# Confidence Estimation Methods
# for Partially Supervised Relation Extraction

Eugene Agichtein*

**Abstract**

Text documents convey valuable information about entities and relations between entities that can be exploited in structured form for data mining, retrieval, and integration. A promising direction is a family of *partially-supervised* relation extraction systems that require little manual training. However, the output of such systems tend to be noisy, and hence it is crucial to be able to *estimate* the quality of the extracted information. We present Expectation-Maximization algorithms for automatically evaluating the quality of the extraction patterns and derived relation tuples. We demonstrate the effectiveness of our method on a variety of relations.

## 1 Overview

Text documents convey valuable *structured* information. For example, medical literature contains information about new treatments for diseases. More specifically, information extraction systems can identify particular types of entities (such as company, location, and person names) and relationships between entities (mergers and acquisitions of companies, locations of company headquarters, and names of company executives) in natural language text for storage and retrieval in a structured database [7]. Once created, the database can be used to answer specific questions quickly and precisely by retrieving answers instead of complete documents, for sophisticated query processing, for integration with relational databases, and for traditional data mining tasks.

A fundamental problem in information extraction is how to train an extraction system for an extraction task of interest. Traditionally, this training required substantial human effort and hence the development of information extraction systems was generally expensive and time consuming. An attractive approach to reduce the training cost, pioneered by Brin [3], is to start with just a handful of "seed" tuples for the relation of interest, and automatically discover extraction patterns for the task. These patterns, in turn, help discover new tuples for the relation, which could be used as new seed tuples for a next iteration of the process. In practice, however, this bootstrapping approach requires distinguishing between valid and invalid tuples proposed by the system. We present general Expectation-Maximization (EM) algorithms for estimating pattern and tuple confidence. Our specific contributions include:

- A formalization of the pattern confidence estimation problem (Section 2).
- A general EM-based method for estimating the confidence of automatically generated patterns and the extracted relation tuples (Section 3).
- An evaluation of different pattern weighting methods over multiple relation extraction tasks (Section 5).

*Microsoft Research Email:eugeneag@microsoft.com

## 2 Partially Supervised Relation Extraction

Our goal is to extract structured relations between *named entities* (e.g., a company name, a location name, or a name of a drug or a disease) from unstructured documents with minimal human effort. These entities can be identified in a document by a *named entity tagger* (e.g., [2]), which assigns a predefined *entity type* to each entity (e.g., all recognized "company" entities will be assigned the "ORGANIZATION" type). To determine if a tuple is a valid instance of a relation, we need to examine the documents –and associated *text contexts*– where the tuple occurs. For example, a text fragment "The headquarters of Google are in Mountain View" supports the validity of tuple $t = \langle$*Google, Mountain View*$\rangle$ for the relation *CompanyHeadquarters*. More formally, given a relation $R(A_1, \ldots, A_n)$ and a document collection $D$, we want to extract all valid tuples for $R$ from $D$, using the text contexts associated with each tuple to estimate its validity.

While our confidence estimation approach is general, for concreteness we describe it in the context of the *Snowball* information extraction system [1]. *Snowball* extracts a relation from text by starting with just a handful of example tuples for the relation. A *Snowball* extraction pattern as a tuple $\langle acceptor_1, e_1, acceptor_2, e_2, \ldots, e_n, acceptor_{n+1} \rangle$, where $acceptor_1, \ldots, acceptor_{n+1}$ are rules for determining whether a text span is "appropriate" or not, and $e_1, \ldots, e_n$ are the named-entity types or tags of the relation attributes, in the order in which the pattern expects them to appear. To match a text context, each acceptor assigns a score to the respective text span, which determines the degree to which the text span is deemed appropriate for the target relation.

We explored three complementary classes of acceptor rules for *Snowball*:

- **String Match Acceptors (*DIPRE*)**: A string acceptor consists of a single rule comprised of a text string and a scoring function that returns 1 if a given text span matches the text string associated with the rule and 0 otherwise. This approach to relation extraction is proposed in [3], where extraction patterns equivalent to those using string-match acceptors were used for extracting relations from unstructured HTML documents.

