Preemptive Information Extraction using Unrestricted Relation Discovery

Yusuke Shinyama **Satoshi Sekine**

New York University
715, Broadway, 7th Floor
New York, NY, 10003
{yusuke,sekine}@cs.nyu.edu

Abstract

We are trying to extend the boundary of Information Extraction (IE) systems. Existing IE systems require a lot of time and human effort to tune for a new scenario. Preemptive Information Extraction is an attempt to automatically create all feasible IE systems in advance without human intervention. We propose a technique called Unrestricted Relation Discovery that discovers all possible relations from texts and presents them as tables. We present a preliminary system that obtains reasonably good results.

1 Background

Every day, a large number of news articles are created and reported, many of which are unique. But certain types of events, such as hurricanes or murders, are reported again and again throughout a year. The goal of Information Extraction, or IE, is to retrieve a certain type of news event from past articles and present the events as a table whose columns are filled with a name of a person or company, according to its role in the event. However, existing IE techniques require a lot of human labor. First, you have to specify the type of information you want and collect articles that include this information. Then, you have to analyze the articles and manually craft a set of patterns to capture these events. Most existing IE research focuses on reducing this burden by helping people create such patterns. But each time you want to extract a different kind of information, you need to repeat the whole process: specify articles and adjust its patterns, either manually or semi-automatically. There is a bit of a dangerous pitfall here. First, it is hard to estimate how good the system can be after months of work. Furthermore, you might not know if the task is even doable in the first place. Knowing what kind of information is easily obtained in advance would help reduce this risk.

An IE task can be defined as finding a relation among several entities involved in a certain type of event. For example, in the MUC-6 management succession scenario, one seeks a relation between COMPANY, PERSON and POST involved with hiring/firing events. For each row of an extracted table, you can always read it as "COMPANY hired (or fired) PERSON for POST." The relation between these entities is retained throughout the table. There are many existing works on obtaining extraction patterns for pre-defined relations (Riloff, 1996; Yangarber et al., 2000; Agichtein and Gravano, 2000; Sudo et al., 2003).

Unrestricted Relation Discovery is a technique to automatically discover such relations that repeatedly appear in a corpus and present them as a table, with absolutely no human intervention. Unlike most existing IE research, a user does not specify the type of articles or information wanted. Instead, a system tries to find all the kinds of relations that are reported multiple times and can be reported in tabular form. This technique will open up the possibility of trying new IE scenarios. Furthermore, the system itself can be used as an IE system, since an obtained relation is already presented as a table. If this system works to a certain extent, tuning an IE system becomes a search problem: all the tables are already built "preemptively." A user only needs to search for a relevant table.

Article	dump	be-hit
2005-09-23	Katrina	New Orleans
2005-10-02	Longwang	Taiwan
2005-11-20	Gamma	Florida

Keywords: storm, evacuate, coast, rain, hurricane

Table 1: Sample discovered relation.

We implemented a preliminary system for this technique and obtained reasonably good performance. Table 1 is a sample relation that was extracted as a table by our system. The columns of the table show article dates, names of hurricanes and the places they affected respectively. The headers of the table and its keywords were also extracted automatically.

2 Basic Idea

In Unrestricted Relation Discovery, the discovery process (i.e. creating new tables) can be formulated as a clustering task. The key idea is to cluster a set of articles that contain entities bearing a similar relation to each other in such a way that we can construct a table where the entities that play the same role are placed in the same column.

Suppose that there are two articles A and B, and both report hurricane-related news. Article A contains two entities "Katrina" and "New Orleans", and article B contains "Longwang" and "Taiwan". These entities are recognized by a Named Entity (NE) tagger. We want to discover a relation among them. First, we introduce a notion called "basic pattern" to form a relation. A basic pattern is a part of the text that is syntactically connected to an entity. Some examples are "`X is hit`" or "`Y's residents`". Figure 1 shows several basic patterns connected to the entities "Katrina" and "New Orleans" in article A. Similarly, we obtain the basic patterns for article B. Now, in Figure 2, both entities "Katrina" and "Longwang" have the basic pattern "`headed`" in common. In this case, we connect these two entities to each other. Furthermore, there is also a common basic pattern "`was-hit`" shared by "New Orleans" and "Taiwan". Now, we found two sets of entities that can be placed in correspondence at the same time. What does this mean? We can infer that both entity sets ("Katrina"-"New Orleans" and "Longwang"-"Taiwan") represent a certain relation that has something in common: *a hurricane name and the place it affected*. By finding multiple parallel correspondences between two articles, we can estimate the similarity of their relations.

