# Being Lazy and Preemptive at Learning toward Information Extration

by

*Yusuke Shinyama*

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Computer Science Department

Courant Institute of Mathematical Sciences

New York University

August 2007

Satoshi Sekine

# Abstract

This thesis proposes a novel approach for exploring Information Extraction scenarios. Information Extraction, or IE, is a task aiming at finding events and relations in natural language texts that meet a user's demand. However, it is often difficult to formulate, or even define such events that satisfy both a user's need and technical feasibility. Furthermore, most existing IE systems need to be tuned for a new scenario with proper training data in advance. So a system designer usually needs to understand what a user wants to know in order to maximize the system performance, while the user has to understand how the system will perform in order to maximize his/her satisfaction.

In this thesis, we focus on maximizing the variety of scenarios that the system can handle instead of trying to improve the accuracy of a particular scenario. In traditional IE systems, a relation is defined *a priori* by a user and is identified by a set of patterns that are manually crafted or acquired in advance. We propose a technique called Unrestricted Relation Discovery, which defers determining what is a relation and what is not until the very end of the processing so that a relation can be defined *a posteriori*. This laziness gives huge flexibility to the types of relations the system can handle. Furthermore, we use the notion of recurrent relations to measure how useful each relation is. This way, we can discover new IE scenarios without fully specifying definitions or patterns, which leads to Preemptive Information Extraction, where a system can provide a user a portfolio of extractable relations and let the user choose them.

We used one year news articles obtained from the Web as a development set. We discovered dozens of scenarios that are similar to the existing scenarios tried by many IE systems, as well as new scenarios that are relatively novel. We have evaluated the existing scenarios with Automatic Content Extraction (ACE) event corpus and obtained reasonable performance. We believe this system will shed new light on IE research by giving various experimental IE scenarios.

# Acknowledgments

First of all, I would like to thank my research advisor, Satoshi Sekine and Ralph Grishman. They were very supportive and tolerant.

I thank another reader Dan I. Melamed, who gave me various advices. I also thank Ernest Davis and Robert Grimm who kindly became the committee members.

I thank many colleagues. Adam Meyers gave me various advices as well as his expertise in linguistics. I also thank Javier Artiles, Edgar Gonzales, Heng Ji, Shasha Liao, Cristina Mota, Charles Shoopak, Joseph Turian, Ben Wellington and David Westbrook, for their advices and encouragements.

Finally, I thank my parents, Kazuo and Kazuko. I appreciate their support, and especially their tremendous anxiety about the loafer son. I have been long aware of it, though you never expressed it in front of me.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 What is Information Extraction?

We are doing Information Extraction (IE) every day. IE can be described as finding a small piece of information from a large amount of text. For example, when people search the web, they usually try to find the information they need from the search snippets. When they read classified ads to find a particular item such as apartment rooms, they normally try to pick up the small description about a room as well as its price and location.

Information Extraction is defined as a computational task to convert unstructured information such as natural language texts into structured information like lists or tables. A piece of extracted information is normally called a "relation," which is similar to a mathematical relation between multiple objects. However, in Information Extraction, an item involved with each relation is not a number or set, but an object, generally referred to by a Named Entity (NE) such as the name of a person or monetary quantity. For example, the following sentences include a murderer-and-victim relation whose Named Entities are both person names:

- `Alice killed Bob.`
- `Charlie murdered Dave.`

The result of Information Extraction is usually presented to the user in the form of tables. From the above sentences, an IE system might create the table shown in Table 1.1. Each instance of the relation (the murder of Bob and Dave, respectively), or an *event*, is presented as a row and each column has a role that the filled objects should take. We suppose the dates of these events are given by some meta-information attached to each sentence. The system takes unstructured information (sentences) and converts them into structured information (a table).

Normally, an Information Extraction system extracts a particular type of relation that is specified by a user in advance. However, creating an Information Extraction system has been an expensive task due to the flexibility of natural language.

| Date | Murderer | Victim |
|---|---|---|
| 2001.1.1 | Alice | Bob |
| 2002.2.2 | Charlie | Dave |
| ... | ... | ... |

Table 1.1: Murderer and victim table

## 1.2 Problems in Information Extraction

Research in IE can be roughly divided into two types. One type of research attempts to improve the performance (precision and recall) at each stage of the processing. Several basic components such as a Named Entity tagger, a parser and a pattern recognizer are improved along this line of research. The other type of research focuses on expanding the range of possible relations a system can extract with as little human effort as possible. In this section, we illustrate this second issue, which is the main focus of this thesis.

The main part of IE can be formulated as pattern matching, or a classification problem. An IE system tries to determine whether a particular string of text describes a certain relation or not. One of the simplest forms of IE is regular expression matching. For example, to obtain the above murder-victim relations, one can simply search the texts with a regular expression pattern like

"`([A-Za-z]+) (killed|murdered) ([A-Za-z]+).`"

This way, one can extract person names involved with this particular relation. However, natural language expressions are normally much more varied. For example, the tense of the verb "`kill`" or "`murder`" might not be limited to the past tense. An additional word or phrase might be inserted in the sentence. The name of the person might not expressed with a single token, or an actual name might not be directly referred to as "`Alice`," but referred to by a pronoun like "`she`." Furthermore, there are a lot of possible expressions that imply killing other than the word "`kill`" or "`murder`."

To absorb this variety, an IE system is normally composed of several layers of components, such as a part-of-speech tagger, a Named Entity (NE) recognizer, a (shallow or deep) parser in order to handle these expressions in a uniform way. These layers form a pipeline of the processing where each layer tries to reduce the variations of expressions and provides uniform outputs to the next layer. However, at the highest level, an IE system still has to rely on a layer that is manually crafted: a specification of relations to be extracted.

In IE, a description of target relations is often called "scenario." For most current IE systems, there are mainly two ways to obtain the specification of the relations for a given scenario. One is to write a set of patterns using some predefined formal language, just like regular expression patterns. People usually craft their own patterns to use for a scenario. However, these patterns highly depend on that particular scenario, so one has to modify the significant part of an IE system in order to switch from one scenario to another. The other way is to let a machine obtain these patterns automatically from human annotated corpora. However, because these annotations also highly depend on each scenario, one has to spend a huge amount of labor for corpus annotation for every scenario. Unlike other components such as NE recognition or parsing that can be trained independently and reused for many different tasks, the specification for a relation has to be created

every time a user changes his/her scenario. Both approaches suffer from this relatively larger cost compared to other components for IE systems. Furthermore, creating a good specification for a relation does not necessarily guarantee satisfactory extraction results. If a certain type of relation does not appear on the text frequently enough, they cannot be extracted anyway. How one can tell a certain task is feasible in the first place before actually trying it?

## 1.3 Problem We Address

In order to mitigate the cost associated with obtaining specifications and creating patterns from scratch, we propose an alternative approach: to discover a feasible scenario which is close to a user's demand and improve it.

In this thesis, we present a novel approach to perform Information Extraction *without* specifying a scenario. We call this technique Preemptive Information Extraction. Preemptive IE can obtain a large number of relations without any human intervention. It also provides their preliminary extraction results. This gives us an opportunity to explore what kind of relations an IE system can potentially handle. We expect these outputs can be used as a guidepost for future research of Information Extraction.

In order to achieve this goal, we introduced two important ideas. Firstly, we separate the notion of relation *detection* and relation *identification*. Relation detection is the task of extracting a tuple of Named Entities which has *some* relation among them. Relation identification is figuring out what kind of relation these tuples form. This separation allows us to determine what kind of relation *could* be formed, after seeing all the tuples we extracted from news texts.

The second important idea is to perform relation identification as a clustering task. Clustering is a common technique to group things based on their similarities. Since we do not aim for some predefined relations, the obtained tuples normally do not have a clear identity. We use clustering techniques to reveal their identities. In short, our key strategy is "extract everything extractable first and then figure out relations among them." This way, we can obtain relations without using any predefined pattern, which makes Preemptive Information Extraction feasible.

As we explain later, however, there is no clear-cut definition of relations that satisfies every user's need. Therefore we tried to optimize several aspects of the system in the hope of maximizing the usefulness of our system for most users. We focused on tuning two parameters that are likely to be important for discovering new scenarios: the number of relation types a system can obtain and the generality of each relation.

## 1.4 Outline of This Thesis

In this thesis, we first consider the nature of relations in Chapter 2. We discuss various aspects of relations that can be extracted from news articles and present the basic idea of Preemptive Information Extraction. In Chapter 3, we present the detailed implementation. Since our Preemptive IE system is large and complicated, we illustrate its components one by one in each section. The actual data we used and its experimental results are presented in Chapter 4. Then in Chapter 5, we exhibit several anecdotal results and try to draw some remarks. We present the related work in Chapter 6.

# Chapter 2

# Exploring Relations

## 2.1 Two Properties that Make Information Extraction Hard

Over the years, IE researchers have been tackling how to distinguish, or *not* to distinguish, a particular relation. Although these two problems are the other sides of the same coin, the second one has not been paid as much attention as the first one. In this section, we step into an in-depth question about relations; namely, *what really is a relation?*

Mathematically, a relation in Information Extraction is nothing more than a table-like structure whose rows consist of a tuple of Named Entities. However, there are two properties making it difficult to recognize IE relations correctly: the variety of natural language expressions and the generality of IE relations. These are also closely related to each other. In fact, this is another formulation of the problem of relation identification. In this section, we will take a look at each of them and discuss the way to mediate these two.

### 2.1.1 Variety of Natural Language Expressions

Natural language expressions are varied. Creative writers or news editors often try paraphrasing the same fact in various ways to amuse their readers. This has been a major obstacle for IE, because there are thousands of different ways to express the same type of relations:

- `Alice killed Bob.`
- `Alice murdered Bob.`
- `Bob was shot to death by Alice.`

All the above sentences express some kind of attack, or murder, of a person by another person. If an IE system has to capture all of them in one table, it has to identify these expressions (e.g. "`murder`" and "`be shot to death`") as "equivalent" and recognize them as the same type of relation. In the IE research community, there have been many attempts to list these equivalent expressions to describe a particular relation, either manually or automatically. However, now think about the following sentences:

a. `Alice killed Bob.`

b. `Dave was shot to death by Charlie.`

According to the previous example, these two sentences express the same kind of relation, because the expression "`murder`" and "`be shot to death`" is regarded equivalent. But this seems a little awkward to some people, because sentence a. only stated the killing and does not state any associated weapon, whereas sentence b. clearly indicates the use of some firearms. Are they really the "same" relation? Of course, this depends on the user's demand. If s/he wanted to see any sort of killing, the system *must* identify these as the same relation. This leads to another problem in IE, the generality of relations.

### 2.1.2 Generality of IE Relations

In Information Extraction, the definition of relations is normally given by a scenario. Traditionally, IE scenarios were specified by a user as precisely as possible in order to specify its generality to a certain level. These descriptions are carefully written to eliminate all the subjective judgments by humans. For example, the "terror" scenario used in the Message Understanding Conference (MUC-3) evaluation contains more than 1,000 words to describe what is considered as a "terror" act [1]. In the Automatic Context Extraction (ACE) evaluation in 2005, about thirty types of events were defined [12]. Although these ACE descriptions are much shorter compared to the MUC ones, each definition still has several paragraphs. For event types which are similar to each other, they provide a special note about how to distinguish one type from another.

However, when the number of relations a system can handle is increased, the boundary between each relation gets blurred. This is especially true when a system tries to discover a "new" relation which hasn't been observed, because there are various ways of separating one relation from another. Let us take a look at the following examples:

1a. `Walt Disney acquired Pixar.`
1b. `Yankees acquired Alex Rodriguez.`

Traditionally, a relation in IE is identified by its surrounding expressions, or patterns. However, although both above sentences use the same expression "`acquire`," some might think these two relations are not exactly the *same type*. Indeed, in most newspapers, these two events are categorized as different types: one as Business news, the other as Sports news. We could distinguish these two by using some other contextual information. But how can you tell there might be multiple "acquisition" types before looking at actual news articles?

Here is another example:

2a. `Bloomberg beat Ferrer in New York City.` *(Politics)*
2b. `Sharapova beat Henin in Queens.` *(Sports)*

Again, both sentences state a result of competition by two individuals. But many people will see these two as different types of events, therefore they should be put in separated tables. Furthermore,

another different "beat" event might be discovered in future [1]. Therefore, contrary to the previous argument, in this case the system *must not* identify these as the same relation. Of course, one can insist *any* relation is unique and they are by no means regarded as equivalent [2]. And we can't object to this because any event that happens at a different place and different time in the universe is, indeed, unique. However, if we admit this, we cannot "group" events to construct a table, which renders Information Extraction fundamentally impossible in the first place.

### 2.1.3 Ambivalent Constraints

Unfortunately, there is no definite way to solve the above problems completely, because the above two properties conflict with each other. When a user tries to accept as varied forms of expressions as possible, it inevitably extends the notion that a user tries to find. For example, the following patterns roughly indicate $A$'s murder of $B$:

1. $A$ `killed` $B$ `with a gun.`
2. $A$ `fatally stabbed` $B$`.`
3. $A$ `strangled` $B$ `with a rope.`
4. $B$ `was shot to death by` $A$`.`

It is hard to imagine a category that includes the pattern 1., 2. and 3. but does not include pattern 4. Therefore, if a user wants to find 1., 2. and 3. as the *same* relation type (i.e. within one table,) s/he has to automatically accept pattern 4. too. Contrary, if a user only wants a case that involves firearms, s/he has to give up all the patterns but pattern 4. In this case, the former relation has more room of interpretation than the latter one. In other words, the former relation is more *general*.

When we perform IE without specifying a scenario, we cannot really tell the accuracy of its extraction results unless we agree on some sort of guideline. However, it is up to a user how general each relation should be. If a relation is too general, it will probably capture more events but also look pointless. Contrary, if a relation is too specific, it will contain a very few instances. While a natural language expression is a concrete object that can be manipulated by a machine, a relation is an abstract notion that only exists in a person's mind. This ambivalence is somewhat similar to the Precision and Recall problem in Information Retrieval. We normally assume there is some agreement between a user and an IE system designer about the degree of generality of relations, but this problem hasn't been a concern when they decided on the scenario description by themselves.

[1] In the classical AI approach, this problem could be solved by decomposing a word into a more fine-grained set of "semantic ingredients." In this case, one could differentiate victory at election ($\text{beat}_1$) and victory at sports competition ($\text{beat}_2$). However, we have yet to know how many meanings each word has, and if there is any plausible decomposition that most of us can agree on. In the senses defined in WordNet [9], both use of the word "`beat`" fall into the same sense, because they are indeed a result of a competition:

1. (18) beat, beat out, crush, shell, trounce, vanquish – (come out better in a competition, race, or conflict; "Agassi beat Becker in the tennis championship"; "We beat the competition"; "Harvard defeated Yale in the last football game")

[2] For example, the meaning of "kill" might be different depending on the target (a person or a germ). For even legal or technical terms, we cannot fully determine its implication. For example, the definition of the term "first degree murder" is different in New York and Pennsylvania.

| Acquisition of Companies | Acquisition of Baseball Players |
|---|---|
| Walt Disney acquired Pixar. | Yankees acquired Alex Rodriguez. |
| • ... Disney's board ... | • ... Yankees offered ... |
| • ... Disney's shareholder ... | • ... trade for Rodriguez ... |
| • ... purchasing Pixar ... | • ... Rodriguez will play ... |

Table 2.1: Expressions to distinguish two relations

## 2.2 Unrestricted Relation Discovery

Now, our goal is to discover relations that have a certain degree of generality. As we stated in the previous section, IE relations are normally identified by a pattern. In this thesis, we extend this notion and use other contextual expressions as *features* to distinguish relations. For example, a relation for *acquisition of companies* and a relation for *acquisition of baseball players* can be still distinguished by using the expressions shown in Table 2.1.

We use a clustering technique to identify these relations by grouping events that have similar features to each other. However, to obtain the features for a certain relation, we have to identify them first. How can we identify an instance of relation before it is actually distinguished from others? To solve this dilemma, we tried to rethink the whole pipeline of an IE system with the "end-to-end" principle.

### 2.2.1 End-to-End Principle

In 1980, Saltzer et al. introduced an influential idea called "the end-to-end principle [20]." The end-to-end principle was originally introduced in the context of computer networks: it claims that the performance of a communication channel should be measured and tuned between both ends of the channel, rather than the intermediate layers in the network. According to the end-to-end principle, the resource management and error correction should be performed mostly at the end layer. Since only the end layers know the true goal of this communication, the intermediate components must not try to introduce an arbitrary bias that might interfere with the performance of the whole pipeline.

Since an IE system is also composed of multiple layers, we can imagine an IE system as a transformation of information from an unstructured form into a structured one. Reviewing the structure of an IE system from this end-to-end viewpoint, we noticed that the existing approach of IE does not precisely follow this principle for our purpose. In a traditional IE system, relation extraction is performed by pattern matching. For example, a regular expression pattern:

"`([A-Za-z]+) (killed|murdered) ([A-Za-z]+).`"

detects a relation *and* identifies the type of the relation at the same time. By using pattern matching,

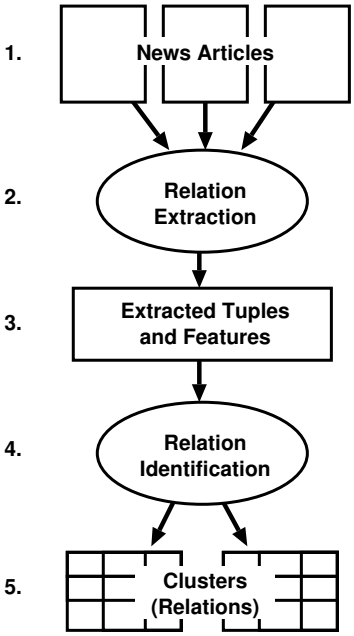


Figure 2.1: Overall algorithm

we can maximize the performance of extraction results for a *particular* relation. Although relation discovery also requires both detecting and identifying a relation, doing these two tasks at once is irrelevant, because in relation discovery we try to maximize the *number* of relation types that have a certain generality, rather than the extraction *performance* for a particular relation. Using any predefined pattern for detecting relations could impose an unnecessary bias that might affect the type of relation we could discover, which violates the end-to-end principle. In order to maximize the number of relation types, we need to perform relation detection and relation identification separately.