- **Vector Space Acceptors (*VS*)**: Patterns that require exact string matches may be too rigid a choice for many relations. For example, text spans "renovated headquarters in," "'s spacious headquarters in," and many other analogous variations should be acceptable for the *Company-Headquarters* relation. In order to match a text span with the corresponding vector-space acceptor $VS_i$, we represent $s_i$ as a vector of features $v_i$ with associated weights. We compute the *score* of $v_i$ and vector-space acceptor $VS_i$ as the cosine similarity [12] between $v_i$ and $VS_i$. The overall score of the text context and a vector-space pattern is a linear combination of the respective *Score* values. In order to generate a pattern, *Snowball* groups occurrences of known tuples in documents, if the contexts surrounding the tuples are similar.

- **Classifier Acceptors** *NB*: *Snowball* can use as an acceptor rule a *classifier* trained to assign higher probability to the tuples extracted from "valid" contexts than to the tuples from "invalid" contexts. Here, the scoring function is just the score assigned to the text span by the respective classifier. In our setting we train a Naive-Bayes classifier on occurrences of seed tuples as positive examples and example contexts automatically recognized as containing invalid tuples.

**2.1 Extracting and Evaluating New Tuples** After generating extraction patterns for a target relation and a document collection, *Snowball* scans the collection documents to discover new tuples using Algorithm 1. *Snowball* first identifies candidate text contexts that contain the target entities. A candidate tuple $t_{CA}$ is generated from a candidate context $C_{CA}$ if there is at least a pattern $P$ such that $Match(P, C_{CA}) \geq \tau_{score}$, where $\tau_{score}$ is the *extraction similarity threshold* for the minimum degree of match between a text context and an extraction pattern. If the *BestScore* value of a candidate text context and an extraction pattern is at least $\tau_{score}$, we add the candidate tuple $t_{CA}$ to the set of *CandidateTuples*. Each candidate tuple will, have a number of patterns that helped generate it with an associated degree of match.

*Snowball* uses the extracted tuples as seed to further expand the set of extraction patterns. Invalid tuples can potentially be used as seed tuples for a later iteration of the algorithm. This might result in even more invalid tuples being generated. To prevent this, we only keep tuples that have a high *confidence*. To estimate the confidence of a candidate tuple $t_{CA}$, we analyze the set of patterns $\{P_1, \ldots, P_k\}$ that produced $t_{CA}$, together with the degree of match between the context in which $t_{CA}$ occurred and the matching pattern. Let us assume for the moment that we know the probability $Prob(P_i)$ with which each pattern $P_i$ generates valid tuples. If these probabilities are independent of each other, then the probability that the candidate text context $C_{CA}$ is valid can be calculated as:

$$Prob(C_{CA}\ is\ valid) \quad = \quad 1 - \prod_{i=1}^{k} (1 - Prob(P_i))$$

Since we do not know the value of *Prob*($P_i$) exactly, we will approximate it as *Conf*($P_i$) (Section 3). By substituting this approximation into the equation above, and scaling the

```
Algorithm ExtractTuples(Patterns, Documents)
  CandidateTuples = {};
  foreach document in Documents do
      candidateContexts = generateTextContexts(document);
      foreach context C_CA in candidateContexts do
          BestScore = 0;
          foreach pattern P in Patterns do
              Score = Match(P, C_CA);
              if Score >= BestScore then
                  BestPattern = P;
                  BestScore = Score;
              end
          end
          if BestScore >= τ_score then
              Create candidate tuple t_CA from C_CA;
              Associate pattern BestPattern with t_CA,
                with match degree BestScore;
              Add t_CA to CandidateTuples;
          end
      end
  end
  return CandidateTuples
```

Figure 1: Extracting tuples from documents for a relation

confidence by the degree of match between the pattern and each context, we compute the *confidence* of a candidate tuple $t_{CA}$ generated by extraction patterns $P_1, \ldots, P_k$ as:

$$Conf(t_{CA}) = 1 - \prod_{i=1}^{k} (1 - Conf(P_i) \cdot Match(P_i, C^i_{CA})) \qquad (2.1)$$

where $C^i_{CA}$ is the text context associated with the occurrence of $t_{CA}$ that matched $P_i$ with highest degree of match, adjusted by the degree of match between $P_i$ and $C^i_{CA}$.

After specifying the confidence of the candidate tuples using the definition above, *Snowball* discards all tuples that have low confidence and include as seed for the next iteration the tuples that have confidence of at least $\tau_{seed}$, where $\tau_{seed}$ is a threshold set during the system tuning. We now turn to estimating the quality of the extraction patterns.

