In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*

Unsupervised Learning of Generalized Names

Roman Yangarber, Winston Lin, Ralph Grishman
Courant Institute of Mathematical Sciences
New York University
{roman|winston|grishman}@cs.nyu.edu

Abstract

We present an algorithm, NOMEN, for learning *generalized names* in text. Examples of these are names of diseases and infectious agents, such as bacteria and viruses. These names exhibit certain properties that make their identification more complex than that of regular proper names. NOMEN uses a novel form of bootstrapping to grow sets of textual instances and of their contextual patterns. The algorithm makes use of competing evidence to boost the learning of several categories of names simultaneously. We present results of the algorithm on a large corpus. We also investigate the relative merits of several evaluation strategies.

1 Introduction

This research grew out of the Integrated Feasibility Experiment on Biological Infectious Outbreaks (IFE-BIO), a project to build an Information Extraction (IE) system for identifying events related to outbreaks and epidemics of infectious disease, (Grishman et al., 2002).

IE generally relies on knowledge bases of several kinds, and the most fundamental of these is the domain-specific lexicon—lexical items that are not likely to be found in general-purpose dictionaries. This particular scenario requires a comprehensive list of disease names. Other requisite classes of names include: biological *agents* causing disease, such as viruses and bacteria; *vectors*—organisms or animals capable of transmitting infection; and possibly names of drugs, used in treatment.

1.1 Generalized Names

Names of these kinds, *generalized names* (GNs), differ from conventional proper names (PNs) that have been studied extensively in the literature, e.g., as part of the traditional Named Entity (NE) categorization task, which evolved out of the MUC NE evaluation, (Wakao et al., 1996; Bikel et al., 1997; Borthwick et al., 1998; Collins and Singer, 1999). The three mainstream NE kinds are *location*, *person*, and *organization*, and much research has centered on these "classical" kinds of proper names.

On the other hand, the vast field of terminology has traditionally dealt with identifying single- and multi-word domain-specific expressions, for various NLP tasks, and recent years have seen a growing convergence between the two fields.

In fact, good identification of names of both kinds is essential for IE in general. In IFE-BIO, for example, the text:

> National Veterinary Services Director Dr. Gideon Bruckner said no cases of mad cow disease have been found in South Africa.

exhibits more than one problem of name identification and classification. We focus on generalized names, which pose numerous challenges.

The classification process usually starts with identification, but the primary cue for a proper name—capitalization (in English text)—is unavailable for generalized names. GNs are not always capitalized ("mad cow disease" or "tuberculosis") or may be partially capitalized ("Ebola haemorrhagic fever", "E. coli"). GNs often have multiple pre- and post-modifiers—"*(new) variant* Creutzfeldt-Jacob disease," or may modify the head of a noun group—"Bacillus anthracis *infection*." Locating the boundaries of GNs is much harder than for PNs.

The problem of ambiguity affects generalized names, as it does proper names. *E. coli* can refer to the organism or to the disease it causes; *encephalitis* can mean a disease or a symptom.

1.2 Why Learning?

Why is it undesirable to rely on fixed, specialized, domain-specific lists or gazetteers?

1. Comprehensive lists are not easy to obtain.

2. Lists are never complete, since new names (locations, diseases) periodically enter into existence and literature.

3. A typical text contains all the information that is necessary for a human to infer the category. This makes discovering names in text an interesting research problem in its own right.

The following section introduces the learning algorithm; Section 3 compares our approach to related prior work; Section 4 presents an evaluation of results; we conclude with a discussion of evaluation and current work, in Section 5.

2 NOMEN: The Learning Algorithm

NOMEN is based on a bootstrapping approach, similar in essence to that employed in (Yangarber et al., 2000).[1] The algorithm is trained on a large corpus of medical text, as described in Section 4.

2.1 Pre-processing

A large text corpus is passed through a zoner, a tokenizer/lemmatizer, and a part-of-speech (POS) tagger. The zoner is a rule-based program to extract *textual* content from the mailing-list messages, i.e., stripping headers and footers. The tokenizer produces lemmas for the inflected surface forms. The statistical POS tagger is trained on the Wall Street Journal (possibly sub-optimal for texts about infectious disease). Unknown or foreign words are not lemmatized and marked *noun* by the tagger.

2.2 Unsupervised Learning

0. Seeds: The user provides several *trusted* seeds of each category we intend to learn. E.g., we selected the 10 most common diseases as seeds for the disease category; the same for locations and several other categories.[2]

For each category, the set of accepted names, $AcceptName$, is initialized with the seeds.