### 2.2.2 Overall Algorithm

We can now present the overall algorithm of Unrestricted Relation Discovery, or URD. URD has the following steps (Figure 2.1) :

1. Obtain news articles.
2. List all the possible relation instances. *(Relation Detection)*
3. Obtain the features of significant relation instances.
4. Perform clustering of relation instances and identify each cluster as a type of relation. *(Relation Identification)*
5. Present the obtained clusters in a tabular form.

In the following two subsections, we explain the relation detection and relation identification layer.

### 2.2.3 Relation Detection

As we stated above, the key idea of URD is to perform relation detection and relation identification separately. However, how to detect that there is *some* relation between Named Entities without using any pattern? Going back to the definition in the beginning of this chapter, if you don't limit the type of relations, actually any combination of Named Entities can be a relation instance. Therefore, we can take every combination of tuples of NEs that appear in a document as a potential relation. By separating relation detection and relation identification, we could collect as many features as we want for clustering without imposing an arbitrary restriction on the type of possible relations.

However, listing all the possible relation instances at Step 1. encounters a formidable computational cost. There could be too many possible instances for each document. When an article contains $n$ Named Entities, the number of all possible relations, i.e. the number of all the combinations of NEs can be computed as follows:

$$R(n) = \sum_{i=1...n} \frac{n!}{(n-i)!}$$

If an article contains ten different NEs, the number of possible relation instances is $R(10) = 9864091$. Since we need to obtain the instances for all the articles, the actual number of relation instances and their features we have to handle would be much larger than this. To reduce this cost, we use a set of comparable articles instead of a single article in order to weigh significant relations and eliminate trivial ones.

#### Comparable News Articles

Comparable articles are a set of articles that have the roughly same contents. In the context of Information Extraction, we are especially interested in comparable news articles that report the same event. Comparable news articles are nowadays readily available from multiple news sources. There have been various research attempts to use comparable news articles for acquiring linguistic knowledge.

Comparable news articles give several important benefits to us. Firstly, we can weight the NEs used in comparable articles based on their frequency. Since some NEs are indispensable for describing a certain event, we can expect that those significant NEs appear many times among multiple articles. For example, the following sentences are taken from each article that reports the negotiation about nuclear weapons between the United States and North Korea:

1. `North Korea urged the United States to supply light-water nuclear reactors.`
2. `Condolezza Rice has rejected a statement that North Korea would begin dismantling its nuclear program only if the United States provided a light-water reactor.`
3. `North Korea said Tuesday it would not dismantle its nuclear weapons until the United State first provides an atomic energy reactor.`

It is noticed that the NEs "`the United States`" and "`North Korea`" appear in every article while the NEs such as "`Condolezza Rice`" or "`Tuesday`" don't. Since these NEs are likely to be

more important than others, we can expect a relation between "the United States" and "North Korea" is more important than, say, a relation between "Condolezza Rice" and "Tuesday."

The second purpose of using comparable articles is to enrich the feature set for a relation instance. In this thesis, we use expressions that modify a certain NE as features. Since comparable articles use slightly different expressions to describe the same event, we could collect more varied features for each NE. Furthermore, if a certain expression appears multiple times across different news articles, we can weight the expressions (features) in the same way as we do for the Named Entities. We assume that most newspapers agree on the basic description for a certain event. So even if their expressions are varied, it is likely that there is some tendency of expressions that are used to describe the same event.

By using comparable articles, we can weight both the relations and their features obtained from a certain event. This way, we can greatly reduce the computational cost by limiting the relation instances to only important ones.

### 2.2.4 Relation Identification

After obtaining tuples of Named Entities and their corresponding features, we perform relation identification. Relation identification in URD is done by clustering. In order to discover a new relation, we want to maximize the number of relation types, i.e. the number of clusters. However, as we stated in Section 2.1.3, there is a conflict between the number of relation types and the generality of each relation. Each relation has to be general to some extent, but not too much.

In URD, the generality of each relation can be adjusted as a clustering threshold. Since the clustering threshold determines how similar the items in the same cluster must be, decreasing the threshold means gathering a broader range of items, which is more general. Of course, however, some expressions (features) are more significant than others for us to recognize a certain characteristic of a news event[3]. Although we have yet to know how to choose the features and their weights that best fit to our intuition, we hope the features we present in this thesis will give some indications for future research direction.

Generally, we can expect the following changes as we adjust the clustering threshold:

- When we decrease the threshold, we increase the generality of relations. More individual events get merged together in this way, but there might be a risk that rather different types of events are incorrectly grouped into the same cluster. This is roughly equivalent to increasing the inverse purity of clusters at the expense of their purity.
- When we increase the threshold, we decrease the generality of relations. This way, each event in a relation is more strongly associated to each other. But there might be a risk that events which should have been grouped into one cluster are spread over many different tables. This is roughly equivalent to increasing the purity of clusters at the expense of their inverse purity.

[3]Actually, this also depends on the style of newspapers and their cultural backgrounds. For example, the word "campaign," which originally referred to a military operation, is now used for referring to political or commercial activities in most newspapers and rarely used for the original sense. However, as Saussure has pointed out, this association is arbitrary and might be changed in the future. In this research, we assumed that many newspapers and readers roughly agree on the current association.

Note that each cluster in this stage is a set of tuples of Named Entities. These tuples are grouped in such a way that each element of a tuple has a consistent role within the cluster. In other words, they form a *table*. Since a relation is normally presented as a tabular form, each cluster provides the results of Information Extraction for a certain relation. This way, we can perform IE without specifying a scenario.

### 2.2.5 Overall Algorithm - Revisited

Now we present the modified algorithm for using comparable articles:

1. Obtain comparable news articles from multiple news sources.
2. List all the possible NE tuples. *(Relation Detection)*
3. Obtain the features of significant NE tuples.
4. Perform clustering of NE tuples and identify each cluster as a type of relation. *(Relation Identification)*
5. Present the obtained clusters in a tabular form.

At the first step, we use a simple bag-of-words based clustering technique to identify comparable articles from multiple news sources. Note that this first clustering is different from the later clustering to identify a relation. At Step 2., we list all the possible relation instances, i.e. all the NE tuples in each set of comparable articles. To compute the significance of each Named Entity, we first obtain the frequency of each NE among the articles within each set. We use an NE tagger and within-document and cross-document coreference resolver to identify NEs that appear in multiple articles. At Step 3., we obtain top-ranked Named Entities and its corresponding features for each article set. In order to obtain features for each NE, we use a parser and tree regularizer. Then we cluster each tuple of NEs at Step 4. Finally, we obtain a set of relations that are presented as a table.

Features of a relation instance must contain information that helps in identifying its relation type, and be independent from actual events that are filled in the table. For example, the features of an event "`Katrina hit New Orleans`" should not contain any information that implies this particular event, but contain something that implies this is a hurricane event.

We use two different types of features: one is local and the other is global. A "local feature" is a feature that is attributed to each individual Named Entity, whereas a "global feature" is a feature that is attributed to the whole tuple of NEs. In this thesis, we used a regularized expression of phrases that modify each NE of a tuple as a local feature, and a bag-of-words of the article set from which the whole tuple is extracted as a global feature. For two NE tuples to be grouped into the same cluster, they must satisfy a certain level of similarity in both local and global features.

#### Non-hierarchical Clustering

In URD, a certain NE tuple can sometimes represent several relations. Suppose there is the following article:

```
Yankees acquired Alex Rodriguez.  He led the team to victory in the first game.
```

This article contains at least two distinct relations that are likely to grow into large clusters, both of which involve the NE tuple `Yankees` and `Alex Rodriguez`:

1. `Yankees acquired Alex Rodriguez.` (an acquired player and its team)
2. `Alex Rodriguez led Yankees to victory in the first game.` (a winning team and its contributor)

To ensure that we can discover both relations, we allow each NE tuple to belong to multiple clusters. This type of clustering method is called "non-hierarchical clustering." The major drawback of non-hierarchical clustering is that it may produce too many clusters that have only a small difference from others. In our implementation of URD, we tried to merge similar clusters as post processing and took only clusters that are large enough.

# Chapter 3

# Implementation

In this chapter, we present the detailed description of our system components and experimental settings. Since URD requires various types of linguistic knowledge, the actual implementation is divided into more components than the abstract layers that we presented in Section 2.2.2. We first present an overview of our system, and then describe each component in detail.

## 3.1 System Overview

Our implementation has the following components:

### 1. Obtain comparable news articles from multiple news sources

- Web Crawler and HTML Zoner
- Comparable Articles Finder

### 2. List all the possible entity tuples (Relation Detection)

- Named Entity Tagger *
- Within-document Coreference Resolver *
- Cross-document Coreference Resolver

### 3. Obtain the features of significant entity tuples

- Full Parser *
- Tree Regularizer *
- Feature Extractor and Indexer

### 4. Perform clustering of entity tuples

- Clusterer
- Cluster Merger

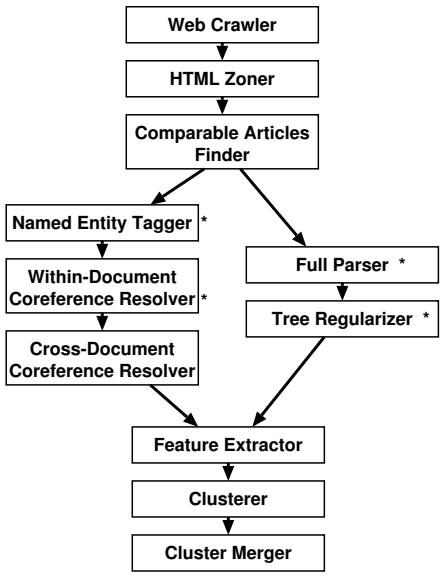


Figure 3.1: System components

A star sign (*) indicates that we used existing software packages. Although most components are used in this order, the actual data flow was split into two ways as shown in Figure 3.1, due to implementation restrictions.

## 3.2 Web Crawling and HTML Zoning

An easy way to obtain comparable news articles is to obtain articles from multiple news sources on the same day. There are many online news sites that publish or update a number of news articles on daily basis, some of which are overlapping in their topics. So we used these articles by crawling news sites every day.

We built an in-house Web crawler and HTML zoner that is suitable to collect a medium-size website and extract its main text. The crawler follows the links in an HTML page recursively from a starting URL and stores the page contents on files. The HTML zoner tries to rip off the HTML tags and redundant texts that are not the part of an article. In this section, we briefly explain the outline of this component.

### 3.2.1 Web Crawling

Our Web crawler is a simple TCP client which follows page links recursively and stores the HTML contents in a compressed format. For our purpose, the crawler implementation does not necessarily have to be efficient. So our crawler handles each page sequentially. It parses the page content and stores the anchor texts of each link separately from the page. Since most news sites have a unique URL for each distinct article, our crawler tries to maintain a set of URLs that have been visited before and avoid to retrieve the same link twice. This way, we can collect 20 news sites each day

in about one hour. The average number of collected pages ranges from 3,000 to 5,000 per day. The crawler also supports cookie handling and gzip compression in HTTP because several sites required it.

### 3.2.2 Layout Analysis

Taking news articles from a number of websites is a more complicated task than it first appears. There are normally extra texts such as ads or navigation links in each page in addition to the article text itself. We have implemented a simple method to remove them by analyzing the layout of each webpage. The idea is that each page consists of fixed parts and varied parts and the article text always appears only in the varied parts. The fixed parts are not changed within a certain website. We used a clustering technique to figure this out automatically.

First, we convert an HTML page structure which is a tree of HTML elements into a sequence of HTML block elements to make it easy to compare. Then we computed the maximum common subsequence of two element sequences. We use the following formula to compute the similarity between two HTML pages, $\mathrm{Sim}(A, B)$ for two sequences of HTML block elements $A = e_{A1}, e_{A2}, ...$ and $B = e_{B1}, e_{B2}, ...$:

$$\mathrm{Sim}(A, B) = \frac{\sum_{e \in MCS(A,B)} W(e)}{\sum_{e \in A} W(e) + \sum_{e \in B} W(e)}$$

where $W(e)$ is the total number of alphabets of all the strings included in the element $e$, and $MCS(A, B)$ is the maximum common subsequence between two element sequences $A$ and $B$. Note that the contents (texts) within these elements are ignored when comparing page layouts: two HTML elements are considered equivalent if they have the same tag and attributes. Same elements that appear consecutively are grouped and treated as a single element. This way, HTML pages that have a similar layout structure to each other gets a higher similarity value, regardless of their contents. Clustering is performed in a hierarchical manner. In our system, we used the similarity threshold as $T = 0.97$.

After clustering all the pages, we try to find out the fixed and varied parts from HTML elements. Each cluster can be considered as a set of pages that have the almost identical layout structure. First, we align all the elements in each sequence within a cluster. Then we number each group of the equivalent elements as $E_i$. Now we calculate the differential score $\mathrm{Diff}(E_i)$ for each group, as in:

$$\mathrm{Diff}(E_i) = \frac{\sum_{s_1, s_2 \in E_i} W(s_1) + W(s_2) - 2 \cdot |MCS(s_1, s_2)|}{\sum_{s_1, s_2 \in E_i} W(s_1) + W(s_2)}$$

We found a portion of a webpage that has $\mathrm{Diff}(E_i) < 0.8$ is likely to be ads, banners or navigation texts, which are not the part of news texts. After removing all these portions we take the remaining parts as the main text of that page. We observed that we can extract the main text accurately from about 90% of all the obtained HTML pages.

### 3.2.3 Preprocessing

After taking the article texts, we performed preprocessing in order to eliminate noise and regulate the surface differences. First we apply a tokenizer and sentence splitter, and then a sentence

trimmer. Sentence trimming is necessary to remove extra words that most NLP components do not expect as input (shown in underline):

- NEW YORK, Oct. 17 --- One of the largest apartment complexes in the nation ...
- WASHINGTON, Feb. 1 (AP) _ Justice Samuel A. Alito Jr. cast his first vote ...

We also removed sentences that are shorter than 10 words. We limit the number of sentences per article to 20 at maximum, since we observed that the most significant event for an article normally appears within the first 20 sentences.

## 3.3 Obtaining Comparable Articles

As we explained in Section 2.2.3, there are two purposes for using a set of comparable articles instead of a single article. One is to weight important Named Entities, and the other is to enrich the feature set for each entity in a relation. After obtaining a number of articles from multiple news sources, we cluster them for finding comparable news articles. Note that this "clustering" is merely to find a set of articles that report the same event on a particular day, and is different from the clustering for relation identification we explained in 2.2. Finding a set of articles that share the same topic has been a well-established task, since Topic Detection and Tracking (TDT) [25]. When collecting comparable articles that report the same event, we can expect that many proper nouns should be overlapped. Since proper nouns normally have a higher IDF (Inverse Document Frequency,) we expected a vector space model using bag-of-words should work well for this purpose.

We eliminate stop words and stem all the other words using a Porter stemmer [18], then compute the similarity between two vectors of articles. In news articles, a sentence that appears in the beginning of an article is usually more important than the others. So we modified a traditional cosine distance so that it preserved the word order to take into account the location of each sentence. A word vector from each article is computed as:

$$V_A(w) = [\mathrm{IDF}(w) \cdot \sum_{i \in POS(w,A)} \exp(-\frac{i}{\text{Avg. words}})]$$

where $V_w(A)$ is a vector element of word $w$ in article $A$ and $POS(w, A)$ is a list of $w$'s positions in the article. Avg. words is the average number of words for all articles. $IDF(w)$ is the inverse document frequency of word $w$, given by:

$$\mathrm{IDF}(w) = -\ln \frac{\text{Number of documents that contain } w}{\text{Total number of documents}}$$

Then we calculated the cosine value of each pair of vectors:

$$\mathrm{Sim}(A_1, A_2) = \cos(V_{A_1} \cdot V_{A_2})$$

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 cluster. The accuracy of these pair-wise links was about 60% in F-score. Although we don't know if this performance is optimal, we can say it is good enough for our purpose by observing the results in later stages.

### 3.3.1 Handling Many Articles Efficiently

When we have to cluster thousands of articles, the number of comparisons of two individual articles could be millions. However the actual number of comparisons gets even larger because some news events kept being continuously reported across several days. Furthermore, the time of reporting a certain event might be varied depending on newspapers. Since we want to avoid computing the similarity for all possible combinations of articles, we have introduced a technique to omit a great deal of the comparisons.

The speedup technique is twofold. Firstly, all the words (elements in a feature vector) are weighted with IDF. Since normally low-weighted words does not contribute much to the cosine distance of two vectors, we can use the high-IDF words in an article to narrow the range of comparisons. We indexed all the content words of an article that have the top-$n$ highest IDF values. When comparing articles, we picked the $n$ top words from each article and look up all the articles which also have any of these words in their top words list. After retrieving these articles, we compute the similarity against them and pick the article which has the highest similarity. For this experiment, we use $n = 7$, which is about 3% of the words included in an average article. We got roughly the same result as using all the words.