## 3 Estimating Extraction Pattern Confidence

Deriving *good* extraction patterns (and identifying such patterns among the candidates) is a challenging task. We consider two different ways of automatically evaluating the quality of extraction patterns. The first approach exploits relation constraints (e.g., the key constraints) to automatically detect "negative" pattern matches. The second, more general EM-based approach does not require such constraints, focusing instead on the patterns that tend to match the seed tuples.

**Constraint-Based Confidence Estimation:** In this approach, we invalidate an extracted tuple for $R$ if it violates any of the known integrity constraints for $R$. Any such tuple would count as a "negative" match for all patterns that generated it. The constraints that we can utilize include key constraints (i.e., requiring key attributes of a tuple to uniquely determine the remaining attribute values), domain constraints (i.e., the values of an attribute are restricted to a particular domain or range), and identity constraints (e.g., attributes of the same tuple must not be equal).

Intuitively, a selective pattern will have many positive matches, and few negative ones. Accordingly, we define the *confidence* of a pattern $P$ as:

$$Conf(P) = \frac{P.positive}{P.positive + P.negative} \quad (3.2)$$

where $P$.positive is the number of positive matches for $P$ and $P$.negative is the number of negative matches. Our definition of confidence of a pattern above is only one among many possibilities (e.g., as described in [1, 11] and others).

The techniques for automatic pattern evaluation described above rely on the existence of constraints (e.g., the key constraint) over the target relations.

**EM-Based Confidence Estimation:** This approach is based on the observation that an extraction pattern can be regarded as a classifier that assigns some score to a text context. If labeled data were available, we could estimate the accuracy of the "classifiers" by using cross-validation, trusting their predictions accordingly. Unfortunately, in a partially supervised scenario where labeled data is not available, we cannot apply standard cross-validation techniques. However, we can apply recent developments in the area of *partially supervised* text classification (e.g., [10]) to our problem.

Our *ScorePatternsEM* algorithm is shown in Figure 2. As input, the algorithm accepts a set of extraction patterns (*Patterns*), a set of positive seed tuples (*Seed*$_P$), a set of negative seed tuples (*Seed*$_N$, which could be an empty set), a set of all extracted tuples (*Tuples*, all generated by the *Patterns*), and the maximum number of iterations of the algorithm *MaxIteration*. In the initialization stage, all candidate tuples in *Tuples* are assigned a confidence score of $0$, and all *Seed* tuples are assigned the confidence score of $1$. In the expectation stage of the algorithm, we update the confidence of each pattern $P_j$ using the positive and negative counts of matching tuples (Equation 3.2), and compute the confidence of $P_j$ as in Equation 3.2. In the maximization stage of the algorithm, we compute the confidence of each tuple as in Equation 2.1. This process is iterated *MaxIteration* times.

**The *EM-Spy* Pattern Evaluation Algorithm:** The *ScorePatternsEM* algorithm above sometimes converges on excessively high confidence scores for patterns and tuples. To address this problem, we adapt an extension of the classic *EM* classification algorithm, recently presented in [10] for document classification.

The *ScorePatternsEM-Spy* algorithm is shown in Figure 3. During the initialization stage, a set of *Spy* tuples is created as a random subset (e.g., half) of the *Seed*$_P$ positive seed tuples. Then, the *ScorePatternsEM* is initialized with the remaining (non-"spy") *Seed*$_{PS}$ tuples, and an empty set of *Seed*$_N$ negative seed tuples. The *ScorePatternsEM* algorithm runs for *MaxIteration* iterations as described above, adjusting the confidence scores of all candidate tuples not included as *Seed*$_{PS}$ parameter to *ScorePatternsEM*, which includes our *Spy* tuples. We then automatically set the cutoff confidence value for the tuples that are most likely to be valid, based on the distribution of confidence values for the "spy" tuples. The *ScorePatternsEM* algorithm is then reinitialized with the extended *Seed*$_{PS}$ and *Seed*$_N$ sets of positive and negative seed tuples. Finally, when *ScorePatternsEM* completes, we recompute the $\tau_t$ threshold and can use it to select new reliable seed tuples for the next iteration of *Snowball* training.

```
Algorithm ScorePatternsEM(Patterns, Seed_P, [Seed_N,] Tuples,
MaxIteration)
  initialize Conf(T_i) = 0 for each tuple T_i in Tuples;
  initialize Conf(T_s) = 1 for each tuple T_s in Seed_P;
  foreach iteration in 1, ..., MaxIteration do
    //E step: predict pattern confidence scores
    foreach pattern P_j in Patterns do
      Pos = Neg = 0;
      foreach tuple T_i generated by P_j do
        if Conf(T_i) ≥ τ_t then Pos=Pos + 1;
        else Neg= Neg +1;
      end
      // Use Equation 3.2 to update pattern confidence
      UpdatePatternConfidence(P_j, Pos, Neg);
    end
    Normalize pattern scores;
    //M step: recompute tuple confidence scores
    foreach tuple T_i in Tuples do
      if T_i ∈ Seed_P then Conf(T_i)=1;
      else if T_i ∈ Seed_N then Conf(T_i)=0;
      // Use Equation 2.1 to update tuple confidence
      else UpdateTupleConfidence(T_i, Patterns);
    end
  end
```