1. Tagging: For each accepted name in each category C to be learned, *Nomen* tags the lemmatized, POS-tagged training corpus, placing left and right tags around each occurrence of the name—e.g., `<disease>` and `</disease>`.

2. Pattern Generation: For each tag T inserted in the corpus on Step 1, NOMEN generates 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.

Note, the tag of the pattern, $Tag(p) = T$, indicates both a *direction*, either "left" or "right," $Dir(p) \in \{left, right\}$, and a category, $Cat(p)$. E.g., if $Tag(p) =$ `</disease>`, then $Dir(p) = right$ and $Cat(p) = disease$.

Then p is transformed replacing each element in the w-window by its generalization; in the current simple scheme, the only generalization can be a wildcard. These patterns form the set of *potential* patterns, Φ. Note that each pattern matches on only one side of an instance, either its beginning or its end.

3. Pattern Matching: Match every pattern $p \in \Phi$ against the entire training corpus. In a place where the context of p matches, p predicts where one boundary of a name in text would occur. Let pos_a be the position of this boundary. Then use a noun group (NG) regular expression[3] to search for the other, *partner* boundary, say, at position pos_b. For example, suppose p matches in the text

$$the\ \overbrace{\underbrace{\langle_1 yellow\ \ fever\rangle_2}_{\longrightarrow}\ vaccine\rangle_3}^{\longleftarrow}\ to\ \ villagers$$

at $pos_a = 2$ and $Dir(p) = right$; then $pos_b = 1$. However, if $pos_a = 1$ and $Dir(p) = left$ then $pos_b = 3$. (Note, the search proceeds in the opposite direction of $Dir(p)$.) Next, we check whether the NG between positions pos_a and pos_b has already been accepted as a name in some category; the result can be:

[1] For a detailed comparison of the algorithms, cf. (Yangarber, 2002).

[2] Frequency counts are computed from a large IE database, of more than 10,000 records. The most common disease names: cholera, dengue, anthrax, BSE, rabies, JE, Japanese encephalitis, influenza, Nipah virus, FMD (for foot-and-mouth disease).

[3] Using heuristics, as in terminology discovery, (Frantzi et al., 2000); we use a simple NG regular expression, `[Adj* Noun+]`.

- *positive*: The NG has already been accepted as a name in the same category as $Cat(p)$;
- *negative*: The NG has already been accepted as a name in a different category, $C' \neq Cat(p)$;
- *unknown*: The NG has not yet been accepted as a name in any category.

The *unknown* case is where a new candidate of the category $Cat(p)$ may potentially be discovered.

4. Pattern Acquisition: For each pattern $p \in \Phi$, this gives us instance-based lists of positive $pos(p)$, negative $neg(p)$ and unknown $unk(p)$ NGs. To compute $Score(p)$, we first define the corresponding *type-based* sets:

- $pos^*(p) =$ set of *distinct* names of category $Cat(p)$ from $AcceptName$ that p matched.
- $neg^*(p) =$ set of distinct names of a wrong category.
- $unk^*(p) =$ set of distinct NGs of unknown type.

To score the patterns in Φ, we currently use the *accuracy* and *confidence* measures:

$$acc^*(p) = \frac{|pos^*|}{|pos^*| + |neg^*|}$$

$$conf^*(p) = \frac{|pos^*|}{|pos^*| + |neg^*| + |unk^*|}$$

Patterns with accuracy below a precision threshold $acc^*(p) < \theta_{prec}$, are removed from Φ. The remaining patterns are ranked as follows. The score is computed as:

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

Add the n–best patterns for each target category to the set of *accepted* patterns, $AcceptPat$.

In the first term of the scoring function, higher confidence implies that we take less risk if we acquire the pattern, since acquiring the pattern affects the unknown population. The second term favors patterns which select a greater number of *distinct* names in $AcceptName$.

5. Application: Apply each pattern $p \in AcceptPat$ to the entire corpus.

The noun groups in the set $unk^*(p)$ are the candidates for being added to the category $Cat(p)$. Let Ψ be the list of candidate types:

$$\Psi = \bigcup_{p \in AcceptPat} unk^*(p)$$

6. Candidate Acquisition: Compute a score for each candidate type $t \in \Psi$, based on

- how many *different* patterns in $AcceptPat$ match an instance of type t,
- how reliable these patterns are.