Another technique for speedup is incremental clustering. To find events that have kept being reported for a certain duration, we need to compare articles up to several days ago. However, we can also assume that an article which appears too long ago will never get clustered. So we maintain a working set of all the recent clusters and try to grow those clusters by adding newly obtained articles every day. If a certain cluster stop growing for some time, we can regard it as a complete cluster that is not updated anymore. For each cluster in the working set, we computed the average time of the 10 newest articles that have been added to that cluster. If a cluster is not updated for more than two days, we remove it from the working set ("Garbage Collection") and output it as a complete cluster. This way, we can keep the actual number of comparisons as small as possible while obtaining a cluster that spreads temporally.

After taking comparable articles, every sentence in each article set is assigned a Sentence ID and stored on the disk for later retrieval. An example of comparable articles is shown in Figure 3.2.

## 3.4 Named Entity Tagging and Coreference Resolution

After obtaining a set of articles that report the same event, we apply Named Entity Tagging and Coreference Resolution for every article in each set of articles that report the same event. First, we applied an HMM-based NE tagger to each article. This tagger recognizes five different types that are defined in the Automatic Context Extraction (ACE) guidelines: `PERSON`, `ORGANIZATION`, `GPE`, `LOCATION` and `FACILITY` [1]. The performance of this NE tagger is 85% in F-score. The tagger also recognizes a certain kind of noun phrases which can be used for coreference resolution in the later stage. In this system, we actually introduced another pseudo-NE type `NAN` (Not A Name), which is an entity that is not recognized as a name but still recognized as a noun phrase. We assigned

[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/`

```
Article Set ID:
  C200705270801-R200705190801-www.washingtonpost.com-e90b21fce9db498f

Similarity: 0.9491
Article-1:
  Justice-White House face-off

Sentence-1-1:
  A top Justice Department official thought President George W. Bush 's no-warrant
  wiretapping program was so questionable that he refused for a time to reauthorize
  it , leading to a standoff with White House officials at the bedside of the ailing
  attorney general , a Senate panel was told yesterday .
Sentence-1-2:
  Former Deputy Attorney General James Comey told the Senate Judiciary Committee
  that he refused to recertify the program because Attorney General John Ashcroft
  had reservations about its legality just before falling ill with pancreatitis in
  March 2004 .
...

Similarity: 0.9464
Article-2:
  President Intervened in Dispute Over Eavesdropping

Sentence-2-1:
  President Bush intervened in March 2004 to avert a crisis over the National Security
  Agency 's domestic eavesdropping program after Attorney General John Ashcroft ,
  Director Robert S. Mueller III of the F.B.I. and other senior Justice Department
  aides all threatened to resign , a former deputy attorney general testified Tuesday .
Sentence-2-2:
  James B. Comey , an ex-deputy attorney general , testified Tuesday .
...
```

Figure 3.2: One of obtained comparable articles (tokenized and trimmed)

$E = \{e_1, e_2, ..., e_n\}$
while $E$ is not empty {
    $C = \cup_{e \in E} Rep(e)$
    $s = \text{argmax}_{s \in C} Link(s)$
    $X = \{e | s \in Rep(e)\}$
    Take $X$ as a cross-document entity.
    Remove all $e \in X$ from $E$.
}

Figure 3.3: Cross-document coreference resolution procedure

`NAN` type for noun phrases that are not classified to any existing NE type. This type is intended to be used for accommodating all other entities that are not covered in the current ACE definitions.

After NE tagging, we performed coreference resolution for each article. Coreference Resolution is a task to connect the mentions of the same entity that appear among different sentences. For example, mentions "`George W. Bush`", "`he`" and "`the President`" may be mentions that all refer to the same entity. By using coreference resolution, we can link features from many different mentions of an entity, which would have been separated otherwise.

The coreference resolution in our system is divided into two phases: within-document resolution and cross-document resolution. Within-document resolution connects mentions of entities within one article. Cross-document resolution connects mentions from different articles. When a set of articles report the same event, we can expect a lot of entities are shared among them. Therefore, by connecting entities across multiple articles, we can obtain more features per entity. Normally, within-document resolution involves some sort of sentence or context analysis in order to handle pronouns or nominals. In contrast, we used a simple string-match based algorithm for cross-document resolution because there is no contextual information available for individual documents.

For within-document resolution, We used an existing coreference resolution package that achieves 85% in F-score. A cross-document resolution problem can be regarded as a problem to find the best partition of these entities. We first enumerate multiple canonical representations for each entity within a document. For example, if an entity within an article has three mentions "`George W. Bush`", "`he`" and "`the president`", its canonical representations are "`GEORGE_W_BUSH`", "`W_BUSH`", "`BUSH`", and "`PRESIDENT`." All words are capitalized and their preposition and articles are removed. The basic strategy is that we pick the most connected canonical representation first, and group the connected entities as one big cross-document entity. The actual procedure is shown in Figure 3.3. $Rep(e)$ is a set of canonical representations for the entity $e$, and $Link(s)$ is the number of unique entities that have string $s$ in their canonical representations.

### 3.4.1 Weighting Important Named Entities

After finding all the entities and their mentions, we rank them by their importance. We assumed the importance of entities can be decided with the following factors:

- The location of each mention for an entity. A mention that appears earlier in an article is

more important than one that appears later.

- The number of mentions for each entity. An entity that is referred more often is more important.

Based on the above intuitions, we used the following formula to compute $W(X)$, the weight for each cross-document entity $X$, which is a set that consists of one or more within-document entities $X = e_1, e_2, ..., e_n$:

$$W(X) = \exp(C \cdot \frac{\sum_{e \in X} \text{FirstLoc}(e)}{|X|}) \cdot \sum_{e \in X} (1 + \ln \text{Mention}(e))$$

where FirstLoc($e$) is the location where one of the mentions of the entity $e$ first appears in an article, and Mention($e$) is the number of mentions of the entity $e$. In our system, we further assumed that the number of entities involved in a certain event is at most five and used only the top five entities for each event (i.e. article set). This greatly reduces the cost of enumerating all possible NE tuples in the later stage.

To show the relevance of this method, we conducted a quick evaluation. We asked annotators to choose the five most important entities from 20 news events, and measured the overlap of human-selected entities and the entities selected with the above formula. We got 60% F-score. A sample result is shown in Figure 3.4. Most of the errors were due to the original coreference errors.

## 3.5 Feature Extraction

At this point, we can define a set of relation instances (entity tuples) for each event as follows:

$$R = \{(x_1, x_2, ..., x_n) | x_i \in X\}$$

where $X$ is a set of the top five important entities for the event that are obtained in the previous stage. In short, a relation instance is a tuple (i.e. Cartesian product) consisting of any combination of the important entities. Each event generates multiple relation instances, as shown in Figure 3.5. Note that we still don't know what these relations really *are*, because our strategy is "to extract first and then find out what we extracted."

From now on, we need to extract features to identify the relation type for each relation instance. As we explained in 2.2.5, there are two types of features for relation identification: local features and global features. A local feature is a feature that is attributed to each entity in a relation individually. Thanks to the cross-document coreference resolution, we group features obtained for multiple mentions as a set of features that belong to a single entity. A global feature is a feature that is attributed to the whole tuple. Both types of features should be specific for a particular relation type, but *not* be specific for a particular event. In other words, we can use common nouns or verbs that may be used for any other event as features, but *cannot* use Named Entities.

Choosing a good feature set is always a difficult task in most clustering applications. Features must be both specific *and* general, i.e. they have to be specific to capture a certain aspect of things we want to distinguish, but at the same time they have to be general to ignore all other details we don't want to care. However, since we cannot stipulate what we really need in order to distinguish relations, a lot of trial and error and intuitive guess is needed.

```
Article Set ID:
  C200705270801-R200705190801-www.washingtonpost.com-e90b21fce9db498f

Significant Entities:
  341.0409 PER:GONZALES
  156.2985 ORG:JUSTICE_DEPARTMENT
  151.2717 ORG:JUDICIARY_COMMITTEE
  88.4490 PER:COMEY
  83.5786 PER:ASHCROFT

Article ID:
  A200705270801-200705160801-www.newsday.com-5432e67dad2ce694-15

Sentence ID:
  S200705270801-200705160801-www.newsday.com-5432e67dad2ce694-15-0
Sentence (Entity Tagged):
  A <refobj xobjid="NAN:OFFICIAL">top Justice Department official</refobj>
  thought <refobj netype="PERSON" xobjid="PER:GONZALES">President George W.
  Bush</refobj> 's no-warrant wiretapping program was so questionable that
  <refobj netype="PERSON" xobjid="PER:GONZALES">he</refobj> refused for
  <refobj xobjid="NAN:TIME">a time</refobj> to reauthorize
  <refobj xobjid="NAN:TIME">it</refobj> , leading to <refobj xobjid="NAN:STANDOFF">
  a standoff</refobj> with <refobj>White House officials</refobj> at the
  bedside of <refobj>the ailing attorney general</refobj> , <refobj>a Senate
  panel</refobj> was told yesterday .

...
```

Figure 3.4: Article with coreference resolution and NE weighting

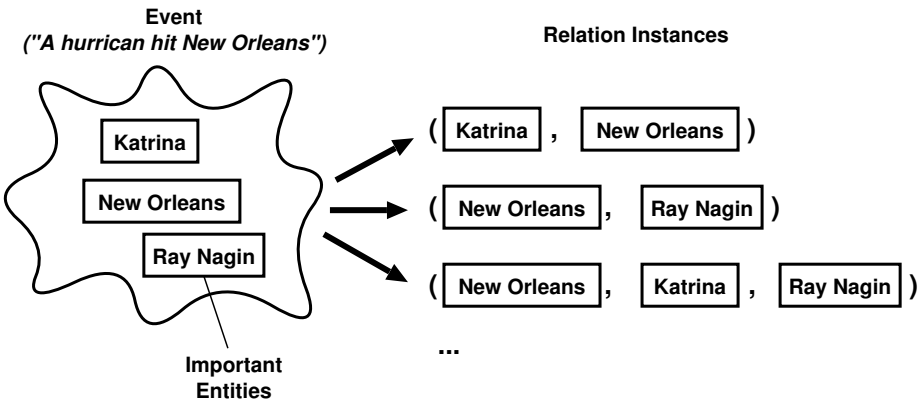


Figure 3.5: Relation instances generated from an event.

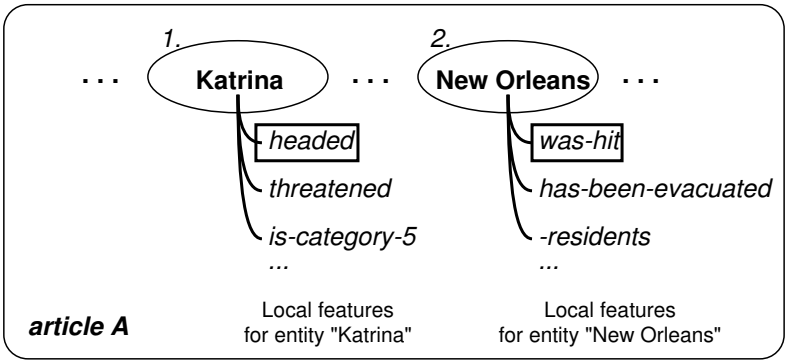


Figure 3.6: Local features

### 3.5.1 Local Features

In our system, we used the expressions taking each entity as arguments as local features of that entity. For example, when there are two entities, "`Katrina`" and "`New Orleans`", involved with a hurricane event, we could take expressions like "`Katrina headed ...`" or "`New Orleans was hit ...`" as shown in Figure 3.6.

Now, the question is *how to capture these expressions?* A trivial way is to take the surrounding words of each entity within a certain window. However, it is known that such simplistic features tend to be affected by extra phrases such as adverbs inserted around an entity. One of the solutions is to use a (shallow or deep) parser in order to trim these extra phrases. In our system, we used a tree regularization schema called GLARF (Grammatical and Logical Argument Representation Framework) [15, 16, 14] as the basic building block of local features.

The general idea of GLARF is to provide extra feature structures on top of traditional constituent parses. GLARF has a number of nice properties that make the features meet our demand. Firstly, it can capture a relationship between a predicate and its logical subject or object. This allows us to treat expressions like "`Katrina hit New Orleans`" and "`New Orleans was hit by Katrina`" in a uniform way, which would be difficult when using a constituent parse only. Sec-

| Label | Description |
|---|---|
| SBJ | Subject |
| OBJ | Object |
| COMP | Complement |
| ADV | Adverb |
| T-POS | Possessive, Number or Quantifier |
| N-POS | Noun Modifier |
| A-POS | Adjective |
| AUX | Auxiliary |
| SUFFIX | Suffix |

Table 3.1: Edge types defined in GLARF (selected)

ondly, it can take care of various linguistic phenomena such as raising, control, parentheticals and coordination. For example, we can obtain the same subject-verb-object relationship for the verb `hit` from the following sentences:

- `Katrina hit New Orleans.`
- `New Orleans was hit by Katrina.`
- `Satellites photographed Katrina hitting New Orleans.`
- `Katrina is expected to hit New Orleans.`
- `Katrina and Rita hit New Orleans.`

For the purpose of Information Extraction, we want to be able to capture the fact that "*Katrina hit New Orleans*" from any of the above sentences. Although we don't discuss the semantic implication introduced by GLARF here, we can say that generalization by GLARF is helpful to make features good enough for relation identification. In GLARF, each sentence is represented as a single connected graph. Each node in a graph corresponds to a single word or multi-word Named Entity. An edge between nodes represents the relationship between the words or phrases. The edge types defined in GLARF are shown in Table 3.1. Figure 3.7 shows a GLARF structure from the sentence "`Katrina hit Louisiana's coast.`"

In GLARF, an expression that modifies a certain node can be represented as a thread of edges starting from that node. In Figure 3.7, for example, there are two entity nodes "`Katrina`" and "`Louisiana`." Therefore the local features from this sentence are "*(Katrina)* `-hit`", "*(Katrina)* `-hit-the-coast`", and "*(Louisiana)* `-'s-coast`." The procedure of feature extraction is shown in Figure 3.8.

The actual implementation of feature extraction is divided into several steps. First, we apply a full constituent-based parser that produces Penn TreeBank outputs [7] to all the sentences in an article set. Then we run a GLARF regularizer against the constituent trees to obtain the regularized structures. Then we merge this output with the result of cross-document coreference resolution we obtained in the previous stage. This process is equivalent to connecting each entity node with a coreferential link, forming one big graph whose NE nodes are interconnected across different sentences (Figure 3.9). Local feature extraction is performed against this graph. The obtained features are stored as a string as shown in Figure 3.10.

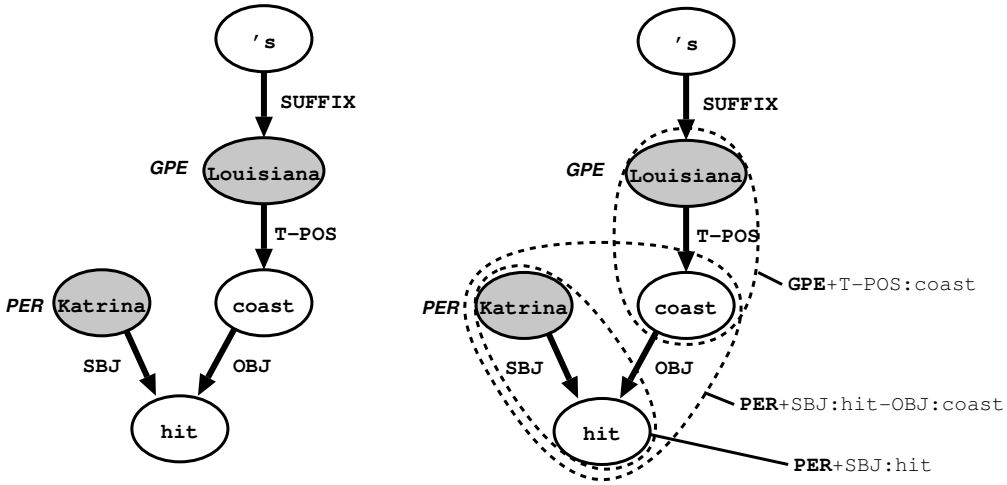


Figure 3.7: GLARF structure and the local features

$G = \{n_1, n_2, ...\}$
for each entity node $n \in G$ {
    $C =$ (empty sequence)
    $p = n$
    while $p$ is not a predicate node {
        Append $p$ at the end of $C$.
        $p \leftarrow$ parent($p$)
    }
    Expand $C$ if the last node has a coordination.
    Add "`NOT`" to the last node if it is connected to any negation node ("`not`," "`never`," "`barely`," "`rarely`," "`hardly`," or "`seldom`").
    Print $C$ as a local feature of $n$.
}

Figure 3.8: Local feature extraction procedure

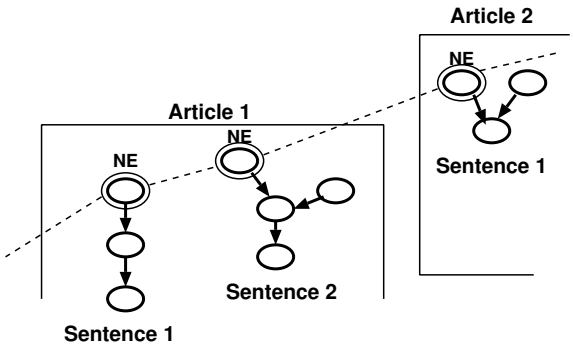


Figure 3.9: NE nodes connected across sentences

### 3.5.2 Global Features

Global features are features that are extracted from every distinct event (i.e. article set) and shared by all the entity tuples that are extracted for the event. Every event generates exactly one global feature set. Global features characterize the overall topic of an event. By using global features, we can distinguish tuples that have exactly same local features such as "`Bloomberg beat Ferrer in New York City`" and "`Sharapova beat Henin in Queens`." Since these sentences appear in rather different topics, we can avoid grouping them into one cluster if the global features from these are very different.