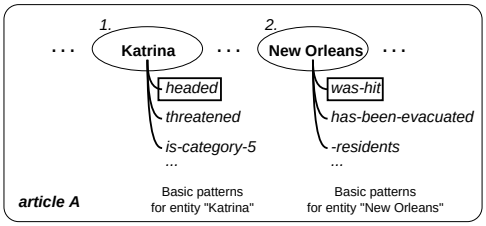
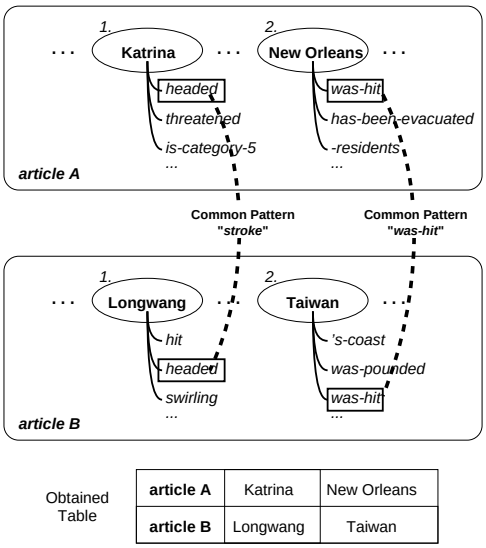


Figure 1: Obtaining basic patterns.



Obtained Table

article A	Katrina	New Orleans
article B	Longwang	Taiwan

Figure 2: Finding a similar relation from two articles.

Generally, in a clustering task, one groups items by finding similar pairs. After finding a pair of articles that have a similar relation, we can bring them into the same cluster. In this case, we cluster articles by using their basic patterns as features. However, each basic pattern is still connected to its entity so that we can extract the name from it. We can consider a basic pattern to represent something like the "role" of its entity. In this example, the entities that had "`headed`" as a basic pattern are *hurricanes*, and the entities that had "`was-hit`" as a basic pattern are *the places it affected*. By using basic patterns, we can align the entities into the corresponding column that represents a certain role in the relation. From this example, we create a two-by-two table, where each column represents the roles of the entities, and each row represents a different article, as shown in the bottom of Figure 2.

We can extend this table by finding another article

in the same manner. In this way, we gradually extend a table while retaining a relation among its columns. In this example, the obtained table is just what an IE system (whose task is to find a hurricane name and the affected place) would create.

However, these articles might also include other things, which could represent different relations. For example, the governments might call for help or some casualties might have been reported. To obtain such relations, we need to choose different entities from the articles. Several existing works have tried to extract a certain type of relation by manually choosing different pairs of entities (Brin, 1998; Ravichandran and Hovy, 2002). Hasegawa et al. (2004) tried to extract multiple relations by choosing entity types. We assume that we can find such relations by trying all possible combinations from a set of entities we have chosen in advance; some combinations might represent a hurricane and government relation, and others might represent a place and its casualties. To ensure that an article can have several different relations, we let each article belong to several different clusters.

In a real-world situation, only using basic patterns sometimes gives undesired results. For example, "*(President) Bush flew to Texas*" and "*(Hurricane) Katrina flew to New Orleans*" both have a basic pattern "`flew to`" in common, so "Bush" and "Katrina" would be put into the same column. But we want to separate them in different tables. To alleviate this problem, we put an additional restriction on clustering. We use a bag-of-words approach to discriminate two articles: if the word-based similarity between two articles is too small, we do not bring them together into the same cluster (i.e. table). We exclude names from the similarity calculation at this stage because we want to link articles about the same type of event, not the same instance. In addition, we use the frequency of each basic pattern to compute the similarity of relations, since basic patterns like "say" or "have" appear in almost every article and it is dangerous to rely on such expressions.

