and one "right" pattern gives a boost in precision.

Note that there is a limit in recall for any method built on the Hearst patterns – about 1/3 of the noun phrases in our test set do not occur in any Hearst patterns in our large web corpus.

Combining Evidence with SVM

Although simple rules based on extraction frequency and the existence of both right and left patterns proved surprisingly effective, there are other types of evidence that can help distinguish correct hypernyms. We created HYPERNYMFINDER$_{svm}$ that uses an SVM classifier to assign a probability of correctness (Platt 1999) to a proposed hypernym pair $<e, c>$ that has matched at least one Hearst pattern on a large text corpus.

We used the following features to classify the proposed hypernym $<e, c>$:

- Total number of distinct pattern matches.
- Total number of *left* pattern matches.
- Total number of *right* pattern matches.
- Total number of matches for "e is a c".
- Fraction of matches e which follow articles, determiners, or quantifiers.
- The rank by frequency among hypernyms of e.

While frequency is the best indicator of hypernym correctness, we wanted to combine this flexibly with other weaker evidence. Our rule-based classifier filtered out proposed hypernyms where either left or right pattern matches had zero frequency, but this is a bit extreme. Many hypernyms with unbalanced left-right patterns are actually correct.

We included the frequency of matches on "e is a c" because this is somewhat less reliable than the other patterns. The ratio of e which follow articles, determiners and quantifiers helps avoid matches on sentences such as "Our dog is a bloodhound ...". This does not imply that all dogs are bloodhounds.

We trained a Support Vector Machine (SVM) classifier on hand-labeled data using 10 fold cross validation. For these experiments, we split the test referenced earlier into a set of 583 proper nouns and a set of 370 common nouns, and evaluated each set separately [1].

We consider a pair of NPs $<e, c>$ in our test sets as correct if e belongs to the class c for some sense of e. Thus the entity "cedar" has hypernyms "wood", "softwood", and "material" for one sense of cedar, and also has hypernyms "tree" and "conifer" for another sense. We let each method output up to 5 hypernyms for each NP in the test set.

WordNet is a baseline for high-precision hypernyms – if a word is found in WordNet, it is guaranteed to have at least one hypernym. By our definition of correct hypernyms, WordNet has nearly perfect precision. So a practical hypernym finder would first look up a word in WordNet, before augmenting this with other methods.

In the remainder of the experiments, we begin by looking up each e in WordNet and consider the words found in WordNet to be "covered" with at least one correct hypernym. Thus, each of our recall-precision curves starts at recall 0.17 and precision 1.00 for proper nouns, and starts at recall 0.64 and precision 1.00 for common nouns. These represent the words found in WordNet. We found 43% of the proper nouns and 9% of the common nouns which were covered by WordNet to be multi-word phrases. Our interest is in extending the recall to words not in WordNet.

As in earlier experiments, we count recall as the number of e for which at least one correct hypernym c is found. Precision is computed as the fraction of c which are correct.

Figures 4 and 5 present results for proper nouns and common nouns, respectively. The lines labeled "Rule-based Classifier" are for the filtered frequency implementation of HYPERNYMFINDER$_{freq}$The lines labeled "SVM" are from HYPERNYMFINDER$_{svm}$ using the features enumerated in this section.

A simple WordNet lookup covers 17% of proper nouns. HYPERNYMFINDER$_{freq}$ extends the recall to 0.32 at precision 0.90 and recall 0.65 at precision 0.64. The curve for HYPERNYMFINDER$_{svm}$ is consistently higher than for HYPERNYMFINDER$_{freq}$ and has recall 0.37 at precision 0.90 and a higher precision of 0.73 at the tail of the curve.

There was less room for improvement over WordNet for common nouns, of which WordNet covers 64%. Here HYPERNYMFINDER$_{freq}$ extends recall to 0.67 at precision 0.90 and gives recall of 0.80 at precision 0.61. Again, the curve for HYPERNYMFINDER$_{svm}$ is consistently higher than for HYPERNYMFINDER$_{freq}$ with recall 0.70 at precision 0.90.

These methods are straightforward ways to find hypernyms for arbitrary proper nouns that extend recall beyond that of WordNet, particularly the SVM classifier, with a graceful decline in precision at higher recall. Yet these results are limited by the coverage of Hearst patterns. For 20% of the common nouns and 36% of the proper nouns in our test set, there were no matches of Hearst patterns that give a correct hypernym. In the following section, we describe experiments to extend the coverage of our HYPERNYMFINDER to find hypernyms for some of these NPs that are beyond the reach of Hearst patterns.