Global features are represented as a bag of stemmed words. We take words that are not the part of an NE and have appeared in at least 30% of all the articles of an article set. In the actual implementation, we took only words that are either verb, noun, adjective or adverb.

## 3.6 Clustering Tuples

After taking all the entity tuples and their features, we finally perform clustering on them using their global and local features. Unlike other types of clustering such as document clustering, we do *not* directly cluster entity tuples for relation identification. Instead, we cluster an object called a "*mapping*" between two entity tuples. In this section, we first illustrate a notion of mapping, and then explain how to construct one and use it in actual clustering algorithm.

### 3.6.1 Finding Mapping

A mapping is defined between two relation instances (i.e. entity tuples) individually taken from different article sets. Since each article set represents a different event, a mapping tells which NE in one event could be replaced with which NE in the other event. When a certain entity from one event shares a lot of its features with the other entity from another event, and those features are *not* specific to these particular events, we can infer that two entities play a similar role in each article set. If we find such correspondence *parallelly* between multiple entities, we are more confident that the tuple of entities from one article set represents a similar relation to the tuple of entities from

```
Article ID:
  A200610180801-200610140801-www.nytimes.com-de1561a6dd63129e-28

Sentence ID:
  S200610180801-200610140801-www.nytimes.com-de1561a6dd63129e-28-0

Sentence:
  The United States pressed for a Saturday vote on a Security Council
  resolution that would impose sanctions on North Korea for its reported
  nuclear test , but questions from China and Russia on Friday evening
  cast the timing and possibly the content of the document into doubt .

Entities:
  GPE:CHINA, GPE:NORTH_KOREA, GPE:UNITED_STATES, GPE:NORTH_KOREA

Local Features:
 ("question from GPE:CHINA cast ...")
  GPE:CHINA @GPE:OBJ:from/IN:COMP:question/N
  GPE:CHINA @GPE:OBJ:from/IN:COMP:question/N:COMP+on/IN
  GPE:CHINA @GPE:OBJ:from/IN:COMP:question/N:SBJ:cast/V
  GPE:CHINA @GPE:OBJ:from/IN:COMP:question/N:SBJ:cast/V:OBJ+timing/N
  GPE:CHINA @GPE:OBJ:from/IN:COMP:question/N:SBJ:cast/V:OBJ+content/N
  GPE:CHINA @GPE:OBJ:from/IN:COMP:question/N:SBJ:cast/V:COMP+into/IN
 ("impose ... for GPE:NORTH_KOREA 's test")
  GPE:NORTH_KOREA @GPE:T-POS:test/N
  GPE:NORTH_KOREA @GPE:T-POS:test/N:OBJ:for/IN:ADV:impose/V
  GPE:NORTH_KOREA @GPE:T-POS:test/N:OBJ:for/IN:ADV:impose/V:OBJ+sanction/N
  GPE:NORTH_KOREA @GPE:T-POS:test/N:OBJ:for/IN:ADV:impose/V:COMP+on/IN
 ("GPE:UNITED_STATES press ...")
  GPE:UNITED_STATES @GPE:SBJ:press/V
  GPE:UNITED_STATES @GPE:SBJ:press/V:COMP+for/IN
  GPE:UNITED_STATES @GPE:SBJ:press/V:COMP+on/IN
 ("impose ... on GPE:NORTH_KOREA")
  GPE:NORTH_KOREA @GPE:OBJ:on/IN:COMP:impose/V
  GPE:NORTH_KOREA @GPE:OBJ:on/IN:COMP:impose/V:OBJ+sanction/N
  GPE:NORTH_KOREA @GPE:OBJ:on/IN:COMP:impose/V:ADV+for/IN

...
```

Figure 3.10: Obtained local features

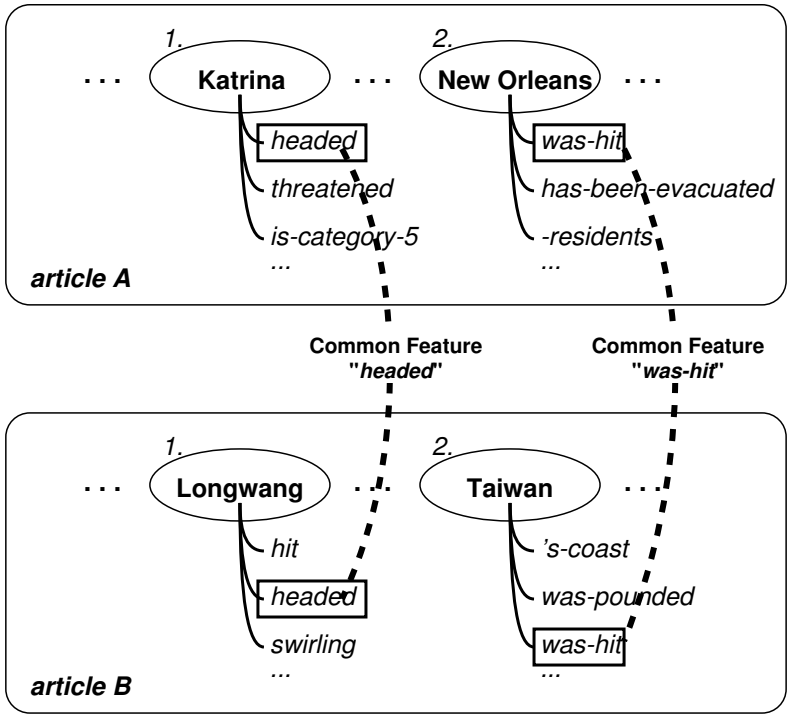


Figure 3.11: Mapping between two entity tuples

another. Figure 3.11 shows a mapping between two distinct events, one from the event of hurricane Katrina, and the other from the event of hurricane (typhoon) Longwang. If each NE from both tuples shares the same local features ("`Katrina`" and "`Longwang`" share the feature "`hit`" and "`New Orleans`" and "`Taiwan`" share the feature "`was-hit`"), we can say that the entities "`Katrina`" and "`Longwang`" play similar roles to each other in both articles, and the entities "`New Orleans`" and "`Taiwan`" do as well. Or we can say that the relation expressed between "`Katrina`" and "`New Orleans`" is similar to the relation expressed between "`Longwang`" and "`Taiwan`".

Mathematically, a mapping between two entity tuples $A$ and $B$ is defined as follows:

$$M(A, B) = \{(A_i, B_j, \text{feature}(A_i) \cap \text{feature}(B_j))|i, j = 1...n\}$$

where $n$ is the number of entities from both tuples, and we try each different permutation of entities from $A$ and $B$. There are $n!$ possible mapper functions for a given $n$. Each pair of entities is associated with the shared local features from both entities, which are basically a set of expressions that modify them. In fact, we don't have to try all $n!$ permutations because we can ignore a mapping that does not have any shared feature. In the actual system, we first indexed all the entity tuples with their local features (strings) so that we can quickly retrieve a set of tuples that share a certain feature. Furthermore, we used only significant features whose frequency is more than 10% of the maximum frequency for each entity. This way, we can improve the speed by cutting off trivial features that are unlikely to contribute to the overall score. We also hope we can exploit the redundancy of multiple sentences to remove erroneous parses. The procedure for finding mappings efficiently for a set of documents $D$ is shown in Figure 3.12.

$D = \{d_1, d_2, ...\}$
for each cluster $d_a \in C$ {
    $F = \{\}$
    for each entity $n_a$ that belongs to $d_a$ {
        for each local feature $p$ that belongs to $n$ {
            Find $(d_b, n_b)$ such that $p$ belongs to $n_b$ that belongs to $d_b$.
            $m = (n_a, n_b)$
            Add $p$ to $F[m]$.
        }
    }
    for every combination $[m_1, m_2...]$ of elements in $F$ {
        $F = \cup_i F[m_i]$
        Print $[m_1, m_2, ...]$ and $F$ as a mapping of two entities and the corresponding local features.
    }
}

Figure 3.12: Finding mappings procedure

As we explained in Section 3.5.1, each entity in a mapping is associated with a set of local features (expressions) that are shared by the entities from both tuples. Since each tuple in a mapping is taken from different events, we can infer that these shared features are preserved even if the actual participant of the event is changed. In other words, by taking the shared features between two independent events, we can eliminate the features that are specific to a particular entity. Suppose we have a strong mapping between two entity tuples, say, ("`Katrina`," "`New Orleans`") and ("`Longwang`," "`Taiwan`"). Now we can imagine there are many expressions that apply to "`New Orleans`" or "`Taiwan`" only. By taking the shared features between both entities, we can filter these expressions and extract the features that characterize this type of the event. The features in a mapping are used to identify the relation type in the later clustering stage.

### 3.6.2 Scoring Mappings

A score is assigned to each mapping object. The score indicates how strong the connection between the two events (entity tuples) is. We used the following metrics for a mapping with given tuples $A, B$ and a mapping function $f$:

$$\text{Score}(A, B, f) = \ln(\text{Sim}(G(A), G(B))) + \sum_i \ln(\text{Sim}(L(A_i), L(B_{f(i)})))$$

where $G(A)$ and $G(B)$ are the global features for the whole tuples $A$ and $B$, and $L(A_i)$ and $L(B_i)$ are the local features for $i$-th entity in $A$ and $B$, respectively. $\text{Sim}(X, Y)$ is a similarity function between two feature vectors $X$ and $Y$. For the similarity function, we used a cosine metric:

$$\text{Sim}(X, Y) = \frac{\sum_{p \in X \cap Y} X_p \cdot Y_p}{\sqrt{\sum_{p \in X} (X_p)^2} \cdot \sqrt{\sum_{p \in Y} (Y_p)^2}}$$

where each element of the vector $X_p$ and $Y_p$ is the product of the frequency and weight of the feature $p$ within each article set.

For the weighting function for a feature, we used the idea of ICF (Inverse Cluster Frequency) which is similar to IDF (Inverse Document Frequency) in traditional document clustering except we take the number of article sets instead of the number of articles. Actually we used a slightly different ICF function for a global feature and local feature. For a global feature, we used a formula that is identical to the original IDF:

$$ICF_{global}(p) = -\ln(\frac{\text{frequency}(p)}{\text{Total}})$$

where frequency($p$) and Total is the frequency of the feature $p$ and the total number of article sets throughout the corpus, respectively. However, for a local feature, we used the following function:

$$ICF_{local}(p) = -\ln(\frac{\text{entfreq}(p) \cdot (\frac{\text{frequency}_{(p)}}{\text{entfreq}_{(p)}})^M}{\text{Total}})$$

where entfreq($p$) is the number of distinct entities for the local feature $p$. This way, we can penalize local features (expressions) that are associated with multiple entities within one article set. For example, if there are two people $A$ and $B$ involved with a certain event, and both entities are associated to expressions like "$A$ `said` ..." and "$B$ `said` ...," then the expression "`said`" is less significant than an expression associated with either one entity.

### 3.6.3 Clustering Mapping Objects

As we explained in Section 2.2.5, we want to allow one entity tuple belong to multiple clusters. For example, an NE tuple ("`Yankees`," "`Alex Rodriguez`") might represent two distinct relations such as player trading and game results. In hierarchical clustering that does not allow overlapping clusters, this is impossible. We achieved this by clustering mappings (pairs of tuples) rather than clustering tuples themselves. The key idea here is that the similarity is computed for pairs of items rather than individual items.

#### Pairwise Clustering

A mapping object has a set of local features that are shared by the entities in both sides. Suppose there are two other entity tuples (`A`, `B`) and (`P`, `Q`) connected with the ("`Yankees`," "`Alex Rodriguez`") tuple. Now, note that the shared features associated with a mapping are different depending on the entities at both ends. It is possible that the mapping between (`A`, `B`) and ("`Yankees`," "`Alex Rodriguez`") has common features like "beat" or "score," whereas the mapping between (`P`, `Q`) and ("`Yankees`," "`Alex Rodriguez`") has common features like "agree" or "acquire." By clustering mapping objects rather than single entity tuples, we can distinguish two distinct relations that contain the same entity tuple. We call this technique "pairwise clustering" because it tries to cluster a pair of objects rather then individual objects. In pairwise clustering, a cluster (i.e. relation) can be formed as a connected graph whose vertices are individual objects (i.e. entity tuples) and edges are object pairs (i.e. a mapping object). Clustering a mapping object

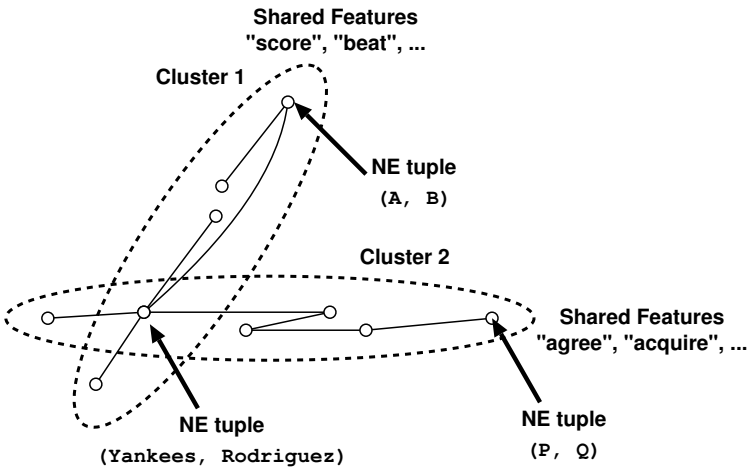


Figure 3.13: Pairwise clustering

is equivalent to clustering an edge between two individual objects (Figure 3.13). This way, while using the hierarchical clustering algorithm, we can still allow an individual object to belong to multiple clusters.

The actual implementation of pairwise clustering in our system is pretty straightforward. We simply treat each mapping object as an item to be clustered. Initial clusters are created from every mapping object. Then we try to grow each cluster by adding another mapping object to the cluster. The clustering procedure for given mappings $M$ is shown in Figure 3.14. The similarity of two clusters is calculated based on the overlapping features from both tuples, just as in the same way as a vector cosine distance of two different vectors that consists from global and local features:

$$\mathrm{Sim}(A, B) = \frac{\sum_{p \in A \cap B} A_p \cdot B_p}{\sqrt{\sum_{p \in A} (A_p)^2} \cdot \sqrt{\sum_{p \in B} (B_p)^2}}$$

where each element of the vector $A_p$ and $B_p$ is the product of the frequency and weight of the feature $p$ within each mapping object.

An alternative view of pairwise clustering is shown in Figure 3.15. Each entity tuple is represented as an extent that covers some features, and the shared features between tuples are represented as the overlapping area. In pairwise clustering, we try to create clusters in such a way that the shared features of mapping objects gets strengthened. This is equivalent to finding a tuple (i.e. circle) that covers the shaded area as much as possible. As long as two shaded area are disjoint to each other, these two clusters (areas shown with dotted lines) get never merged even if both have the same entity tuple.

The obtained clusters contain a set of mapping objects. Since each mapping contains two tuples that have the same number of entities which are associated with each other, we can construct a table from these mappings in a way that each entity was aligned into the corresponding column. Finally, a table contains rows that are entity tuples extracted from different events, and a set of local features that are associated from each column (Figure 3.16).