To rank a candidate type $t \in \Psi$ consider the set of patterns in $AcceptPat$ which match on some instance of t; let's call this set M_t. If $|M_t| < 2$, the candidate is discarded.[4] Otherwise, 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 combines evidence by favoring candidates matched by a greater number of patterns; on the other hand, the term $conf^*(p)$ assigns more credit to the more reliable patterns.

For each target category, add the m best-scoring candidate types to the set $AcceptName$.

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

3 Prior Work

The NOMEN algorithm builds on some ideas in previous research. Initially, NE classification centered on supervised methods, statistically learning from tagged corpora, using Bayesian learning, ME, etc., (Wakao et al., 1996; Bikel et al., 1997; Borthwick et al., 1998). (Cucerzan and Yarowsky., 1999) present an unsupervised algorithms for learning proper names. AutoSlog-TS, (Riloff and Jones, 1999), learns "concepts" (general NPs) for filling slots in events, which in principle can include generalized names. The algorithm does not use competing evidence. It uses syntactic heuristics which mark whole noun phrases as candidate instances, whereas NOMEN also attempts to learn names that appear as modifiers within a NP.

[4] Note, this means that the algorithm is unlikely to learn a candidate which occurs only once in the corpus. It can happen if the unique occurrence is flanked by accepted patterns on both sides.

In the area of NE learning, $(LP)^2$, (Ciravegna, 2001), is a recent high-performance, supervised algorithm that learns contextual *surface-based* rules separately for the left and the right side of an instance in text. Separating the two sides allows the learner to accept weaker rules, and several correction phases compensate in cases of insufficient evidence by removing uncertain items, and preventing them from polluting the set of good seeds.

Research in automatic terminology acquisition initially focused more on the problem of identification and statistical methods for this task, e.g., (Justeson and Katz, 1995), the C-Value/NC-Value method, (Frantzi et al., 2000). Separately, the problem of classification or clustering is addressed in, e.g., (Ushioda, 1996)

(Strzalkowski and Wang, 1996) presents an algorithm for learning "universal concepts," which in principle includes both PNs and generic NPs—a step toward our notion of generalized names. The "spotter" proceeds iteratively from a handful of seeds and learns names in a single category.

DL-CoTrain, (Collins and Singer, 1999), learns capitalized proper name NEs from a syntactically analyzed corpus. This allows the rules to use deeper, longer-range dependencies, which are difficult to express with surface-level information alone. However, a potential problem with using this approach for our task is that the Penn-Treebank-based parser does not assign structure to noun groups, so it is unclear that it could discover generalized names, as these often occur within a noun group, e.g., "the 4 yellow fever cases." Our approach does not have this limitation.

The salient features of NOMEN: it learns

- generalized names, with no reliance on capitalization cues, as would be possible in the case of proper names (in English).
- from an un-annotated corpus, bootstrapping from a few manually-selected seeds
- rules for left and right contexts independently (as $(LP)^2$ to boost coverage).
- several categories simultaneously, and uses additional categories for negative evidence to reduce overgeneration.

4 Results

The algorithm was developed using a corpus drawn from the ProMed mailing list. ProMed is a global forum where medical professionals post information regarding outbreaks of infectious disease (using at times informal language).

Our full training corpus contains 100,000 sentences from 5,100 ProMed articles, from the beginning of 1999 to mid-2001. A subset of that, used for *development*, contains 26,000 sentences from 1,400 documents (3.2Mb) from January to July 1999.

Our evaluation strategy differs from those in some of the prior work. We discuss the competing evaluation strategies in detail in Section 5.2.

To measure performance, we constructed several *reference lists* as follows. First, a *manual list* of disease names was hand-compiled from multiple sources.[5] The manual list consists of 2,492 disease names.

The *recall list* is automatically derived from the manual list by searching the training corpus for disease names that surface more than once.[6] The recall list for the 26,000-sentence corpus contains 322 disease names, including some aliases and common acronyms.

The *precision list* is constructed as the union of the manual list with an automatically generated list of acronyms (made by collecting first letters of all multi-token names in the manual list). We applied the same procedure to generate recall and precision lists for locations.

Then, we judge the recall of NOMEN against the recall lists, and precision against the precision lists. The list sizes are shown in Table 1.

We focus on two categories, diseases and locations, while learning several categories simul-

[5] Using a disease IE database (Grishman et al., 2002), the Gideon disease database, and Web search. The list includes some common acronyms, like HIV and FMD.