[1] 56% of the proper nouns and 27% of the common nouns were multi-word phrases

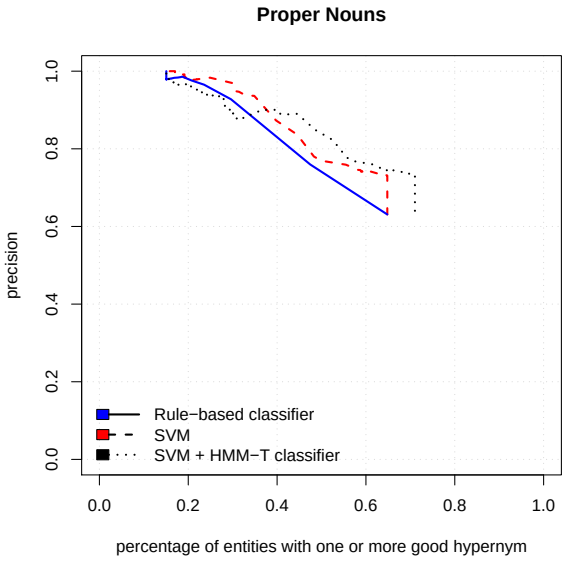

Figure 4: Each of our methods extends recall for proper nouns far beyond the recall of 0.17 from WordNet lookup. An SVM classifier uniformly raises precision over simple rules using Hearst pattern frequency. Adding an HMM language model extends recall even further.

Extending Recall Beyond Patterns

Here is the intuition of our approach to extending recall beyond the Hearst patterns. If we cannot find hypernyms for entity e using the methods presented thus far, perhaps we can find a sufficiently similar entity e' whose hypernyms we know. Given a good similarity metric, there should be a high probability that e and e' share the same hypernyms.

HMM Language Models

Inspired by REALM (Downey, Schoenmackers, and Etzioni 2007) we investigated using Hidden Markov Models (HMMs) (Rabiner 1989) as the similarity metric for another version of HYPERNYMFINDER. REALM uses HHMs to perform type checking of arguments in an open-domain IE system.

Given an HMM with n hidden states, which has been trained to model the generation of our corpus, we can compute the probability that each NP is generated by each of the

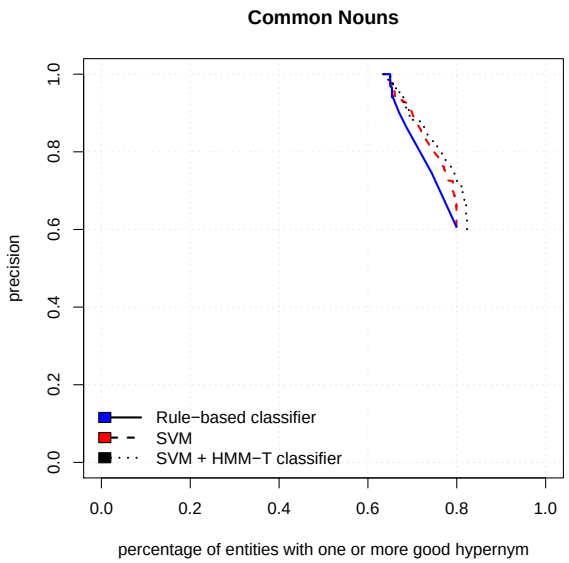

Figure 5: Performance on common nouns has qualitatively similar results to that of proper nouns, but with a higher initial recall from WordNet and a smaller gain in recall from our other methods.

HMM's hidden states. This produces for each NP in our corpus, a vector of n probabilities that can be used as features to a classifier or distance measure.

Downey *et al.* performed experiments that suggest that HMMs produce better similarity measures than traditional term-vector-based approaches. The intuitive argument for this is that many coordinate terms (terms that share the same hypernym) appear in similar, though not identical contexts. For example *Pickerington* and *Chicago* are coordinate terms, because they share the common hypernym, *City*.

Now suppose that we see the following two contexts:

"Pickerington, Ohio"

"Chicago, Illinois"

These two contexts are not identical, so in a term-vector-based similarity measure, they provide no evidence that Pickerington and Chicago are coordinate terms. However the two contexts, "· , Ohio" and "· , Illinois" are quite similar and so it is likely that they were generated by the same hidden states in the HMM. So by viewing the hidden state distribution of an HMM as context vectors, these two cities have a high similarity given the above evidence.

Using HMMs for Hypernym Discovery

We wish to create a HYPERNYMFINDER$_{hmm}$ that retains the performance of HYPERNYMFINDER$_{svm}$ on NPs that are covered by Hearst patterns, but which additionally finds hypernyms for NPs that do not match any of the Hearst patterns in our corpus. This HYPERNYMFINDER$_{hmm}$ assigns to each NP a probability that is a linear combination of the probability from HYPERNYMFINDER$_{svm}$ and the probability from a new classifier HYPERNYMFINDER$_{coord}$.

HYPERNYMFINDER$_{coord}$ uses HMM hidden state distribution as features to find potential hypernyms c for an arbitrary NP e as follows. Rather than compute the probability that c is a hypernym of e, this classifier computes the probability that e has a coordinate term e' that is a *hyponym* of c.