$M = \{m_1, m_2, ...\}$
$C = \{\}$
for each mapping $m \in M$ {
    $R = \{(m, c) | c \in C$ is involved in $m\}$
    if $R$ is empty {
        Add a new cluster $[m]$ to $C$.
    }
    if $R$ is not empty {
        $m' =$ the mapping that has the highest score.
        Add the mapping $m'$ to the corresponding cluster $c'$.
    }
}
for each mapping $c \in C$ {
    Print the table of all the articles connected with $m \in c$ in such a way that all the corresponding values in $m$ are aligned in the same column.
}

Figure 3.14: Pairwise clustering procedure

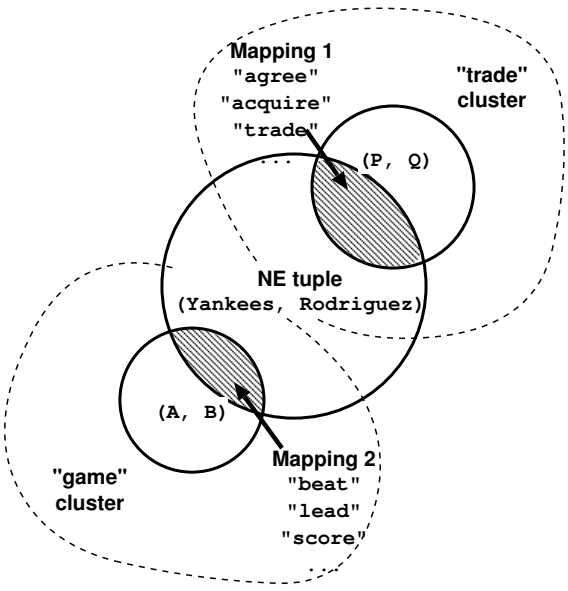


Figure 3.15: Alternative view of pairwise clustering. Two clusters (dotted lines) include the same NE tuple ("`Yankees`", "`Rodriguez`").

```
Cluster ID: 13155

Patterns associated with Column 0:
  @ORG:N-POS:quarterback/N:8 @ORG:T-POS:offense/N:3 @ORG:OBJ:beat/V:3 ...
Patterns associated with Column 1:
  @PER:APPOSITE:quarterback/N:7 @PER:SBJ:lead/V:COMP+to/TO:5 @PER:SBJ:lead/V:5 ...
Patterns associated with Column 2:
  @PER:SBJ:connect/V:COMP+with/IN:5 @PER:SBJ:connect/V:5 @PER:T-POS:pass/N:1 ...

Event ID at Row 0:
  C200510250801-A200510230801-latimes-f2df66d4ed9483356b2851a825b1e6e5-28-13
Entity tuples at Row 0:
  ORG:UCLA PER:MOORE PER:OLSON

Event ID at Row 1:
  C200510270801-A200510250801-abcnews-addbf18637d3e0a6361ae331e2fe4372-24-11
Entity tuples at Row 1:
  ORG:JETS PER:VICK PER:VINNY_TESTAVERDE

Event ID at Row 2:
  C200511030801-A200511010801-abcnews-0e8d2c60608c639881ff596f4f8cf0b4-17-11
Entity tuples at Row 2:
  ORG:PITTSBURGH_STEELERS PER:ROETHLISBERGER PER:WRIGHT

Event ID at Row 2:
  C200511110801-A200511080801-abcnews-ad881d77fb3a1d5bbe85c645bcaa949f-20-19
Entity tuples at Row 2:
  ORG:PATRIOTS PER:PEYTON_MANNING PER:TOM_BRADY

...
```

Figure 3.16: Obtained cluster (relation table)

Table A.

| | hit | 's-coast |
|---|---|---|
| 1. | Katrina | New Orleans |
| 2. | Longwang | Taiwan |
| 3. | Glenda | Australia |

Table B.

| | veer | be-struck |
|---|---|---|
| 2. | Longwang | Taiwan |
| 3. | Glenda | Australia |
| 4. | Xangsane | Vietnam |

Merged Table

| | hit<br>veer | 's-coast<br>be-struck |
|---|---|---|
| 1. | Katrina | New Orleans |
| 2. | Longwang | Taiwan |
| 3. | Glenda | Australia |
| 4. | Xangsane | Vietnam |

Figure 3.17: Merging Tables

## 3.7 Merging Clusters

In the clustering procedure we have presented so far, clusters are only getting grown and never get merged. We developed a separate procedure for cluster merging in order to fully exploit the duality of relation identification: a relation can be identified by its local or global features, but it can also be identified by its entities in each row.

Suppose we have created two distinct tables in the clustering procedure. One table contains the tuples ("Katrina", "New Orleans"), ("Longwang", "Taiwan") and ("Glenda", "Australia"). The other table contains the tuples ("Longwang", "Taiwan"), ("Glenda", "Australia") and ("Xangsane", "Vietnam"). These clusters were separately organized by different feature sets. However, we can still recognize these two clusters are "similar" by comparing its entities (Figure 3.17). [2] This is the basic idea of separated cluster merging.

The actual merging procedure is simple and straightforward. We take all the possible pairs of the existing clusters (i.e. tables) $A$ and $B$, and compute its similarity score $\text{Score}(A, B)$. The similarity score is a combined score of the similarity of local features and entities in a table:

$$\text{RowScore}(A, B) = \frac{|A_{Rows} \cap B_{Rows}|}{\min(|A_{Rows}|, |B_{Rows}|)}$$

$$\text{FeatScore}(A, B) = \min_i \text{Sim}(L(A_i), L(B_i))$$

$$\text{Score}(A, B) =^C \sqrt{\text{RowScore}(A, B)^C + \text{FeatScore}(A, B)^C}$$

where $X_{Rows}$ is the entity tuples contained in the table $X$ and $L(X_i)$ is the set of local features associated with the $i$-th column of the table. We again used a cosine distance as the similarity metric of two feature sets $\text{Sim}(X, Y)$. If the score is above a certain threshold, merge the two tables and their features. In this experiment, we used the threshold 0.7 and $C = 2$. We tried to merge the smallest tables first and gradually merge larger tables. In order to speed up the merging process, we first indexed all the tuples in every table to avoid comparing unrelated tables.

We performed cluster merging only once after building all the clusters in the previous procedure, in order to let each cluster fully develop. Theoretically, cluster developing and cluster merging can be done iteratively in a shorter cycle, just like ordinary bootstrapping approaches. We haven't explored this possibility in this research.

[2] Note that the order of entities in each row matters. For example, a table that contains (A, B) and (C, D) is different from a table that contains (A, B) and (D, C).

## 3.8 Front-end Interface

After obtaining all the relations from corpora, an actual IE task can be converted into a search problem. In an actual IE system, a user still has to rely on some sort of user interface mechanism to search and find relevant tables. In this research, we didn't focus on this issue. However, for the convenience of evaluation, we provided a simple user interface in our system. We assumed that a user might want to search a relevant table with one of the following conditions:

- A string within an Named Entity that appeared in a certain event.
- Global features. This is a keyword such as "`hurricane`," "`murder`," or "`tennis`" that is related to the topic of events.
- Local features. This is an expression such as "`shoot`," "`win`," or "`meet`" that is normally seen in a certain kind of events.

We indexed all the words that appear in every table and built the transpose matrix. Each table $T$ is associated with a list of words $w$ with a certain score:

$$\text{Score}(T, w) = \sum_{p \in G(T) \wedge w \approx p} W(p) + \sum_{p \in L(T) \wedge w \approx p} W(p) + \sum_{p \in E(T) \wedge w \approx p} 1$$

where $G(T)$ and $L(T)$ is the global features and local features contained in the table $T$, and $E(T)$ is a set of words that are included in all the entities in $T$. $W(p)$ is the weight of feature $p$. The expression $x \approx y$ denotes the word $x$ is contained in the string $y$.

When the system receives a user's query, it ranks all the corresponding tables with $\sum_w \text{Score}(T, w)$. After retrieving each table, it presents the table contents along with the actual article texts from which each entity tuple is extracted. The local features and entities are converted into a human-readable form. The entity mentions in the article text are highlighted by different colors according to each column. Each table also shows the top 10 significant global and local features in order to let a user grasp the rough concept of the relation and what the values in each column might represent. Additionally, a user can take a look at actual articles that support the extracted values (Figure 3.18).

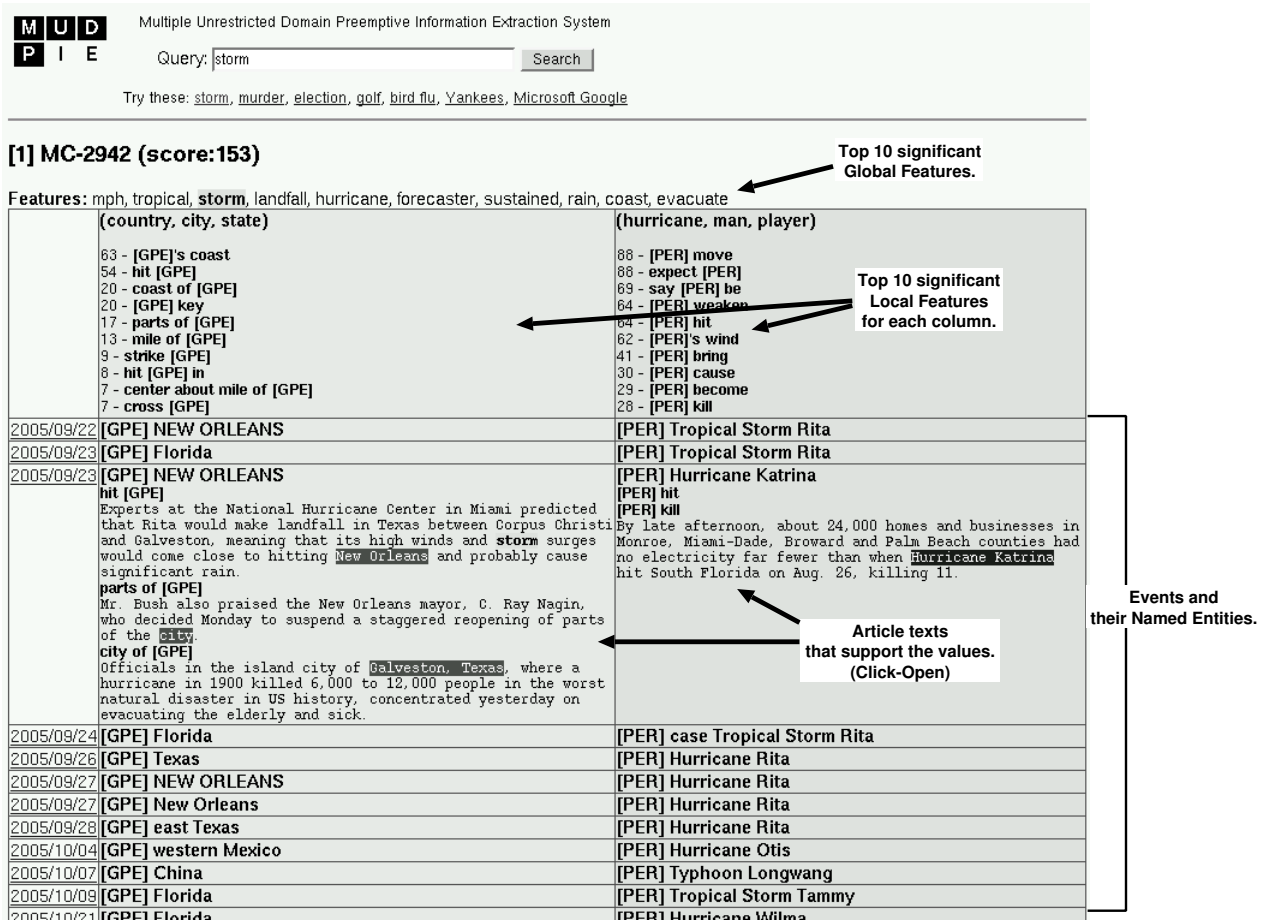


Figure 3.18: User interface screenshot

# Chapter 4

# Experiments

In this chapter, we present the experimental settings and results for evaluating our Preemptive IE system. In the following sections, we first explain how we obtained our training data, and then we describe the evaluation method and its result. In this chapter we mainly discuss the quantitative results. The qualitative aspects of the results are discussed in Chapter 5.

## 4.1 Sources of Relation Discovery

### 4.1.1 News Sources

As we stated in Chapter 2, our system needs a large amount of comparable corpora for input. We obtained thousands of articles daily from the 20 news sites on the Web. Depending on the news site, we successfully extracted article texts from 80 to 100% of the pages we have crawled. The crawling was performed every day from Sep. 2005 to Oct. 2006, and the average time of crawling each day is about an hour. The average number of articles is shown in Table 4.1. After almost one year, we have obtained about 1.3 million pages and 1.1 million articles in total.

### 4.1.2 Feature Sets

After collecting articles, we grouped a set of comparable articles. Examples of comparable articles are shown in Figure 4.1. Actually this step is performed incrementally each day, as we illustrated in Section 3.3.1. After dropping small sets whose number of articles is less than five, we had 35,398 comparable article sets. Since we treat each group of comparable articles as an individual event, we had 35,398 distinct events to be clustered as relations. The numbers of obtained articles and their sentences are shown in Table 4.2.

Next, we applied Named Entity tagging and cross-document coreference resolution in order to find significant entities in each event (i,e. a set of comparable articles). Each event can take up to five significant entities. We obtained 176,985 entities in total. Then we extracted global and local features for each event. Global feature sets are obtained one for each event whereas local feature sets are obtained one for each entity (cf. 3.5).

Originally, we have obtained about 28 million local features in total, 4.4 million different types. From these features, we filtered trivial ones whose frequency is less than a certain threshold. We

| News Site | URL | Avg. Pages Obtained | Avg. Articles Extracted |
|---|---|---|---|
| New York Times | http://www.nytimes.com/ | 552.2 | 488.8 (88%) |
| Newsday | http://www.newsday.com/ | 454.7 | 373.7 (82%) |
| Washington Post | http://www.washingtonpost.com/ | 367.3 | 342.6 (93%) |
| Boston Globe | http://www.boston.com/news/ | 354.9 | 332.9 (93%) |
| ABC News | http://abcnews.go.com/ | 344.4 | 299.7 (87%) |
| BBC | http://www.bbc.co.uk/ | 337.4 | 283.3 (84%) |
| Los Angels Times | http://www.latimes.com/ | 345.5 | 263.2 (76%) |
| Reuters | http://www.reuters.com/ | 206.9 | 188.2 (91%) |
| CBS News | http://www.cbsnews.com/ | 190.1 | 171.8 (90%) |
| Seattle Times | http://seattletimes.nwsource.com/ | 185.4 | 164.4 (89%) |
| NY Daily News | http://www.nydailynews.com/ | 147.4 | 144.3 (98%) |
| International Herald Tribune | http://www.iht.com/ | 126.5 | 125.5 (99%) |
| Channel News Asia | http://www.channelnewsasia.com/ | 126.2 | 119.5 (94%) |
| CNN | http://www.cnn.com/ | 73.9 | 65.3 (89%) |
| Voice of America | http://www.voanews.com/english/ | 62.6 | 58.3 (94%) |
| Independent | http://news.independent.co.uk/ | 58.5 | 58.1 (99%) |
| Financial Times | http://www.ft.com/ | 56.6 | 55.7 (98%) |
| USA Today | http://www.usatoday.com/ | 46.7 | 44.5 (96%) |
| NY1 | http://www.ny1.com/ | 37.1 | 35.7 (95%) |
| 1010 Wins | http://www.1010wins.com/ | 16.1 | 14.3 (88%) |
| Total | - | 4349.2 | 3829.1 (88%) |

Table 4.1: News sites and the average number of articles per day

| | |
|---|---|
| Days crawled | 349 |
| Pages obtained | 1,276,403 |
| Articles obtained | 1,127,124 |
| | |
| Comparable article sets | 154,551 |
| Comparable article sets (size $\geq 5$) | 35,398 |
| Articles grouped | 391,384 |
| Sentences obtained | 4,718,657 |

Table 4.2: Obtained article sets

```
Event ID:
  C200610180801-R200610160801-www.boston.com-ce13007b9c3b4896-14
  (108 articles)

  Article 0: A200610180801-200610140801-www.nytimes.com-de1561a6dd63129e-28
    The United States pressed for a Saturday vote on a Security
    Council resolution that would impose sanctions on North Korea for
    its reported nuclear test, but questions from China and Russia on
    Friday evening cast the timing and possibly the content of the
    document into doubt. ...

  Article 1: A200610180801-200610160801-www.cbsnews.com-1b00c12ef54ab76e-25
    The United States is pressing China to enforce U.N. punishment
    of its ally North Korea ahead of Secretary of State Condoleezza
    Rice's trip to Asia. ...

  Article 2: A200610180801-200610160801-www.iht.com-5dcdae625683e9de-24
    The United States on Sunday pressed China to enforce the United
    Nations's punishment against North Korea and use economic
    leverage to persuade Beijing's communist ally to renounce its
    nuclear weapons program and rejoin international disarmament talks. ...

  ...
```

Figure 4.1: Obtained comparable articles

| | All | Significant |
|---|---|---|
| Events | 35,398 | 35,398 |
| Entities | | |
| (NAN) | 60,515 | 15,402 |
| (PER) | 52,546 | 16,670 |
| (GPE) | 31,114 | 10,161 |
| (ORG) | 28,590 | 9,039 |
| (FAC) | 2,517 | 716 |
| (LOC) | 1,703 | 480 |
| Total | 176,985 | 52,468 |
| | | |
| Global feature sets | 35,398 | 35,398 |
| Local features (token) | 28,544,893 | 2,178,407 |
| Local features (type) | 4,417,109 | 530,571 |

Table 4.3: Obtained entities and features

call the remaining features "significant features." An example of significant features is shown in Figure 4.2. After reducing features, we also dropped orphaned entities that do not have any feature associated. Finally, we had 2.2 million local features associated with 52,468 entities, or about 42 local features (i.e. expressions) for each entity, as shown in Table 4.3.

The whole processing from web crawling to feature extraction and indexing took about 10 hours per day with one standard PC that has 2.4GHz CPU and 4GBytes of memory.

## 4.2 Obtained Clusters

After collecting all the features, we performed event clustering to form relations. As we described in Section 3.6.3, we first obtained a mapping, a pair of NE tuples, to connect distinct events. The number of all possible NE tuples (relation instances) for 35,398 events was about 5 million (although we didn't actually generate them). We have obtained 48,810 mapping objects. A sample mapping object is shown in Figure 4.3.

Then we clustered these mappings by using the associated features in order to find a relevant cluster. Since the first stage does not merge clusters, the number of generated clusters increases monotonically, almost linearly as the mappings are processed (Figure 4.4). We merged the obtained clusters and dropped small ones that include less than four events.

At the end, we had 2,193 clusters. Each cluster forms a table which represents a certain relation between Named Entities. Table (cluster) merging increased the average number of rows in each table, but decreased the total number of tables, as shown in Table 4.4. Table merging does not change the column of each table. Table 4.5 shows the distribution of the table sizes in columns and the top 20 frequent NE combinations that appear in the columns.