Increasing Basic Patterns

In the above explanation, we have assumed that we can obtain enough basic patterns from an article. However, the actual number of basic patterns that one can find from a single article is usually not enough, because the number of sentences is rather small in comparison to the variation of expressions. So having two articles that have multiple basic patterns in common is very unlikely. We extend the number of articles for obtaining basic patterns by using a cluster of comparable articles that report the same event instead of a single article. We call this cluster of articles a "basic cluster." Using basic clusters instead of single articles also helps to increase the redundancy of data. We can give more confidence to repeated basic patterns.

Note that the notion of "basic cluster" is different from the clusters used for creating tables explained above. In the following sections, a cluster for creating a table is called a "metacluster," because this is a cluster of basic clusters. A basic cluster consists of a set of articles that report the same event which happens at a certain time, and a metacluster consists of a set of events that contain the same relation over a certain period.

We try to increase the number of articles in a basic cluster by looking at multiple news sources simultaneously. We use a clustering algorithm that uses a vector-space-model to obtain basic clusters. Then we apply cross-document coreference resolution to connect entities of different articles within a basic cluster. This way, we can increase the number of basic patterns connected to each entity. Also, it allows us to give a weight to entities. We calculate their weights using the number of occurrences within a cluster and their position within an article. These entities are used to obtain basic patterns later.

We also use a parser and tree normalizer to generate basic patterns. The format of basic patterns is crucial to performance. We think a basic pattern should be somewhat specific, since each pattern should capture an entity with some relevant context. But at the same time a basic pattern should be general enough to reduce data sparseness. We choose a predicate-argument structure as a natural solution for this problem. Compared to traditional constituent trees, a predicate-argument structure is a higher-level representation of sentences that has gained wide acceptance from the natural language community recently. In this paper we used a logical feature structure called GLARF proposed by Meyers et al. (2001a). A GLARF converter takes a syntactic tree as an input and augments it with several

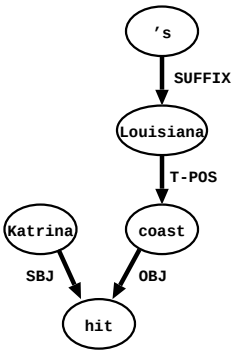


Figure 3: GLARF structure of the sentence "*Katrina hit Louisiana's coast.*"

features. Figure 3 shows a sample GLARF structure obtained from the sentence "*Katrina hit Louisiana's coast.*" We used GLARF for two reasons: first, unlike traditional constituent parsers, GLARF has an ability to regularize several linguistic phenomena such as participial constructions and coordination. This allows us to handle this syntactic variety in a uniform way. Second, an output structure can be easily converted into a directed graph that represents the relationship between each word, without losing significant information from the original sentence. Compared to an ordinary constituent tree, it is easier to extract syntactic relationships. In the next section, we discuss how we used this structure to generate basic patterns.

3 Implementation

The overall process to generate basic patterns and discover relations from unannotated news articles is shown in Figure 4. Theoretically this could be a straight pipeline, but due to the nature of the implementation we process some stages separately and combine them in the later stage. In the following subsection, we explain each component.

3.1 Web Crawling and Basic Clustering

First of all, we need a lot of news articles from multiple news sources. We created a simple web crawler that extract the main texts from web pages. We observed that the crawler can correctly take the main texts from about 90% of the pages from each news site. We ran the crawler every day on several news sites. Then we applied a simple clustering algorithm to the obtained articles in order to find a set of articles that talk about exactly the same news and form a basic cluster.