The HYPERNYMFINDER$_{coord}$ increased the pool of proposed hypernym pairs $<e, c>$, but had relatively low precision. Just because two NPs have highly similar context does not necessarily mean that they share the same hypernyms. We had the highest precision for the more coarse-grained classes such as *person*, *organization*, and *region*. For that reason, we chose a relatively high level in the WordNet hierarchy as a cutoff (6 levels from the top).

We also set a threshold on the HMM similarity metric, so HYPERNYMFINDER$_{coord}$ only outputs $<e, c>$ if it finds a highly similar e' such that e' is a hyponym of a high-level WordNet class c. We experimented with various weights to combined the probabilities and found the best results on our test set to be where HYPERNYMFINDER$_{svm}$ had weight 0.8 and HYPERNYMFINDER$_{coord}$ had weight 0.2.

The result of these settings is shown in Figures 4 and 5 as "SVM + HMM-T classifier". For both common nouns and proper nouns, adding evidence from HYPERNYMFINDER$_{coord}$ extends the curve to the right. This increases recall for common nouns from 0.80 to 0.82 and increases recall for proper nouns from 0.65 to 0.71.

Unfortunately, in both cases the curve for HYPERNYMFINDER$_{hmm}$ has an initial dip. This is because a term e' with the most similar context to e is often not actually a coordinate terms, so assigning the hypernyms of e' to e results in a error.

Conclusions

We have presented a series of methods to find hypernyms of an arbitrary noun phrase e. These provide easy-to-implement methods, the simplest of which requires no tagged training. The main lessons we learned is that fairly simple methods are sufficient to find hypernyms at high precision and with fairly high recall. This is particularly true for proper nouns where our HYPERNYMFINDER more than doubles the recall for proper nouns at precision 0.90 compared to looking up terms in WordNet. The gain in recall is less for common nouns, where WordNet has coverage of 64%.

It is still an open question how best to extend recall beyond hypernyms that match one of the Hearst patterns, although we had some success in using an HMM language model. Some of our future research in machine reading depends on finding hypernyms with high coverage and high precision. We are inclined to try a hybrid approach – combine the methods presented here with using a named-entity tagger to assign a coarse-grained class to proper nouns, using a large suite of weak hypernym patterns, and using lexical patterns as well as a language model to find coordinate terms.

Acknowledgments

This research was supported in part by NSF grants IIS-0535284 and IIS-0312988, ONR grant N00014-08-1-0431 as well as gifts from the Utilika Foundation and Google, and was carried out at the University of Washington's Turing Center.

References

Banko, M.; Cafarella, M.; Soderland, S.; Broadhead, M.; and Etzioni, O. 2007. Open information extraction from the Web. In *Procs. of IJCAI*.

Caraballo, S. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Procs. of the 37th Annual Meeting of the Association for Computational Linguistics*, 120–126.

Downey, D.; Schoenmackers, S.; and Etzioni, O. 2007. Sparse information extraction: Unsupervised language models to the rescue. In *Proc. of ACL*.

Etzioni, O.; Banko, M.; and Cafarella, M. 2006. Machine Reading. In *AAAI*.

Etzioni, O.; Cafarella, M.; Downey, D.; Kok, S.; Popescu, A.; Shaked, T.; Soderland, S.; Weld, D.; and Yates, A. 2004a. Web-Scale Information Extraction in KnowItAll. In *WWW*, 100–110.

Etzioni, O.; Cafarella, M.; Downey, D.; Popescu, A.; Shaked, T.; Soderland, S.; Weld, D.; and Yates, A. 2004b. Methods for domain-independent information extraction from the Web: An experimental comparison. In *Procs. of the 19th National Conference on Artificial Intelligence (AAAI-04)*, 391–398.

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

Hearst, M. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Procs. of the 14th International Conference on Computational Linguistics*, 539–545.

McNamee, P.; Snow, R.; Schone, P.; and Mayfield, J. 2008. Learning named entity hyponyms for question answering. In *To appear in Proceedings of IJCNLP 2008*.

Miller, G.; Beckwith, R.; Fellbaum, C.; Gross, D.; and Miller, K. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography* 3(4):235–312.

Pennacchiotti, M., and Pantel, P. 2006. Ontologizing semantic relations. In *COLING/ACL-06*, 793–800.

Platt, J. C. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, 61–74. MIT Press.

Rabiner, L. R. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2):257–286.

Roark, B., and Charniak, E. 1998. Noun-phrase co-occurence statistics for semi-automatic semantic lexicon construction. In *Procs. of COLING-ACL 1998*, 1110–1116.

Snow, R.; Jurafsky, D.; and Ng, A. Y. 2005. Learning syntactic patterns for automatic hypernym discovery. In *NIPS 2005*.

Soderland, S., and Mandhani, B. 2007. Moving from textual relations to ontologized relations. In *AAAI Spring Symposium on Machine Reading*.