Among all the events, only 5,900 events were grouped into a relation that has four or more rows (Table 4.6). This means the only 16% of all the events were actually recognized as a major type of event that happened more than four times a year. An example of an obtained relation is shown in

```
Event ID:
  C200610180801-R200610160801-www.boston.com-ce13007b9c3b4896-14
  (108 articles)

Significant Entities:
  GPE:NORTH_KOREA
  ORG:SECURITY_COUNCIL
  GPE:UNITED_STATES
  GPE:CHINA
  GPE:JAPAN

Significant Global Features:
  nuclear, program, military, weapon, missile, sanction, ...

Significant Local Features:

  for entity GPE:NORTH_KOREA
    @GPE:T-POS:test/N
    @GPE:T-POS:test/N:OBJ:for/IN:ADV:impose/V
    ...

  for entity GPE:UNITED_STATES
    @GPE:SBJ:press/V
    @GPE:SBJ:press/V:COMP+for/IN
    ...

  for entity GPE:CHINA
    @GPE:OBJ:from/IN:COMP:question/N
    @GPE:OBJ:from/IN:COMP:question/N:COMP+on/IN
    @GPE:OBJ:from/IN:COMP:question/N:SBJ:cast/V
    ...
```

Figure 4.2: Obtained features from one event

```
Mapping:
  Event A: C200510150801-A200510100801-abcnews-8ec6afb27d765ca5a5765cad3f6b4f88
  Event B: C200604060801-A200604020801-iht-02d1508ed7a3a7a9
  Score: -7.86

  Entity 1-A: GPE:CHINA
  Entity 1-B: GPE:AUSTRALIA
  Common features:
    @GPE:OBJ:to/TO:COMP:visit/N
    @GPE:SBJ:allow/V
    @GPE:T-POS:market/N
    ...

  Entity 2-A: PER:SNOW
  Entity 2-B: PER:WEN
  Common features:
    @PER:SBJ:speak/V:COMP+at/IN
    @PER:SBJ:visit/V
    @PER:SBJ:arrive/V
    ...
```

Figure 4.3: Obtained mapping that connects NE tuples, (`CHINA, SNOW`) and (`AUSTRALIA, WEN`) each of which comes from distinct events.

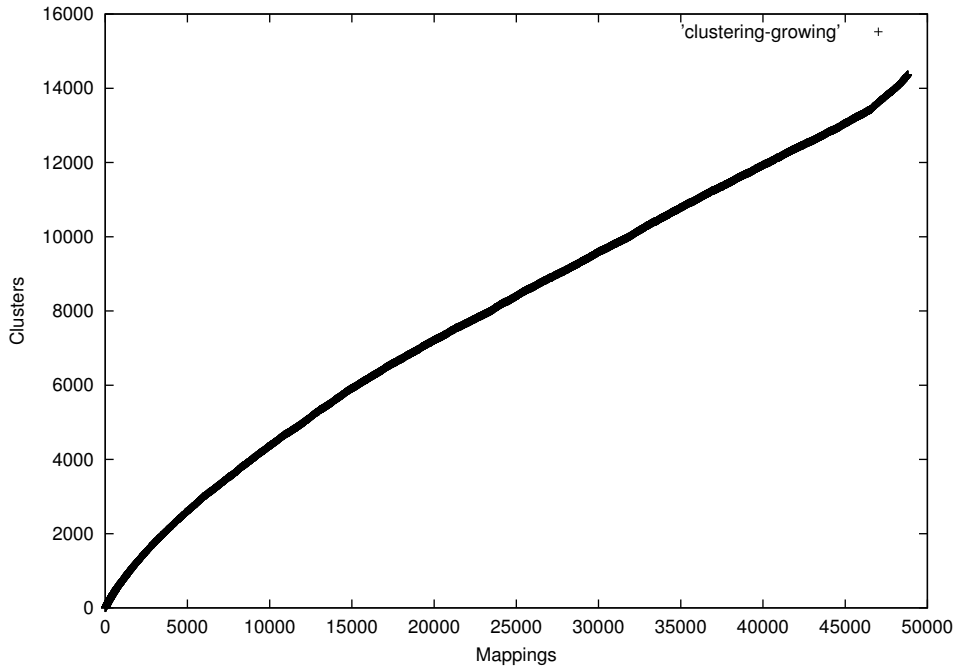


Figure 4.4: Relationship of processed mappings and generated clusters

| Rows | Tables (unmerged) | Tables (merged) |
|---|---|---|
| rows = 4 | 1,084 | 746 |
| rows = 5 | 572 | 386 |
| rows = 6 | 370 | 237 |
| rows = 7 | 246 | 154 |
| rows = 8 | 181 | 122 |
| rows = 9 | 133 | 85 |
| rows = 10...14 | 381 | 203 |
| rows = 15...19 | 167 | 65 |
| rows = 20...29 | 181 | 65 |
| rows = 30...39 | 92 | 45 |
| rows = 40...49 | 52 | 28 |
| rows = 50...99 | 84 | 38 |
| rows = 100...199 | 17 | 15 |
| rows = 200...299 | 2 | 2 |
| rows = 300...399 | 0 | 2 |
| Total | 3,562 | 2,193 |

Table 4.4: Distribution of table sizes before and after merging (rows)

| Columns | Tables | Sample Relation |
|---|---|---|
| columns = 2 | 1,878 | PER was arrested in GPE. |
| columns = 3 | 282 | PER and PER had a talk in GPE. |
| columns = 4 | 33 | PER confronted with PER in ORG of GPE. |

| NE combinations | Tables | Sample Relation |
|---|---|---|
| GPE + PER | 311 | PER visited GPE. |
| PER + PER | 290 | PER and PER got married. |
| ORG + PER | 284 | ORG acquired PER. |
| NAN + PER | 284 | PER was sentenced to NAN. |
| GPE + ORG | 146 | ORG considered to put sanctions on GPE. |
| GPE + GPE | 142 | GPE rejected GPE's demand. |
| NAN + ORG | 138 | ORG reported a death by NAN. |
| GPE + NAN | 92 | GPE's governor opposed NAN. |
| NAN + NAN | 84 | NAN dealt with NAN. |
| ORG + ORG | 75 | ORG beat ORG. |
| ORG + PER + PER | 64 | |
| GPE + ORG + PER | 38 | |
| GPE + PER + PER | 33 | |
| GPE + GPE + PER | 26 | |
| ORG + ORG + PER | 23 | |
| PER + PER + PER | 19 | |
| NAN + ORG + PER | 16 | |
| GPE + GPE + ORG | 14 | |
| FAC + PER | 9 | |
| NAN + PER + PER | 7 | |
| Others | 98 | |
| Total | 2,193 | |

Table 4.5: Distribution of table sizes (columns)

| | |
|---|---|
| All Events | 35,398 |
| Possible NE Tuples | 5,415,894 |
| Mappings generated | 48,810 |
| Clusters generated | 14,362 |
| Clusters after merging (size $\geq 4$) | 2,193 |
| Actual NE Tuples included | 17,301 |
| Events in a cluster | 5,902 |

Table 4.6: Clustering results

Figure 4.5.

## 4.3 Evaluation of Obtained Relations

Finally, it's time to evaluate the obtained relations. As with most other NLP systems, we try to answer the following questions:

1. How many obtained relations are "relevant" from a user's viewpoint?
2. Among all the relevant relations, how many of them were actually obtained?

Let us call these two questions as *Accuracy question* and *Coverage question* respectively, comparing to the precision and recall, the two major metrics commonly used in most applications. The Accuracy question is relatively easy to answer by sampling the obtained relations. However, answering the Coverage question is not easy. Unlike many existing IE systems, we have no established dataset or guideline for experimenting or evaluating the performance of relation discovery. This bears on at least two problems: First, we don't know what kind of relation should be considered as meaningful among others. Second, we need to know the entire set of all the existing relations in the universe to measure how many previously unknown scenarios were discovered. But this is obviously not a feasible task, because there are a virtually infinite number of relations, as people can always conceive a relation among any combination of objects and concepts in their mind.

The above two questions concern only the performance at the *relation* level. In fact, to answer the Accuracy question we also need to measure the performance at the *event* level for each relation, which is to look into the contents in rows and columns of each table. Therefore, the Accuracy question can be further divided into the two sub-questions:

1a. For each relation, how many events in the relation were relevant? *(Event Precision)*
1b. Among all the existing events, how many of them are captured by a relevant relation? *(Event Recall)*

In this section, we try to answer the above two questions. First, we chose several "representative" relations and tried to estimate how many meaningful relations (tables) were obtained. Then we measure the precision and recall at the event level for each relation. Finally, we try to evaluate randomly sampled tables. This way, we might be able to get some insight about what kind of relations our system can potentially discover.

```
Table ID: 2524 (columns=2, rows=201)

Clustered Events:

  Row-0: (Sep. 27, 2005)
    Event ID: C200509270801-A200509240801-abcnews-01a2b33f55ddac8e3ef58eb1a6f5a97f
    Column-1: GPE:WASHINGTON
    Column-2: PER:PRESIDENT_BUSH

  Row-1: (Oct. 15, 2005)
    Event ID: C200510150801-A200510100801-abcnews-8ec6afb27d765ca5a5765cad3f6b4f88
    Column-1: GPE:CHINA
    Column-2: PER:SNOW

  Row-2: (Oct. 15, 2005)
    Event ID: C200510150801-A200510120801-usatoday-0dbff391829441ec573d6f04b906aacd
    Column-1: GPE:KABUL
    Column-2: PER:RICE
  ...

Associated Local Features:

  for Column-1:
    @GPE:OBJ:to/TO:COMP:visit/N (356)    # visit to GPE
    @GPE:OBJ:in/IN:COMP:arrive/V (183)   # arrive in GPE
    @GPE:OBJ:visit/V (178)               # visited GPE
    ...

  for Column-2:
    @PER:T-POS:visit/N (740)             # PER's visit
    @PER:T-POS:visit/N:COMP+to/TO (529)  # PER's visit to
    @PER:SBJ:visit/V (522)               # PER visited
    ...
```

Figure 4.5: Obtained relation that shows a person PER's visit to a GPE.

### 4.3.1 ACE Event Corpus

In this evaluation, we used ACE (Automatic Context Extraction) 2005 Event Corpus for a test set. ACE 2005 Event Corpus is a manually annotated corpus which consists of 332 English articles from several newspapers[1]. The annotation includes entities and their mentions, relations[2], and events and their arguments [17]. Originally, there are 2,104 events in this corpus. Each event is annotated with its main type, subtype, modality, polarity, genericity and tense. In ACE, the notion we called a "relation" in this thesis is close to a set of ACE events that are grouped with a particular event type. We tailored this corpus by taking only events that are tagged as `Asserted` modality and `Specific` genericity in order to limit the events to only specific ones which actually happened. After this, we had 1,471 remaining events.

We used the definition of event types in the ACE annotation guideline for evaluating the representative relations discovered by our system. Of course, the corpus has a limited number of predefined event types, and it does not cover all the possible events and relations. However, we can consider discovering these event types as a minimal requirement for our system. Since our system tried to discover as many different relations as possible, we can expect these relation types to be discovered automatically if a sufficient number of source articles is provided. There are 8 main types and 33 subtypes that are defined in the annotation guideline [12]. The event types and their frequencies are listed in Table 4.7.

The evaluation goes as follows. First, we created a list of keywords to pick the relations that are likely to contain the event types defined in the ACE Corpus. We used the simple search interface that we described in Section 3.8 to obtain the related tables for each keyword. We first tried to see if the retrieved tables contain a relevant relation for a particular ACE event type to measure the performance at relation level. Then we counted the correctly clustered events (rows) included in each table, in order to measure the precision and the recall at event level.

In order to facilitate the evaluation, we have created a simple user interface (Figure 4.6). An evaluator can see each row and column of a table and its original contexts where the entity tuples were extracted. It also allows an evaluator to annotate each table with a description of that table.

### 4.3.2 Evaluation of ACE-like Relations

We evaluated a set of ACE-like relations chosen by keywords. For each retrieved table, an evaluator first takes a look at all the rows and its contexts and determines whether the table is relevant for the given keyword. If the table includes more than 10 rows (events), only ten rows that are sampled randomly are presented. Figure 4.7 shows an example of a table presented to an evaluator. The mentions of each entity are highlighted to make it easy for a user to find the involved entities. If more than half of the rows (events) were relevant for the ACE event type for which the table was retrieved, the table was regarded relevant. In order to avoid the bias of the search facility, which we did not focus on tuning in this research, we took the top three tables for each keyword. We considered a keyword retrieval was "successful" if any of the three tables is considered relevant.

[1] We used `nw` (news wire) and `bn` (broadcast news) section of the corpus. Each section includes articles from several news sources.

[2] In ACE, the word "relation" has a different meaning. An ACE relation is a tuple of two entities that represents some predefined relationship.

| Event Type/Subtype | Frequency |
|---|---|
| Business/Declare-Bankruptcy | 7 |
| Business/End-Org | 9 |
| Business/Merge-Org | 1 |
| Business/Start-Org | 21 |
| Conflict/Attack | 339 |
| Conflict/Demonstrate | 38 |
| Contact/Meet | 116 |
| Contact/Phone-Write | 29 |
| Justice/Acquit | 3 |
| Justice/Appeal | 22 |
| Justice/Arrest-Jail | 47 |
| Justice/Charge-Indict | 63 |
| Justice/Convict | 37 |
| Justice/Execute | 5 |
| Justice/Extradite | 1 |
| Justice/Fine | 10 |
| Justice/Pardon | 1 |
| Justice/Release-Parole | 10 |
| Justice/Sentence | 38 |
| Justice/Sue | 10 |
| Justice/Trial-Hearing | 33 |
| Life/Be-Born | 10 |
| Life/Die | 176 |
| Life/Divorce | 8 |
| Life/Injure | 55 |
| Life/Marry | 17 |
| Movement/Transport | 214 |
| Personnel/Elect | 16 |
| Personnel/End-Position | 58 |
| Personnel/Nominate | 2 |
| Personnel/Start-Position | 33 |
| Transaction/Transfer-Money | 19 |
| Transaction/Transfer-Ownership | 23 |
| Total | 1,471 |

Table 4.7: Event types and frequencies in ACE 2005 Events Corpus (Asserted and Specific events)

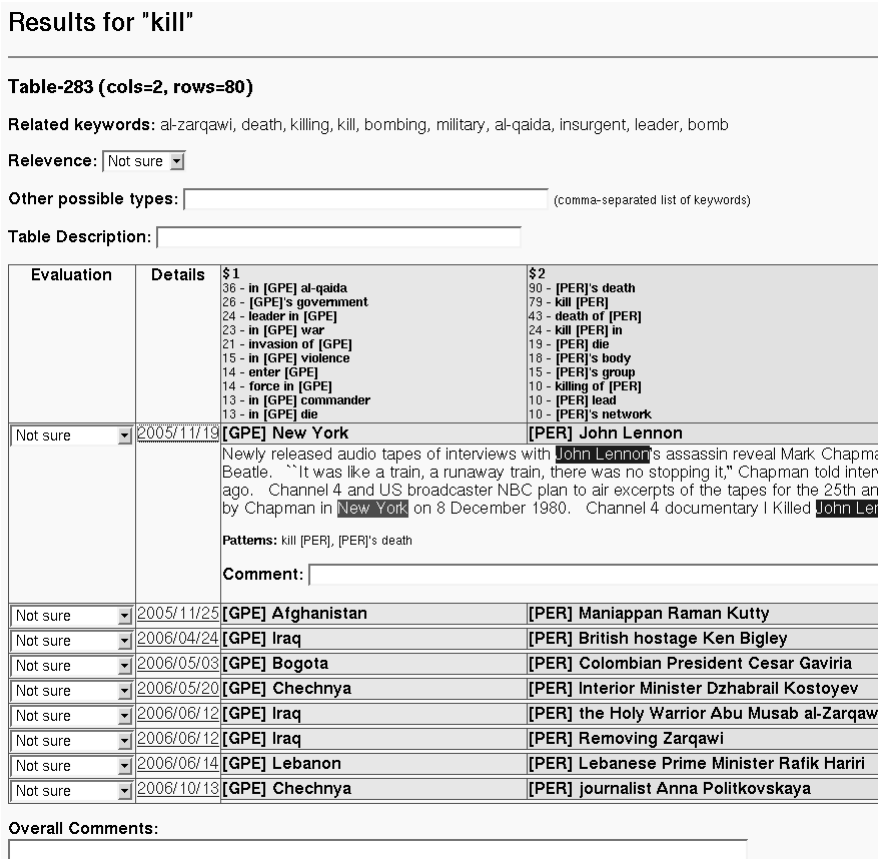


Figure 4.6: Screenshot of Evaluation System

**ACE Event Type:** Life/Die

| PER: Rosa Parks | GPE: Detroit |
|---|---|

... Rosa Parks' body has returned to the city she called home, with thousands waiting in a line more than a quarter-mile long to pay their final respects to the late civil rights leader. Parks was 92 when she died Oct. 24 in Detroit. ...

| PER: Alida Valli | GPE: Italy |
|---|---|

... Alida Valli, one of Italy's great actresses who co-starred in the 1949 film "The Third Man" and Alfred Hitchcock's "The Paradine Case," died Saturday in Rome, the mayor's office said. ...

| PER: Daniel Enchautegui | GPE: Bronx |
|---|---|

... Daniel Enchautegui wasn't just a kind officer, a gentleman and a churchgoing son. He had big dreams for his career in the police department. "He always wanted to be a big boss in the police department one day; that was his dream," the cop's father, Pedro Enchautegui, said outside his Bronx home Sunday. These were some of the thoughts that friends, police buddies and neighbors recalled Sunday about Enchautegui, 28, whose dreams were derailed when he tried to stop a burglary and was killed outside his home in the Bronx on Saturday.