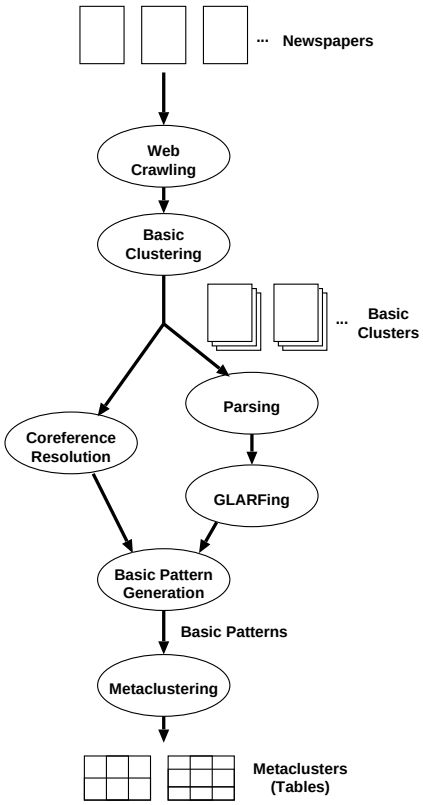


Figure 4: System overview.

We eliminate stop words and stem all the other words, then compute the similarity between two articles by using a bag-of-words approach. In news articles, a sentence that appears in the beginning of an article is usually more important than the others. So we preserved the word order to take into account the location of each sentence. First we computed a word vector from each article:

$$V_w(A) = \text{IDF}(w) \sum_{i \in POS(w,A)} \exp(-\frac{i}{avgwords})$$

where $V_w(A)$ is a vector element of word w in article A, $IDF(w)$ is the inverse document frequency of word w, and $POS(w, A)$ is a list of w's positions in the article. $avgwords$ is the average number of words for all articles. Then we calculated the cosine value of each pair of vectors:

$$Sim(A_1, A_2) = \cos(V(A_1) \cdot V(A_2))$$

We computed the similarity of all possible pairs of articles from the same day, and selected the pairs

whose similarity exceeded a certain threshold (0.65 in this experiment) to form a basic cluster.

3.2 Parsing and GLARFing

After getting a set of basic clusters, we pass them to an existing statistical parser (Charniak, 2000) and rule-based tree normalizer to obtain a GLARF structure for each sentence in every article. The current implementation of a GLARF converter gives about 75% F-score using parser output. For the details of GLARF representation and its conversion, see Meyers et al. (2001b).

3.3 NE Tagging and Coreference Resolution

In parallel with parsing and GLARFing, we also apply NE tagging and coreference resolution for each article in a basic cluster. We used an HMM-based NE tagger whose performance is about 85% in F-score. This NE tagger produces ACE-type Named Entities [1]: PERSON, ORGANIZATION, GPE, LOCATION and FACILITY [2]. After applying single-document coreference resolution for each article, we connect the entities among different articles in the same basic cluster to obtain cross-document coreference entities with simple string matching.

3.4 Basic Pattern Generation

After getting a GLARF structure for each sentence and a set of documents whose entities are tagged and connected to each other, we merge the two outputs and create a big network of GLARF structures whose nodes are interconnected across different sentences/articles. Now we can generate basic patterns for each entity. First, we compute the weight for each cross-document entity E in a certain basic cluster as follows:

$$W_E = \sum_{e \in E} mentions(e) \cdot \exp(-C \cdot firstsent(e))$$

where $e \in E$ is an entity within one article and $mentions(e)$ and $firstsent(e)$ are the number of mentions of entity e in a document and the position of the sentence where entity e first appeared, respectively. C is a constant value which was 0.5 in this experiment. To reduce combinatorial complexity, we took only the five most highly weighted entities from each basic cluster to generate basic patterns. We observed these five entities can cover major relations that are reported in a basic cluster.

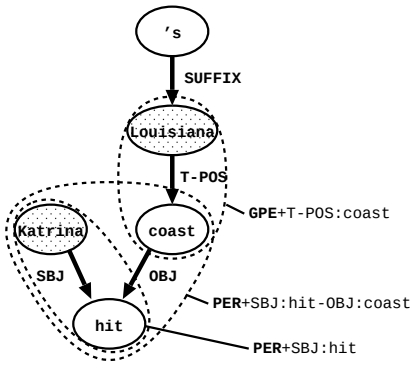


Figure 5: Basic patterns obtained from the sentence "*Katrina hit Louisiana's coast.*"