Figure 2: The *EM* Pattern Evaluation Algorithm

We now delve into the experimental setup and evaluation metrics for comparing variations of *Snowball* and other information extraction techniques.

```
Algorithm ScorePatternsEM-Spy(Patterns, Seed_P, Tuples, MaxIteration)
  Spy = randomly selected half of Seed_P tuples;
  Seed_PS = Seed_P - Spy;
  Seed_N = {};
  ScorePatternsEM(Patterns, Seed_PS, Seed_N, Tuples, MaxIteration);
  Set τ_N such that 10% of Spy tuples have Conf < τ_N;
  Set τ_t such that 10% of Spy tuples have Conf ≥ τ_t;
  foreach tuple T_i in Tuples do
    if Conf(T_i) < τ_N then Add T_i to Seed_N;
    if Conf(T_i) ≥ τ_t then Add T_i to Seed_PS;
  end
  ScorePatternsEM(Patterns, Seed_PS, Seed_N, Tuples, MaxIteration);
  Update τ_t such that 90% of the Spy tuples still have Conf ≥ τ_t;
  return τ_t;
```

Figure 3: The *EM-Spy* Pattern Evaluation Algorithm

## 4 Experimental Setup and Evaluation Metrics

**Data Sets and Relations:** We used three relations extracted from a collection of 145,000 articles from the New York Times from 1996, available as part of the North American News Text Corpus[1]. The relation statistics are summarized in Figure 4.

**Evaluation Methodology: The *Ideal* Relation:** The ultimate goal is to create a high-quality, comprehensive relation that contains all the valid tuples in the collection. We call such "perfect" relation *Ideal*, and evaluate system performance by comparing the tuples in the extracted relation, to those in the *Ideal*. For this, human annotators with the appropriate domain knowledge created an *Ideal* table manually for each relation. Because of the human effort involved in the labeling process, we were forced to restrict the sample size to about 100 documents for each relation combination.

**Evaluation Metrics:** After identifying *Ideal*, we compare it against the tuples produced by the system, *Extracted*, using adapted precision and recall metrics from information retrieval [12]. Given the table *Ideal* and the *ExtractedFiltered* table that we have just created, we define *Recall* as:

$$Recall = \frac{|ExtractedFiltered \bigcap Ideal|}{|Ideal|} \tag{4.3}$$

where the intersection between relations is computed using an approximate string match over attribute values. We give more details on how we handle variations of attribute names in [1]. Similarly, we define *Precision* as:

$$Precision = \frac{|ExtractedFiltered \bigcap Ideal|}{|ExtractedFiltered|} \tag{4.4}$$

**Extraction Methods Compared:** We compared three variations of *Snowball* with two other systems, namely *Baseline* and our implementation of *DIPRE*. These last two methods require minimal or no training input from the user, and hence are comparable. *Snowball*, *DIPRE*, and *Baseline* all rely on a named-entity tagger to identify the entities in the document. For this, we used LingPipe [2], which can be extended with custom gazetteers such as lists of disease names. In contrast, state-of-the-art information extraction systems require substantial manual labor for training.