...

Figure 4.7: An example of a table presented to an evaluator.

In order to see if the system can discover other types of relations in addition to ACE relations, we have added the following extra event definitions and tried the corresponding keywords:

- Baseball-Result (keyword: `baseball`) : Results of baseball games. At least one table should include the name of winning and losing teams.
- Golf-Result (keyword: `golf`) : Results of golf tournaments. At least one table should include the name of winning player.
- Earthquake (keyword: `earthquake`) : List of places affected by earthquakes.
- Hurricane (keyword: `hurricane`) : List of places affected by hurricanes.

Among 40 keywords, 34 of them returned at least one relevant table. The list of event types and associated keywords is shown in Table 4.8. Several event types (Business/Start-Org, Business/End-Org, Justice/Appeal, Justice/Pardon, Justice/Release-Parole and Baseball-Result) weren't successfully obtained. We think the most of these events were simply too infrequent to get a large cluster. However, as for the Baseball-Result events, it turned out that we used an irrelevant query keyword for retrieving tables. Actually, if we look into all the tables, there is a table about baseball results, although it did not appear among the top three tables obtained for the keyword "`baseball`," which returns the news about trading of baseball players, but not the game results.

| Event Type/Subtype | Keyword | Relevant? |
|---|---|---|
| Business/Declare-Bankruptcy | `bankruptcy` | Yes (1/3) |
| Business/End-Org | `shutdown` | No (0/3) |
| Business/Merge-Org | `merger` | Yes (2/3) |
| Business/Start-Org | `launch` | No (0/3) |
| Conflict/Attack | `attack` | Yes (2/3) |
| | `bombing` | Yes (3/3) |
| Conflict/Demonstrate | `demonstration` | Yes (2/3) |
| Contact/Meet | `meet` | Yes (2/3) |
| Contact/Phone-Write | `phone` | Yes (1/3) |
| Justice/Acquit | `acquit` | Yes (2/3) |
| Justice/Appeal | `appeal` | No (0/3) |
| Justice/Arrest-Jail | `arrest` | Yes (1/3) |
| Justice/Charge-Indict | `indict` | Yes (2/3) |
| Justice/Convict | `convict` | Yes (3/3) |
| Justice/Execute | `execute` | Yes (2/3) |
| Justice/Extradite | `extradite` | Yes (2/3) |
| Justice/Fine | `fine` | Yes (2/3) |
| Justice/Pardon | `pardon` | No (0/0) |
| Justice/Release-Parole | `parole` | No (0/3) |
| Justice/Sue | `lawsuit` | Yes (3/3) |
| Justice/Sentence | `sentence` | Yes (2/3) |
| Justice/Trial-Hearing | `testify` | Yes (2/3) |
| Life/Be-born | `birth` | Yes (2/3) |
| Life/Marry | `marriage` | Yes (3/3) |
| Life/Divorce | `divorce` | Yes (3/3) |
| Life/Injure | `injure` | Yes (3/3) |
| Life/Die | `death` | Yes (3/3) |
| | `murder` | Yes (2/3) |
| | `kill` | Yes (3/3) |
| Movement/Transport | `trip` | Yes (2/3) |
| Personnel/Elect | `election` | Yes (1/3) |
| Personnel/End-Position | `resign` | Yes (2/3) |
| Personnel/Nominate | `nomination` | Yes (3/3) |
| Personnel/Start-Position | `appointment` | Yes (3/3) |
| Transaction/Transfer-Ownership | `buy` | Yes (3/3) |
| Transaction/Transfer-Money | `pay` | Yes (2/3) |
| (Baseball-Result) | `baseball` | No (0/3) |
| (Golf-Result) | `golf` | Yes (1/3) |
| (Earthquake) | `earthquake` | Yes (3/3) |
| (Hurricane) | `hurricane` | Yes (2/3) |
| Total | | 34/40 keywords |

Table 4.8: Keywords used to retrieve ACE-like relations. Parenthesized types were not defined in ACE.

### 4.3.3 Measuring Event Precision

Next, we measured the precision at event (row) level for 20 tables that we considered relevant. First, the evaluator describes the relation of each table in a natural language sentence. In this description, the evaluator should use a variable such as `$1` or `$2` to refer to a column in the table (for example, "`$1 killed $2`"). The evaluator has to try to make the description as specific as possible, and use all the variables referring to a name. Finally, the evaluator determines whether the values in each row are correct in terms of the relation between these entities. Each row has to include an NE tuple of correct entities involved with the event, in which each entity is consistent with the role of each column. In order to confirm the values are correct along with their contexts, the evaluator needs to take a look at an excerpt of the article texts, with each involved entity highlighted with colors. The evaluator also can leave optional comments and other related keywords to the table.

For evaluating each row, an evaluator can choose his/her decision from four options: "`Correct`," "`Wrong relation`," "`Wrong value`," and "`Not sure`." The evaluator needs to follow the following guidelines to make a choice:

- If the event described in the text is related to the topic of this relation, and the values (entity names) in the columns are correct according to the text, choose "`Correct`." In this case, the entities in each column have to play a consistent role with the column in this event.
- If the event described in the text is clearly unrelated to the topic of this relation, choose "`Wrong relation`."
- If the event described in the text is related to the topic of this relation, but the values in the column are wrong, or not sure, choose "`Wrong value`."
- If the event described in the text is marginally related to the topic, but the evaluator is not sure if the values are correct due to the lack of information, choose "`Not sure`."

For each table, we have collected all the decisions for each row and the description of the relation, and counted the number of these choices. The evaluation results and descriptions of each table are shown in Table 4.9. 105 rows out of 161 rows (65%) were considered correct.

### 4.3.4 Measuring Event Recall

After measuring the event level precision, we tried to measure this event level recall. In order to measure this, we need to know the number of all the relevant events for each event type throughout the entire test set. We used the ACE corpus to measure this, assuming that all the relevant events for those 33 event types are properly annotated. We performed feature extraction against the texts in the corpus, and tried to measure how many of these events get clustered with other events that were obtained from the news sites in advance.

However, there are at least two complications. Firstly, some of the events in the corpus can never be identified by our system. In the ACE corpus, the entities involved with a certain event are called "arguments." The number of the arguments for each event might vary. For example, the mere phrase "`World War II`" can be annotated as a specific Attack event with no argument, whereas a sentence like "`John killed Fred.`" can be annotated as another Attack event with two arguments, the Attacker ("`John`") and its Target ("`Fred`"). Since our Preemptive IE system

| Table name (Optional keywords) | Correct | Wrong Rel. | Wrong Val. | Not sure |
|---|---|---|---|---|
| Keyword: `attack-1` | 3 | 2 | 3 | 2 |
| Description: `ORG $2 performed a military operation in GPE $1.` | | | | |
| Keyword: `attack-2` (`bombing`) | 4 | 4 | 0 | 1 |
| Description: `there was a bombing attack in GPE $2.` | | | | |
| Keyword: `birth-1` | 3 | 1 | 1 | 0 |
| Description: `a baby was named PER $2.` | | | | |
| Keyword: `birth-2` | 7 | 1 | 0 | 0 |
| Description: `PER $2 gave a birth.` | | | | |
| Keyword: `death-1` | 7 | 2 | 1 | 0 |
| Description: `PER $2 died in GPE $1.` | | | | |
| Keyword: `death-2` (`murder`) | 3 | 0 | 4 | 1 |
| Description: `PER $1 died and PER $2 was involved.` | | | | |
| Keyword: `death-3` (`murder`) | 8 | 1 | 0 | 1 |
| Description: `PER $2 was killed probably in GPE $1.` | | | | |
| Keyword: `divorce-1` | 3 | 0 | 0 | 1 |
| Description: `PER $2 decided to divorce.` | | | | |
| Keyword: `divorce-2` (`lawsuit`) | 4 | 0 | 0 | 0 |
| Description: `PER $1 and PER $2 got divorced through lawyers.` | | | | |
| Keyword: `divorce-3` | 4 | 1 | 1 | 0 |
| Description: `PER $1 and PER $2 got divorced.` | | | | |
| Keyword: `kill-1` (`shooting`) | 6 | 0 | 3 | 1 |
| Description: `PER $2 was killed in GPE $1.` | | | | |
| Keyword: `kill-2` | 7 | 0 | 2 | 0 |
| Description: `PER $2 was killed in GPE $1.` | | | | |
| Comment: This table should have been merged with the other table. | | | | |
| Keyword: `kill-3` | 8 | 0 | 0 | 0 |
| Description: `PER $2 was killed in GPE $1.` | | | | |
| Comment: This table should have been merged with the other table. | | | | |
| Keyword: `marriage-1` | 5 | 2 | 0 | 0 |
| Other possible keywords: `wedding` | | | | |
| Description: `PER $2 planned to hold a wedding in GPE $1.` (might be same-sex marriage) | | | | |
| Keyword: `marriage-2` (`divorce`, `wedding`) | 8 | 0 | 1 | 1 |
| Description: `PER $1 and PER $2 was married at some point.` | | | | |
| Keyword: `marriage-3` | 5 | 0 | 1 | 0 |
| Description: `PER $1 and PER $2 got married.` | | | | |
| Keyword: `murder-1` | 4 | 1 | 2 | 1 |
| Description: `PER $2 was probably killed.` | | | | |
| Keyword: `murder-2` | 6 | 1 | 3 | 0 |
| Description: `PER $1 killed PER $2.` | | | | |
| Keyword: `trip-1` | 6 | 1 | 2 | 1 |
| Description: `PER $2 traveled to meet PER $1.` | | | | |
| Keyword: `trip-2` | 4 | 2 | 1 | 2 |
| Description: `PER $1 visited GPE $2.` | | | | |
| Total | 52 105 | 19 | 25 | 12 |

Table 4.9: Evaluation of 20 tables (by keywords)

```
Article ID:
  CNNHL_ENG_20030312_150218.13

Sentence:
  ... the 300th person executed in the state since 1982, when texas
      resumed capital punishment ...
  Relation instance: (Justice/Execute)
    Arguments:
      Agent: GPE:TEXAS ("the state")
      Person: PER:DELMA_BANKS ("the 300th person")
      Place: GPE:TEXAS ("the state")

    Split1:
      Agent: GPE:TEXAS ("the state")
      Person: PER:DELMA_BANKS ("the 300th person")

    Split2:
      Agent: GPE:TEXAS ("the state")
      Place: GPE:TEXAS ("the state")

    Split3:
      Person: PER:DELMA_BANKS ("the 300th person")
      Place: GPE:TEXAS ("the state")

...
```

Figure 4.8: Example of Ace Event Corpus (after split)

assumes that each relation instance in a cluster has the same number of arguments, some of the event annotations can never get clustered. To fix this issue, we converted the event annotations so that each relation instance has always exactly two arguments. First, we removed all the event annotations that have less than two arguments, and then we split annotations that have three or more arguments. For example, the event annotation of "`John killed Fred in New York`" has three arguments (`Attacker`, `Target` and `Place`). Instead of using this ternary tuple as a single relation instance `(John, Fred, New York)`, we split this into three two-argument relation instances: `(John, Fred)`, `(John, New York)` and `(Fred, New York)`. This way, we can expect all the events to be compatible with existing two-column tables. Furthermore, we removed an event argument which does not have a name mention, and then removed the event mention if the number of its arguments is less than two. After this processing, we had 529 relation instances that can be used for this evaluation. The examples of the split ACE annotations are shown in Figure 4.8.

The second problem of using the ACE corpus is that the number of the articles in the corpus is considerably smaller compared to the articles we used for relation discovery. Since we use

| | | |
|---|---|---|
| Articles used | | 332 |
| Sentences included | | 3,278 |
| Relation instances | | 529 |
| | | |
| Events | | 332 |
| Entities | (Perfect) | (System) |
| (GPE) | 1,329 | 1,063 |
| (PER) | 1,060 | 904 |
| (ORG) | 760 | 545 |
| (LOC) | 107 | 116 |
| (FAC) | 102 | 93 |
| Total | 3,358 | 2,721 |
| | | |
| Local features (token) | 24,838 | 22,179 |
| Local features (type) | 19,326 | 17,473 |

Table 4.10: Features extracted from the ACE Corpus text

comparable articles to obtain features, we want to extract as varied features as possible for each event. However, in the ACE corpus, the number of event mentions is normally very small. This seriously decreases the number of features for event clustering, which makes it extremely difficult to find the overlapping features that are crucial to put a relation instance into clusters. This is especially problematic for local features because the number of local features is roughly proportional to the size of the text that we can use. In fact, the average number of local features extracted from each entity was 161 when we use a set of comparable articles from the Web, whereas it was only 7 from the ACE corpus, as shown in Table 4.10.

To mitigate this problem, we slightly modified our clustering algorithm. Originally, the overlapping features were computed between the sets of local features from two relation instances. But in this case we compute the overlapping local features between the set of local features from an ACE relation instance and the set of local features associated with each column of an entire relation (table). For example, if an obtained relation has two columns that are associated with the two local features "`SBJ:kill`, `SBJ:murder`," and "`OBJ:kill`, `OBJ:murder`" respectively, we allow another two-argument ACE relation instance which is associated with the features "`SBJ:kill`" and "`OBJ:murder`" to be clustered into this relation. This way, we can virtually multiply the number of available local features for matching.

We have conducted two experiments. Since in the ACE corpus every entity (and its all mentions) is also annotated, we first used these annotations as cross-document entities instead of using the system NE tagger and coreference resolver. We got 225 instances out of 529 instances (43% recall) clustered. However, when using the system-generated entities, the result got much worse and we had only 58 instances in total (11% recall). The individual results for each event type are shown in Table 4.11. Note that, in this experiment, we only measured how many relation instances from the ACE corpus were grouped into *any* existing cluster that has been obtained from the training data, and we assumed they are always clustered into the correct relation. Therefore, these numbers

merely indicate the maximum recall of the current system and the actual recall might be lower. Also, the distribution of the event types in the corpus is not necessarily proportional to the actual distribution of the event types in training data.

The recall with the system inputs was unexpectedly low. We think this is probably due to the amount of available text for each event. Since the number of local features for each entity was already small, and in order to cluster a relation instance we need to have at least two features correctly extracted, missing only one entity in a source text can result in the great reduction in the number of available features, harming the performance critically.

### 4.3.5 Evaluation of Random Relations

Finally, we tried to evaluate random tables with various sizes. We picked 20 tables, and conducted the same evaluation as we did for ACE-like relation evaluation. 15 tables out of 20 tables were representing some meaningful relation, and among those meaningful tables 51 rows out of 74 rows (69%) were correct. The evaluation results and descriptions of each table are shown in Table 4.12. This result can also be considered as a rough estimation of the relation-level precision, if we expand the range of possible relations to *any* meaningful relation, not limiting to ACE-like relations.