Next, we obtain basic patterns from the GLARF structures. We used only the first ten sentences in each article for getting basic patterns, as most important facts are usually written in the first few sentences of a news article. Figure 5 shows all the basic patterns obtained from the sentence "*Katrina hit Louisiana's coast.*" The shaded nodes "Katrina" and "Louisiana" are entities from which each basic pattern originates. We take a path of GLARF nodes from each entity node until it reaches any predicative node: noun, verb, or adjective in this case. Since the nodes "hit" and "coast" can be predicates in this example, we obtain three unique paths "*Louisiana*`+T-POS:coast` (*Louisiana's coast*)", "*Katrina*`+SBJ:hit` (*Katrina hit something*)", and "*Katrina*`+SBJ:hit-OBJ:coast` (*Katrina hit some coast*)".

To increase the specificity of patterns, we generate extra basic patterns by adding a node that is immediately connected to a predicative node. (From this example, we generate two basic patterns: "hit" and "hit-coast" from the "Katrina" node.)

Notice that in a GLARF structure, the type of each argument such as subject or object is preserved in an edge even if we extract a single path of a graph. Now, we replace both entities "Katrina" and "Louisiana" with variables

[1] The ACE task description can be found at http://www.itl.nist.gov/iad/894.01/tests/ace/ and the ACE guidelines at http://www.ldc.upenn.edu/Projects/ACE/

[2] The hurricane names used in the examples were recognized as PERSON.

based on their NE tags and obtain parameterized patterns: "*GPE*+T-POS:coast (*Louisiana's coast*)", "*PER*+SBJ:hit (*Katrina hit something*)", and "*PER*+SBJ:hit-OBJ:coast (*Katrina hit some coast*)".

After taking all the basic patterns from every basic cluster, we compute the Inverse Cluster Frequency (ICF) of each unique basic pattern. ICF is similar to the Inverse Document Frequency (IDF) of words, which is used to calculate the weight of each basic pattern for metaclustering.

3.5 Metaclustering

Finally, we can perform metaclustering to obtain tables. We compute the similarity between each basic cluster pair, as seen in Figure 6. X_A and X_B are the set of cross-document entities from basic clusters c_A and c_B, respectively. We examine all possible mappings of relations (parallel mappings of multiple entities) from both basic clusters, and find all the mappings M whose similarity score exceeds a certain threshold. wordsim(c_A, c_B) is the bag-of-words similarity of two clusters. As a weighting function we used ICF:

$$\text{weight}(p) = -\log(\frac{\text{clusters that include p}}{\text{all clusters}})$$

We then sort the similarities of all possible pairs of basic clusters, and try to build a metacluster by taking the most strongly connected pair first. Note that in this process we may assign one basic cluster to several different metaclusters. When a link is found between two basic clusters that were already assigned to a metacluster, we try to put them into all the existing metaclusters it belongs to. However, we allow a basic cluster to be added only if it can fill all the columns in that table. In other words, the first two basic clusters (i.e. an initial two-row table) determines its columns and therefore define the relation of that table.

4 Experiment and Evaluation

We used twelve newspapers published mainly in the U.S. We collected their articles over two months (from Sep. 21, 2005 - Nov. 27, 2005). We obtained 643,767 basic patterns and 7,990 unique types. Then we applied metaclustering to these basic clusters and obtained 302 metaclusters (tables). We then removed duplicated rows and took only the tables that had 3 or more rows. Finally we had 101 tables. The total number the of articles and clusters we used are shown in Table 2.

Source articles	28,009
Basic clusters	5,543
Basic patterns (token)	643,767
Basic patterns (type)	7,990
Metaclusters	302
Metaclusters (rows ≥ 3)	101

Table 2: Articles and obtained metaclusters.

4.1 Evaluation Method

We evaluated the obtained tables as follows. For each row in a table, we added a summary of the source articles that were used to extract the relation. Then for each table, an evaluator looks into every row and its source article, and tries to come up with a sentence that explains the relation among its columns. The description should be as specific as possible. If at least half of the rows can fit the explanation, the table is considered "consistent." For each consistent table, the evaluator wrote down the sentence using variable names ($1, $2, ...) to refer to its columns. Finally, we counted the number of consistent tables. We also counted how many rows in each table can fit the explanation.