[6] This is justified because the current algorithm is unlikely to discover a name that occurs only once.

Reference List	*Disease*	*Location*
Manual	2492	1785
Recall (26K corpus)	322	641
Recall (100K corpus)	616	1134
Precision	3588	2404

Table 1: Reference Lists

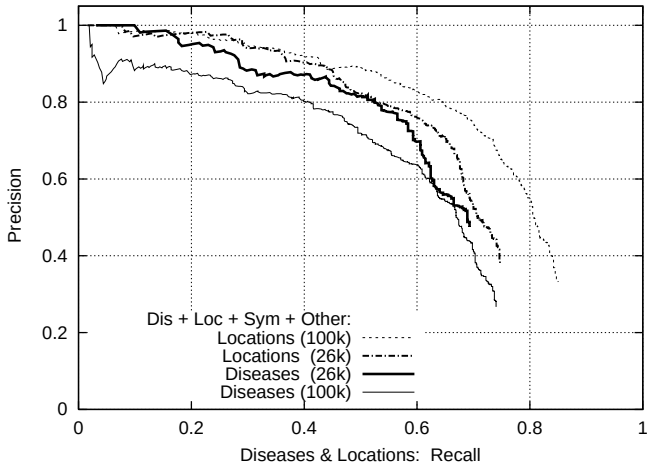

Figure 1: Names: Recall vs. Precision

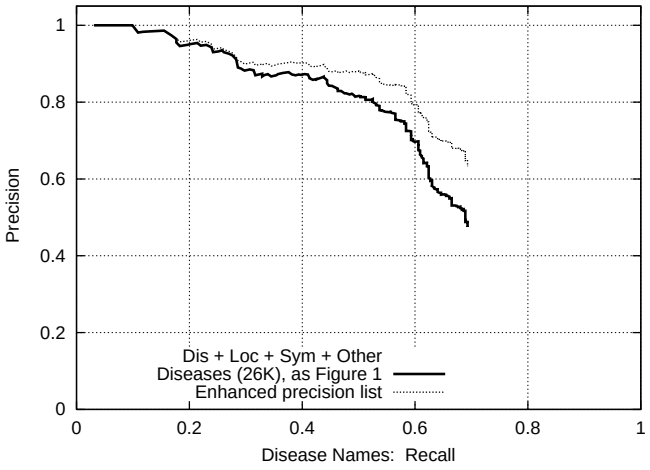

Figure 2: Effect of Understated Precision

taneously.[7] We introduce a category for symptoms, discussed in the next section.

We also introduce a *negative* category for learning terms belonging to none of the classes. As seeds, we use the 10 most frequent NGs in the corpus, excluding disease and location names, and generic words for diseases or locations ("virus," "outbreak," "area").[8]

The parameters in these experiments are: number of seeds = 10 per category; pattern accuracy threshold $\theta_{prec} = 0.80$; $n = m = 5$ for the number of retained patterns and candidates.

The learning curves in Figure 1 show how recall and precision for diseases and locations vary across the iterations. The bold curves show the result for diseases and locations on the development corpus (26K); e.g., by the end, 70% of diseases (from the recall list of 322 items) were learned, at 50% precision—half of the learned names were not on the precision list. On the 100K corpus (with 641 diseases on the recall list) the precision was only slightly lower.

The precision measures, however, are understated. Because it is not possible to get a full list for measuring precision, we find that NOMEN is penalized for finding correct answers. This is a general problem of type-based evaluation.

To quantify this effect, we manually examined the disease names learned by NOMEN on the development corpus and re-introduced those that were *incorrectly* marked as errors, into the precision list only. The updated graph is shown in Figure 2; at 70% recall the true precision is 65%. Note that precision is similarly understated for all type-based curves in this paper.

Among the re-introduced names there were 99 new diseases which were missed in the manual compilation of reference lists.[9] This is an encouraging result, since this is ultimately how NOMEN is intended to be used: for discovering new, previously unknown names.

5 Discussion

5.1 Competing Categories

Figure 3 demonstrates the usefulness of competition among target categories. All curves show the performance of NOMEN on the *disease* category, when the algorithm is seeded only with diseases (the curve labeled *Dis*), when seeded with diseases and locations (*Dis+Loc*), and with symptoms, and the "other" category. The curves *Dis* and *Dis+Loc* are very similar. However, when more categories are added, precision and recall increase dramatically.