- *Baseline*: This is our frequency-only baseline with co-occurrence statistics collected over the whole collection (described in [1].
- *DIPRE*: This is our implementation of the *DIPRE* system with named-entity tags (described in [3]).
- *Constraint*: This is the *Snowball* system with Vector Space (*VS*) acceptors, using *constraint*-based confidence estimation.
- *NB-EM*: This is the *Snowball* system with Naive-Bayes (*NB*) classifier acceptors, using *ScorePatternsEM* confidence estimation algorithm.
- *VS-EM-Spy*: This is the *Snowball* system with *VS* acceptors, using *ScorePatternsEM-Spy* confidence estimation algorithm.

Additionally, we compare *Proteus*, an information extraction system developed for extracting a superset of the *DiseaseOutbreaks* relation [8][2]. *Proteus* was tuned on a different document collection from our test collection. Nevertheless, we believe this is a fair comparison since none of the systems were specifically tuned for the test collection.

[1] http://www.ldc.upenn.edu

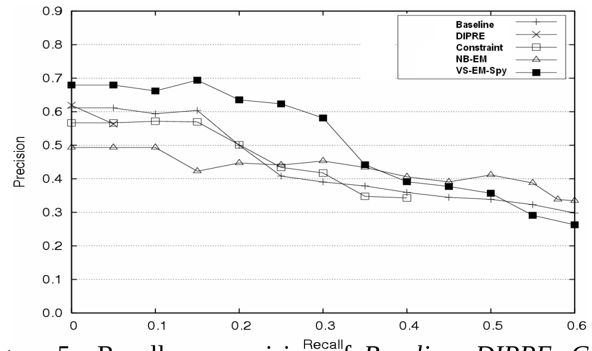


Figure 5: Recall vs. precision of *Baseline*, *DIPRE*, *Constraint*, *NB-EM*, and *VS-EM-Spy* (*CompanyHeadquarters*).

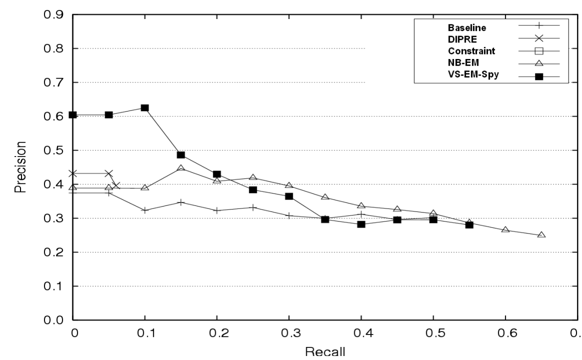


Figure 6: Recall vs. precision of *Baseline*, *DIPRE*, *Constraint*, *NB-EM*, and *VS-EM-Spy* (*MergersAcquisitions*).

## 5 Experimental Results

Figure 5 reports the system accuracy of extracting the *CompanyHeadquarters* relation using our *Ideal* precision and recall definitions. As we can see, both *NB-EM* and *VS-EM-Spy* perform on par with the *Constraint* method on this task. Observe also the strong performance of *Baseline*: for frequently occurring tuples *Baseline* rivals and sometimes exceeds in accuracy the "real" extraction systems that consider the text contexts around the tuples.

Figure 6 reports the extraction accuracy for a more "difficult" relation, *MergersAcquisitions*. As we conjectured, for this relation (which lacks true integrity constraints), the *EM*-based methods have higher accuracy than the constraint-based method. For *Constraint* method, we used the *Target* attribute as key (i.e., a target company is only acquired by one buyer); we also enforced the identity constraint, stating that a company cannot buy itself.

We now evaluate the extraction systems accuracy over the

[2] While the *Proteus* system is not publicly distributed, we were allowed to use an instance that was tuned for extracting infectious disease outbreaks, with kind help and permission from Roman Yangarber, Ralph Grishman, and Silja Hattunen at New York University.

| *Relation* | Description | *Annotated Documents* | \|*Ideal*\| |
|---|---|---|---|
| *CompanyHeadquarters(Organization, Location)* | Name and location of company headquarters | 65 | 124 |
| *MergersAcquisitions(Buyer, Target)* | Name of the buyer and the acquisition target (victim) | 68 | 69 |
| *DiseaseOutbreaks(Disease, Location)* | Name and location of a reported disease outbreak | 125 | 78 |

Figure 4: Statistics on the test relations and labeled test documents.