4.2 Results

We evaluated 48 randomly chosen tables. Among these tables, we found that 36 tables were consistent. We also counted the total number of rows that fit each description, shown in Table 3. Table 4 shows the descriptions of the selected tables. The largest consistent table was about hurricanes (Table 5). Although we cannot exactly measure the recall of each table, we tried to estimate the recall by comparing this hurricane table to a manually created one (Table 6). We found 6 out of 9 hurricanes [3]. It is worth noting that most of these hurricane names were automatically disambiguated although our NE tagger didn't distinguish a hurricane name from a person

[3]Hurricane Katrina and Longwang shown in the previous examples are not included in this table. They appeared before this period.

```
for each cluster pair (c_A, c_B) {
    X_A = c_A.entities
    X_B = c_B.entities
    for each entity mapping M = [(x_A1, x_B1), ..., (x_An, x_Bn)] ∈ (2^|X_A| × 2^|X_B|) {
        for each entity pair (x_Ai, x_Bi) {
            P_i = x_Ai.patterns ∩ x_Bi.patterns
            pairscore_i = Σ_{p∈P_i} weight(p)
        }
        mapscore = Σ pairscore_i
        if T_1 < |M| and T_2 < mapscore and T_3 < wordsim(c_A.words, c_B.words) {
            link c_A and c_B with mapping M.
        }
    }
}
```

Figure 6: Computing similarity of basic clusters.

Tables:

Consistent tables	36 (75%)
Inconsistent tables	12
Total	48

Rows:

Rows that fit the description	118 (73%)
Rows not fitted	43
Total	161

Table 3: Evaluation results.

Description	Rows
Storm $1(PER) probably affected $2(GPE).	8/16
Nominee $2(PER) must be confirmed by $1(ORG).	4/7
$1(PER) urges $2(GPE) to make changes.	4/6
$1(GPE) launched an attack in $2(GPE).	3/5
$1(PER) ran against $2(PER) in an election.	4/5
$2(PER) visited $1(GPE) on a diplomatic mission.	2/4
$2(PER) beat $1(PER) in golf.	4/4
$2(GPE) soldier(s) were killed in $1(GPE).	3/3
$2(PER) ran for governor of $1(GPE).	2/3
Boxer $1(PER) fought boxer $2(PER).	3/3

Table 4: Description of obtained tables and the number of fitted/total rows.

name. The second largest table (about nominations of officials) is shown in Table 7.

We reviewed 10 incorrect rows from various tables and found 4 of them were due to coreference errors and one error was due to a parse error. The other 4 errors were due to multiple basic patterns distant from each other that happened to refer to a different event reported in the same cluster. The causes of the one remaining error was obscure. Most inconsistent tables were a mixture of multiple relations and some of their rows still looked consistent.

We have a couple of open questions. First, the overall recall of our system might be lower than existing IE systems, as we are relying on a cluster of comparable articles rather than a single document to discover an event. We might be able to improve this in the future by adjusting the basic clustering algorithm or weighting schema of basic patterns. Secondly, some combinations of basic patterns looked inherently vague. For example, we used the two basic patterns "`pitched`" and "`' s-series`" in the following sentence (the patterns are underlined):

> Ervin Santana pitched 5 1-3 gutsy innings in his postseason debut for the Angels, Adam Kennedy hit a go-ahead triple that sent Yankees outfielders crashing to the ground, and Los Angeles beat New York 5-3 Monday night in the decisive Game 5 of their AL playoff series.

It is not clear whether this set of patterns can yield any meaningful relation. We are not sure how much this sort of table can affect overall IE performance.

5 Conclusion

In this paper we proposed Preemptive Information Extraction as a new direction of IE research. As its key technique, we presented Unrestricted Relation Discovery that tries to find parallel correspondences between multiple entities in a document, and perform clustering using basic patterns as features. To increase the number of basic patterns, we used a cluster of comparable articles instead of a single document. We presented the implementation of our preliminary system and its outputs. We obtained dozens of usable tables.