When only one category is being learned, $acc(p) = 1.0$ for all patterns p. The lack of an effective accuracy measure causes us to acquire unselective disease name patterns that often also match non-diseases (e.g., "... X has been confirmed"). This hurts precision.

[7] Locations seeds: United States, Malaysia, Australia, Belgium, China, Europe, Taiwan, Hong Kong, Singapore, France.

[8] The negative seeds were: case, health, day, people, year, patient, death, number, report, farm.

[9] Examples of new diseases: rinderpest, konzo, Mediterranean spotted fever, coconut cadang-cadang, swamp fever, lathyrism, PRRS (for "porcine reproductive and respiratory syndrome"); locations: Kinta, Ulu Piah, Melilla, Anstohihy, etc.

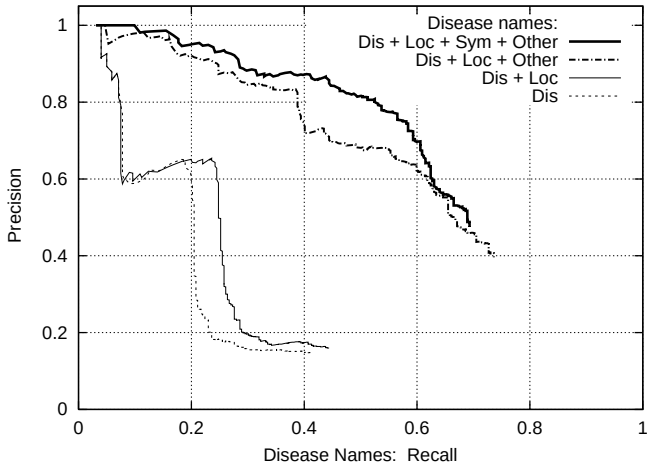

Figure 3: Diseases: Effect of Competition

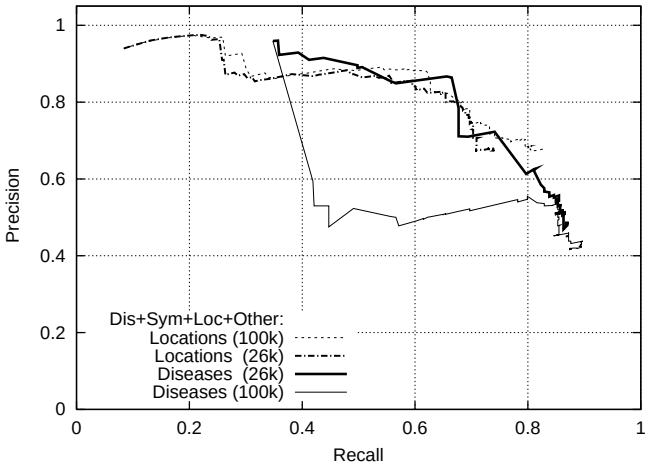

Figure 4: Token-based, MUC-style Evaluation

Recall also suffers, (a) because some patterns that are more selective (but have lower confidence or coverage) are neglected, and (b) because non-diseases contaminate the seed set and generate useless patterns.

(Collins and Singer, 1999) also makes use of competing categories (person, organization, and location), which cover 96% of all the instances it set out to classify. In our case, the sought categories, (diseases and locations), do not cover the bulk of potential candidates for generalized names—word sequences matching `[ADJ* N+]`. Introducing the "negative" category helps us cover more of the potential candidates. This in turn boosts the utility of the accuracy measure.

Additional competing categories may help to prevent a category from "creeping" into an overlapping concept. E.g., we had mentioned that the disease and symptom classes may overlap. When the target categories include diseases but not symptoms, NOMEN learns some names that can function as either. This leads to learning of some patterns which tend to occur with symptoms only, resulting in precision errors. Figure 3 shows the improvement in precision from adding the symptom category.

On the other hand, there may be disadvantages to splitting categories too finely. For example, one problem is metonymy among classes of generalized names. It appears to be distinct from the problem of ambiguity in PNs, e.g., when "Washington" may refer to a person, or a location. In the case of PNs, there are usually clues in the context to afford disambiguation. In the case of GNs, rather, the nature of ambiguity may be related to *regular* metonymy. For example, names of agents regularly function as the name of the disease they cause: "E. coli." Therefore, in learning agents and diseases separately, the algorithm will naturally confound the two classes, which will inhibit learning. In these experiments, we learn them as a single class.