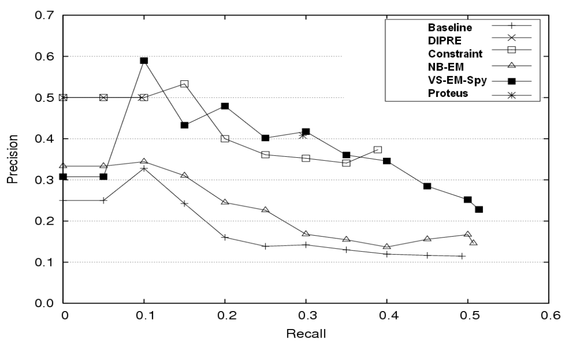


Figure 7: Recall vs. precision of *Baseline*, *DIPRE*, *Constraint*, *NB-EM*, and *VS-EM-Spy*, and *Proteus* (*DiseaseOutbreaks*).

*DiseaseOutbreaks* relation. We compare the extraction systems above and *Proteus*, a state-of-the-art *manually-tuned* information extraction system. As we can see in Figure 7, *Snowball* variations with no specific tuning for *DiseaseOutbreaks* produce results that are comparable with those for *Proteus*. In fact, *Snowball* manages to achieve recall that is substantially higher than that of *Proteus* while maintaining competitive precision.

## 6 Related Work

A partially supervised approach for extracting relations was pioneered by *DIPRE* [3], and extended by others (e.g., references [1, 14, 13, 9]). Recently, [6] presented the URNS model for probabilistically assigning weights to extracted tuples in the supervised setting (i.e., the patterns are specified by hand). Reference [15] applies rational kernels for relation extraction, which assign a confidence to the extracted entities. [4] present method for estimating confidence of extraction for the named-entity recognition tasks. To the best of our knowledge, our work is the first to develop a general confidence estimation method for partially supervised relation extraction that does not rely on relation constraints or other heuristics.

## 7 Conclusions

We presented, to our knowledge, the first general method for estimating confidence for partially supervised relation extraction that models the confidence of automatically derived extraction patterns. Our method can be used for relations with no exploitable integrity constraints with the minimum of human effort. We implemented and evaluated our algorithms for different types of patterns including previously explored vector-space model, as well as for string-match and classification-based extraction patterns. We showed that our general EM-based confidence estimation method improves extraction accuracy over heuristic-based methods, allowing a partially supervised relation extraction system to achieve accuracy comparable to a sophisticated manually tuned state-of-the-art information extraction system.

## References

[1] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*, June 2000.

[2] Breck Baldwin and Bob Carpenter. The LingPipe System, 2003. http://www.alias-i.com/lingpipe/index.html.

[3] Sergey Brin. Extracting patterns and relations from the World-Wide Web. In *Proceedings of the 1998 International Workshop on the Web and Databases (WebDB'98)*, March 1998.

[4] Aron Culotta and Andrew McCallum. Confidence estimation for information extraction. In *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2004.

[5] Arthur P. Dempster, Nan M. Laird, , and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 1977.

[6] Douglas Downey, Oren Etzioni, and Stephen Soderland. A probabilistic model of redundancy in information extraction. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

[7] Ralph Grishman. Information extraction: Techniques and challenges. In *Information Extraction (International Summer School SCIE-97)*. Springer-Verlag, 1997.

[8] Ralph Grishman, Silja Huttunen, and Roman Yangarber. Real-time event extraction for infectious disease outbreaks. In *Proceedings of Human Language Technology Conference (HLT)*, 2002.

[9] Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. Discovering relations among named entities from large corpora. In *Proceedings of the Annual Meeting of Association of Computational Linguistics (ACL)*, 2004.

[10] Bing Liu, Wee Sun Lee, Philip S Yu, and Xiaoli Li. Partially supervised classification of text documents. In *Proceedings of the ICML Conference*, 2002.

[11] Ellen Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[12] Gerard Salton. *Automatic Text Processing: The transformation, analysis, and retrieval of information by computer*. Addison-Wesley, 1989.

[13] Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. Unsupervised discovery of scenario-level patterns for information extraction. In *Proceedings of Conference on Applied Natural Language Processing ANLP-NAACL*, 2000.

[14] Jeonghee Yi and Neel Sundaresan. Mining the web for acronyms using the duality of patterns and relations. In *Proceedings of the 1999 Workshop on Web Information and Data Management*, 1999.

[15] Shubin Zhao and Ralph Grishman. Extracting relations with integrated information using kernel methods. In *Proceedings of the annual meeting of ACL*, 2005.