Article	1:dump	2:coast
2005-09-21 (1)	Rita	Texas
2005-09-23	Rita	New Orleans
2005-09-25	Bush	Texas
2005-09-26	Damrey	Hainan
2005-09-27 (2)	Damrey	Vietnam
2005-10-01	Rita	Louisiana
2005-10-02	Otis	Mexico
2005-10-04	Longwang	China
2005-10-05	Stan	Mexico
2005-10-06	Tammy	Florida
2005-10-07	Tammy	Georgia
2005-10-19 (3)	Wilma	Florida
2005-10-25	Wilma	Cuba
2005-10-25	Wilma	Massachusetts
2005-10-28	Beta	Nicaragua
2005-11-20	Gamma	Florida

1. More than 2,000 National Guard troops were put on active-duty alert to assist as **Rita** slammed into the string of islands and headed west, perhaps toward **Texas**. ...
2. **Typhoon Damrey** smashed into **Vietnam** on Tuesday after killing nine people in China, ...
3. Oil markets have been watching **Wilma**'s progress nervously, ... but the threat to energy interests appears to have eased as forecasters predict **the storm** will turn toward **Florida**. ...

Table 5: Hurricane table ("Storm $1(PER) probably affected $2(GPE).") and the actual expressions we used for extraction.

Hurricane	Date (Affected Place)	Articles
Philippe	Sep 17-20 (?)	6
* Rita	Sep 17-26 (Louisiana, Texas, etc.)	566
* Stan	Oct 1-5 (Mexico, Nicaragua, etc.)	83
* Tammy	Oct 5-? (Georgia, Alabama)	18
Vince	Oct 8-11 (Portugal, Spain)	12
* Wilma	Oct 15-25 (Cuba, Honduras, etc.)	368
Alpha	Oct 22-24 (Haiti, Dominican Rep.)	80
* Beta	Oct 26-31 (Nicaragua, Honduras)	55
* Gamma	Nov 13-20 (Belize, etc.)	36

Table 6: Hurricanes in North America between mid-Sep. and Nov. (from Wikipedia). Rows with a star (*) were actually extracted. The number of the source articles that contained a mention of the hurricane is shown in the right column.

Article	1:confirm	2:be-confirmed
2005-09-21	Senate	Roberts
2005-10-03	Supreme Court	Miers
2005-10-20	Senate	Bush
2005-10-26	Senate	Sauerbrey
2005-10-31	Senate	Mr. Alito
2005-11-04	Senate	Alito
2005-11-17	Fed	Bernanke

Table 7: Nomination table ("Nominee $2(PER) must be confirmed by $1(ORG).")

Acknowledgements

This research was supported by the National Science Foundation under Grant IIS-00325657. This paper does not necessarily reflect the position of the U.S. Government. We would like to thank Prof. Ralph Grishman who provided useful suggestions and discussions.

References

Eugene Agichtein and L. Gravano. 2000. Snowball: Extracting Relations from Large Plaintext Collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries (DL-00)*.

Sergey Brin. 1998. Extracting Patterns and Relations from the World Wide Web. In *WebDB Workshop at EDBT '98*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL-2000*.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the Annual Meeting of Association of Computational Linguistics (ACL-04)*.

Adam Meyers, Ralph Grishman, Michiko Kosaka, and Shubin Zhao. 2001a. Covering Treebanks with GLARF. In *ACL/EACL Workshop on Sharing Tools and Resources for Research and Education*.

Adam Meyers, Michiko Kosaka, Satoshi Sekine, Ralph Grishman, and Shubin Zhao. 2001b. Parsing and GLARFing. In *Proceedings of RANLP-2001*, Tzigov Chark, Bulgaria.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ellen Riloff. 1996. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An Improved Extraction Pattern Representation Model for Automatic IE Pattern Acquisition. In *Proceedings of the Annual Meeting of Association of Computational Linguistics (ACL-03)*.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Unsupervised Discovery of Scenario-Level Patterns for Information Extraction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-00)*.