It may then be more appropriate to apply another procedure to separate the classes based on a measure of *prevalence* of co-occurrence with the respectively characteristic contexts.

5.2 Evaluation

The results in the preceding figures are not directly commensurate with those in the mentioned literature, e.g., (Strzalkowski and Wang, 1996; Collins and Singer, 1999). This relates to the *token-type* dichotomy.

The evaluation in the prior work is token-based, where the learner gets credit—recall points—for identifying an instance correctly, for every time it occurs in the corpus. In our *type-based* evaluation, it gets credit only once per name, no matter how many times it occurs.

We also conducted an instance-based evaluation, more compatible with the mentioned prior work. We manually tagged *all* diseases and locations in a 500-sentence test sub-corpus. Using the output from the runs in Figure 1 we measured recall and precision using the standard MUC NE scoring scheme, shown in Figure 4.[10]

[10]The sharp dip in the "diseases (100K)" curve is due to several generic terms that were learned early on; generics were *not* tagged in the test corpus.

Iteration	*Type-Based*	*Instance-Based*
0	0.03	0.35
20	0.18	0.68
40	0.31	0.85
60	0.42	0.85
300	0.69	0.86

Table 2: Evaluation of Disease Recall

Table 2 contrasts type-based and instance-based recall across the iterations. The instance-based evaluation can hardly distinguish between an algorithm that learns 31% of the types vs. one that learns 69% of the types. The algorithm keeps learning lots of new, infrequent *types* until iteration 340, but the instance-based evaluation does not demonstrate this.

5.3 Current Work

NOMEN can be improved in several respects.

The current regular-expression NG pattern is very simplistic. In its present form, it does not allow "foot and mouth disease" to be learned, nor "legionnaires' disease"; this introduces inaccuracy, since parts of these names are learned and contaminate the pool.

The current pattern generalization scheme could be expanded. $(LP)^2$ generalizes on surface form, case, and semantic information. We could use, e.g., parts of speech from the tagger, as a level of generalization between lemmas and wildcards. A complementary approach would be to use a NP chunker, to capture longer-distance relations, in the heads and prepositions of adjacent phrases. ((Collins and Singer, 1999) achieves this effect by full parsing.)

We are exploring acquisition of more types of generalized names—agents and vectors, as well as people and organizations. What is the effect of learning possibly related classes simultaneously, what happens to the items in their intersection, and to what extent they inhibit learning, remains a practical question.

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.

References

D. Bikel, S. Miller, R. Schwartz, and R. Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Proc. 5th Applied Natural Language Processing Conf.*, Washington, DC.

A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. 1998. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Proc. 6th Workshop on Very Large Corpora*, Montreal, Canada.

F. Ciravegna. 2001. Adaptive information extraction from text by rule induction and generalisation. In *Proc. 17th Intl. Joint Conf. on AI (IJCAI 2001)*, Seattle, WA.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proc. Joint SIGDAT Conf. on EMNLP/VLC*.

S. Cucerzan and D. Yarowsky. 1999. Language independent named entity recognition combining morphological and contextual evidence. In *Proc. Joint SIGDAT Conf. on EMNLP/VLC*.

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

R. Grishman, S. Huttunen, and R. Yangarber. 2002. Event extraction for infectious disease outbreaks. In *Proc. 2nd Human Lang. Technology Conf. (HLT 2002)*, San Diego, CA.

J.S. Justeson and S.M. Katz. 1995. Technical terminology: Some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(1):9–27.

E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proc. 16th Natl. Conf. on AI (AAAI-99)*, Orlando, FL.

T. Strzalkowski and J. Wang. 1996. A self-learning universal concept spotter. In *Proc. 16th Intl. Conf. Computational Linguistics (COLING-96)*.

A. Ushioda. 1996. Hierarchical clustering of words. In *Proc. 16th Intl. Conf. Computational Linguistics (COLING-96)*, Copenhagen, Denmark.

T. Wakao, R. Gaizauskas, and Y. Wilks. 1996. Evaluation of an algorithm for the recognition and classification of proper names. In *Proc. 16th Int'l Conf. on Computational Linguistics (COLING 96)*, Copenhagen, Denmark.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000. Automatic acquisition of domain knowledge for information extraction. In *Proc. 18th Intl. Conf. Computational Linguistics (COLING 2000)*, Saarbrücken, Germany.

R. Yangarber. 2002. Acquisition of domain knowledge. In M.T. Pazienza, editor, *Information Extraction*. Springer-Verlag, LNAI, Rome.