## 4.4 Error Analysis and Possible Improvements

In this section, we look into the detailed causes of the errors and discuss their possible solutions in future. Ultimately, every error in the clustering results can be attributed to wrong features in the inputs. However, since our system consists of a pipeline with several layers of processing, each stage can contribute to wrong features:

- Comparable Article Finding
- Local features (Parsing and Regularization)
- Coreference Resolution
- Finding Mapping

In the previous section, we divided the errors into two types: "Wrong Relation (the relation instance was incorrectly clustered)" or "Wrong Value (the relation was correct, but its values were wrong)." We reviewed the clusters that we used in Section 4.3.3. We looked into the local features used for clustering 10 relation instances for each type of error. Then we tried to find the cause of each error and spot the stage where they were introduced. We found that there are at least six categories of causes, some of which are combinatorial:

a. The feature is ill-formed due to an incorrect parse (PARSE). Since we rely on a constituent parser and a tree regularizer to extract local features, an error in these stages results in an incorrect feature that is associated with a wrong expression. For example, in some cases the word "`birth`" in the expression "`give birth`" was parsed as a direct object, whereas in other cases it was parsed as an indirect object, bearing a different local feature. This type of error can be reduced by improving the parser or tree regularizer.

| Event type/subtype | All | Obtained (Perfect NE) | Obtained (System NE) |
|---|---|---|---|
| Business-Declare-Bankruptcy | 1 | 0 | 0 |
| Business-Start-Org | 10 | 8 | 2 |
| Conflict-Attack | 90 | 30 | 7 |
| Conflict-Demonstrate | 3 | 2 | 2 |
| Contact-Meet | 55 | 14 | 5 |
| Contact-Phone-Write | 3 | 0 | 0 |
| Justice-Acquit | 2 | 0 | 0 |
| Justice-Appeal | 11 | 3 | 1 |
| Justice-Arrest-Jail | 5 | 3 | 1 |
| Justice-Charge-Indict | 13 | 2 | 3 |
| Justice-Convict | 11 | 6 | 2 |
| Justice-Execute | 3 | 2 | 0 |
| Justice-Extradite | 2 | 0 | 1 |
| Justice-Fine | 2 | 2 | 0 |
| Justice-Pardon | 3 | 1 | 0 |
| Justice-Sentence | 6 | 3 | 1 |
| Justice-Sue | 4 | 4 | 1 |
| Justice-Trial-Hearing | 8 | 3 | 1 |
| Life-Be-Born | 2 | 0 | 0 |
| Life-Die | 22 | 11 | 2 |
| Life-Divorce | 2 | 0 | 0 |
| Life-Injure | 3 | 2 | 0 |
| Life-Marry | 2 | 0 | 0 |
| Movement-Transport | 148 | 55 | 11 |
| Personnel-Elect | 8 | 5 | 0 |
| Personnel-End-Position | 37 | 21 | 7 |
| Personnel-Start-Position | 29 | 17 | 3 |
| Transaction-Transfer-Money | 17 | 15 | 5 |
| Transaction-Transfer-Ownership | 27 | 16 | 3 |
| Total | 529 | 225 | 58 |

Table 4.11: Recall for the ACE corpus (for each event type)

| Table Size and Description | Correct | Wrong Rel. | Wrong Val. | Not sure |
|---|---|---|---|---|
| Size: rows=46, columns=GPE+ORG+PER<br>Description: (undetermined) | - | - | - | - |
| Size: rows=32, columns=GPE+ORG+PER<br>Description: ORG $2 won a game in GPE $1 with PER $3's contribution. | 4 | 0 | 4 | 2 |
| Size: rows=10, columns=GPE+PER<br>Description: Famous person PER $2 died in GPE $1. | 6 | 2 | 0 | 1 |
| Size: rows=8, columns=NAN+PER<br>Description: PER $2 gave a birth. | 7 | 1 | 0 | 0 |
| Size: rows=7, columns=ORG+ORG<br>Description: Someone reported something (undetermined) | - | - | - | - |
| Size: rows=5, columns=ORG+ORG<br>Description: ORG $1 beat ORG $2 in a football game. | 2 | 1 | 1 | 1 |
| Size: rows=5, columns=FAC+ORG<br>Description: (undetermined) | - | - | - | - |
| Size: rows=5, columns=NAN+PER<br>Description: PER $2 lost in a boxing game. | 3 | 2 | 0 | 0 |
| Size: rows=4, columns=ORG+PER+PER<br>Description: PER $2 and PER $3 contributed to ORG $1 winning a game. | 2 | 1 | 1 | 0 |
| Size: rows=4, columns=PER+PER<br>Description: $1 and $2 fought in an election. | 3 | 0 | 1 | 0 |
| Size: rows=4, columns=GPE+ORG<br>Description: ORG $2 launched a new product in GPE $1. | 3 | 0 | 1 | 0 |
| Size: rows=4, columns=GPE+PER+PER<br>Description: PER $2 protest to PER $3 about the war in GPE $1. | 4 | 0 | 0 | 0 |
| Size: rows=4, columns=PER+PER+PER<br>Description: CIA leak case (undetermined) | - | - | - | - |
| Size: rows=4, columns=GPE+PER<br>Description: PER $2 died in GPE $1. | 2 | 1 | 1 | 0 |
| Size: rows=4, columns=NAN+PER<br>Description: PER $2 was involved with abusing prisoners. | 4 | 0 | 0 | 0 |
| Size: rows=4, columns=ORG+PER<br>Description: PER $2 had some legal battle with company ORG $1. | 3 | 0 | 1 | 0 |
| Size: rows=4, columns=NAN+PER<br>Description: PER $2 was killed or injured. | 3 | 0 | 1 | 0 |
| Size: rows=3, columns=NAN+PER<br>Description: PER $2 reported a company's profit. | 3 | 0 | 0 | 0 |
| Size: rows=3, columns=PER+PER+PER<br>Description: CIA leak case (undetermined) | - | - | - | - |
| Size: rows=2, columns=NAN+ORG<br>Description: Researchers at ORG $2 had some scientific discovery. | 2 | 0 | 0 | 0 |
| Total | 51 | 8 | 11 | 4 |

Table 4.12: Evaluation of 20 tables (randomly chosen)

b. The feature is associated with a wrong entity (COREF). This is due to errors in the coreference resolution stage. Some anecdotal examples include the confusion of people that have the same surname (e.g. a murder victim and his/her family member), or a sports team name (`ORG`) and its home place (`GPE`) (e.g. "`Detroit Pistons`" and "`Detroit`"). This type of error can be reduced by improving the performance of the coreference resolution.

c. The feature has an incomplete form (INCOMPLETE). In our system, a verb in a local feature is reduced to its base form in order to simplify the feature extraction process. Furthermore, certain verbs are omitted in a regularized tree. For example, the expressions "`visited`" and "`planned to visit`" are both converted into the same local feature "`SBJ:visit`." But in some events, we actually need these information to differentiate two distinct events. These errors may be reduced by using a more complex feature representation.

d. The feature is too weak to be used for identifying the relation type (WEAK). For example, a single expression "`die`" alone is too weak for putting the relation instance into a "kill" table. This is either because the weight assigned to the feature is unreasonably large, or because the weights of other features are unreasonably small. They leads to a false mapping object between two relations. Since we currently rely on a similarity metric of two vectors for the score of a mapping, this type of error may be reduced by introducing a more complicated feature weighting schema.

e. False feature (FALSE). The feature is correctly extracted, but the same feature accidentally appears multiple times for different events in a single article set. For example, many articles about military conflicts between Israel and Palestinian government often provide a historical context of both attacks, so normally such articles include several attack-related events that occurred at a different time. Such features cause several distinct events to be mixed up, resulting in a wrong relation instance clustered into a wrong table.

f. Disjoint feature (DISJOINT). In our system, an expression that takes multiple arguments is split into multiple local features. For example, an expression "`PER visited GPE`" is converted into two local features: "`PER visit`" and "`GPE is visited`." This decomposition normally allows us to capture an event that spans multiple sentences, but sometimes this disjoint nature causes an erroneous effect. An example was found in the following article:

> *BEIJING – Japan's trade minister arrived in Beijing on Tuesday for talks with Chinese Premier Wen Jiabao, the highest-level contact between the two countries since relations soured last October. The trip is part of efforts by Tokyo and Beijing to repair ties severely frayed by disputes over undersea gas deposits, Japanese Prime Minister Junichiro Koizumi's visits to a war shrine, and other issues. Japanese Economy, Trade and Industry Minister Toshihiro Nikai arrived in Beijing Tuesday night, a Japanese Embassy official said on condition of anonymity, in line with policy.*

From the above sentences, we obtained two local features: "`arrive in GPE (Beijing)`" and "`PER (Koizumi) visit`" that are accidentally close to each other. So one tends to infer "`Koizumi visited Beijing`," but this is wrong, as his "visit" in this articles refers to his visit to a war shrine in Japan, not to the capital of China.

Table 4.13 shows the frequency of each error type. Note that some errors may have multiple causes, so that the total number of all errors does not necessarily amount to ten. It turned out

| | Wrong Relation | Wrong Value |
|---|---|---|
| a. PARSE | 0 | 1 |
| b. COREF | 2 | 5 |
| c. INCOMPLETE | 1 | 1 |
| d. WEAK | 8 | 2 |
| e. FALSE | 1 | 1 |
| f. DISJOINT | 1 | 2 |

Table 4.13: Analysis of 20 errors (10 Wrong Relation and 10 Wrong Value). Some errors may have multiple causes.

that "Wrong Relation" errors were largely due to weak features, which bear false mapping objects. Currently, the score of a mapping object is the similarity of two feature vectors whose weight is computed by an IDF (Inverse Document Frequency)-like formula, and we use a simple thresholding mechanism to filter valid mappings. However, since this filtering process can be considered as a classification task, we might be able to filter a mapping object based on more complicated feature weighting and comparison mechanism. Also, we found that many "Wrong Value" errors were due to coreference errors. While the first four errors (INCOMPLETE, PARSE, COREF and WEAK) can be reduced by improving various parts of our system, the last two errors (FALSE and DISJOINT) are more serious and show fundamental limitations of our method, although they were not so frequent. We have not yet found a clear solution to these problems.

# Chapter 5

# Discussion

In the last chapter, we presented our experimental results and its quantitative analysis. However, our system has many facets that cannot be easily captured with quantitative analyses alone. In this chapter, we discuss its qualitative aspects. We also take a look at the way of using the by-products of our system for other tasks.

## 5.1 Coverage of the Variety of Relations

In the previous chapter, we have found that we can obtain most of the ACE-like relations by keyword search. However, we still don't know what kind of relations we can obtain overall. Although it is impossible to answer this question precisely, we can answer its complementary question: what kind of relations *cannot* be obtained? By trying to answer this question, we try to estimate the scope of our system.

To see what kind of relations were missed, we take a look at isolated events (article sets) that were not grouped into any cluster. As shown in Table 5.1, some articles (like a scientific discovery or an explanation of a government's plan) require descriptive statements rather than relational statements between entities. So it's not surprising that our system could not discover a valid relation for these articles. Also, most of these events except Event #7 do not seem very repetitive, although it depends on how we judge a certain event was "repeated." Based on our observation on the obtained tables, we can roughly say that our system can discover most of the relation types that traditional IE systems were targeting, including:

- Personal events (birth, death, marriage, murder, trip or being arrested)
- Military operations (missile launching or bombing)
- Legal events (lawsuit, convict, sentence, etc.)
- Personnel affairs (promotion, resign or trade)
- Business events (merger, product launch)
- Natural disasters (disease outbreak, hurricane, earthquake)
- Sports results

| Event (Size) | Description |
|---|---|
| #1 (118 articles) | The Ethan Allen capsized and sank in Lake George. |
| #2 (38 articles) | Researchers in Merck & Co. discovered a treatment for HPV virus. |
| #3 (109 articles) | Explanation of the Bush administration's plan for pandemics. |
| #4 (34 articles) | People in New Orleans joined demonstration at National Mall organized by Farrakhan. |
| #5 (39 articles) | Roche is pressed to produce more Tamiflu. |
| #6 (74 articles) | President Bush announced a plan for pandemics. |
| #7 (46 articles) | President Bush planed to visit Argentina to attend America's Summit. |
| #8 (46 articles) | Wal-Mart held a conference. |
| #9 (26 articles) | Book review |
| #10 (73 articles) | Airplane ranoff runways at Midway. |

Table 5.1: Isolated events (randomly chosen)

However, some relations are still impossible to discover for a couple of reasons. Since we can only extract what's written in the text, it is impossible to obtain a relation which rarely appears in newspapers in the first place. However, it is also very difficult to discover "implicit" relations. For example, the following sentence:

`Condoleezza Rice met with Chinese President Hu Jintao in Beijing.`

does not mention Condoleezza Rice's physical movement. But, knowing she is the Secretary of State in the U.S., a reader could easily infer that she actually traveled to China to meet its president. We have yet to know any solid approach to solve, or even analyze these problems empirically. These need to be explored more extensively in the future.

## 5.2 Usability Issues

Eventually, all computer systems have to provide a way to interact with their users. For applications like Preemptive IE, the user interface is particularly important because of the complex nature of information that a user has to deal with. This section discusses the possibilities of the user interface of our system. We first overlook the merits and demerits of the interface used in the current system, then we propose another possibility for making use of a Preemptive IE system in a different way.

### 5.2.1 Pros and Cons for Keyword Search

In the system we developed in this thesis, we adopted a simple keyword search: a user can find a desired table by its keywords (global features), typical expressions (local features), and actual entity names filled in each column. We also let a user restrict tables with their Named Entity types (e.g. "find a table that contains a keyword "`baseball`" and has two organization names in its columns.") When there are multiple tables that match the keywords, the system tries to return

all of them. Actually, our impression of this approach was pretty good, as we got tables that were nicely "disambiguated" with various news events. For example, by searching tables with a keyword "`kill`," the following relations were returned as separate tables:

- Death by military conflicts
- Death by murder
- Death by bombing
- Death by assassination
- Death by natural disasters (e.g. storms)

Although some people might want to have a "kill" table that subsumes all the above categories, normally we consider these events as "different" ones. This was an unexpected result, but turned out to be a useful feature of our system.

However, there are still some shortcomings of keyword search. One of the most notable problems was the obscure association between relations and keywords. For example, the keyword "`baseball`" is not very effective for finding articles about baseball games because not many baseball-related articles actually contain the word "baseball", but a user normally does not know this fact in advance. This kind of problems might be solved by a technique called query expansion, which is commonly used in the Information Retrieval community. Also, we have noticed there were some funny associations between the types of events and the parts of speech used in the expressions. For example, we found that when a user types in a keyword "`injure`," which is normally contained in local features, we had several tables with a lot of people injured due to accidents, attacks or natural disasters. But when a user types in a keyword "`injury`," which is contained in global features, the most tables returned were about the injuries of sports players.

### 5.2.2 Alternative Interface - Queryless IE

As an alternative to keyword search, we have also come up with an idea called "Queryless IE." It lets a user pick one article while they are browsing and provides a list of the similar events that happened in the past. For example, if a user is reading an article about a storm, a Queryless IE system can present a list of the past hurricanes and the affected places. This can be done by searching for tables that include the tuples extracted from the event from the current article. To achieve this, the system has to cluster all the past articles including the current article in advance. An interesting feature of this type of system is that the system can present several different viewpoints of "similar" events. For example, when we searched for the tables that contains an NE tuple ("`Peru`", "`Fujimori`",) we found there were three distinct types of events:

- Trial (Mr. Fujimori appeared in a trial in Peru.)
- Extradite (Mr. Fujimori was arrested in Peru.)
- Election (Mr. Fujimori is willing to run for election in Peru again.)

One of the drawbacks of this type of interface is the scarcity of its coverage; because only one sixth of the articles were actually grouped into a large cluster[1], the system cannot always provide

[1]See Table 4.6.

such a result to a user. Although we haven't evaluated the usefulness of this interface, this type of interface could be a useful "plug-in" for existing browser applications.

## 5.3 Applying the Obtained Features to Other Tasks

In this section, we try to use the local features that were used for clustering as "by-products" for other purposes. As we explained in Section 3.5.1, a local feature is an expression (GLARF structure) that takes an entity as its argument. We clustered pairs of NE tuples (mappings) instead of NE tuples individually (cf. Section 3.6.3). Our clustering algorithm works in a way that the existing features in each cluster are reinforced, and a popular feature gets even more popular as the cluster grows. Table 5.2 shows the top five frequent local features at the initial and final stage. For each cluster, the first row shows the local features of a partially grown cluster after the first 5,000 mappings were processed. The second row shows the features of a fully grown cluster after all the mappings were processed. In the most clusters, one can observe that the popular features in its initial stage are likely to remain popular until the end. Finally, we can take the top-ranked local features in each cluster. After identifying each cluster, i.e. figuring out the relation the cluster is representing, one can use these expressions as "seed" patterns for a more sophisticated IE system that is tuned for a particular relation.

To measure how useful these patterns are, we conducted a quick experiment: we built an IE system that relies on only the patterns shown in 5.3. For the sake of simplicity, we only considered two-argument relations. First, we collect the top five features for each argument from a cluster which has grown large enough. For example, from a cluster that represents a "Murderer and Victim" relation, we can take expressions such as "`[PERSON]'s lawyer`" for a murderer and "`[PERSON]'s body`" for a victim. Then we try to extract Named Entities by only using these patterns from all the document sets we have obtained. If patterns for both argument (Murderer and Victim) match, we pick that event as a "Murder" event. Out of 35,398 document sets, we have found 255 Murder events by using these pattern set. We reviewed 20 Named Entity pairs to measure the event-level precision, and found 65% of events (about 165 events) were correct. Then we conducted the same experiment using a hand-crafted pattern set with only one pattern for each argument ("`[PERSON] kill`" and "`[PERSON] is killed`"[2].) We have got 93 events with 85% precision.

We have conducted this experiment for two relations: "Murder" and "Merger," and got similar results for both of them (Table 5.3). In both relations, a hand-crafted pattern set (with only one expression) was better in its precision, but a pattern set obtained from clusters had much better (more than two times) recall. What this result suggests is that, normally a stereotypical expression (such as "`A kill B`" or "`P buy Q`") is not used so frequently, and these events can be written in a more varied, or "paraphrased" form. Local features taken from a cluster can help infer these varied patterns. Also, local features can give a clue of patterns for a relation where a person cannot easily come up with its stereotypical expressions. Table 5.4 shows a couple of such relations. For example, one can easily find a pattern that captures the person who is an election candidate from these expressions.

[2]Note that these patterns are in fact disjointed GLARF structures, so an expression like "`A killed B`" actually matches with both patterns at once.

Cluster-189

| | |
|---|---|
| Initial (rows=15) | ORG's-take (14), ORG-renounce (13), ORG-reject (13), ORG-recognize (13), ORG-win (12) |
| Final (rows=125) | member-of-ORG (57), ORG-win (45), ORG-take (43), ORG-form (36), ORG-recognize (34) |

Cluster-268

| | |
|---|---|
| Initial (rows=25) | ORG-win (20), lead-ORG (20), ORG-take (16), ORG-make (16), ORG-go (15) |
| Final (rows=272) | ORG-win (135), lead-ORG (85), ORG-play (84), ORG-make (83), ORG-lose (79) |

Cluster-304

| | |
|---|---|
| Initial (rows=4) | PER's-client (3), PER's-lawyer (3) PER-ask (2), PER-attorney (2), PER-describe (1) |
| Final (rows=138) | PER-lawyer (84), PER's-client (65), PER-attorney (46), PER-lawyer-say (26), PER-attorney-say (23) |

Cluster-319

| | |
|---|---|
| Initial (rows=11) | GPE's-minister (7), GPE's-withdrawal (6), GPE's-border (6), GPE's-election (5), GPE's-destruction (5) |
| Final (rows=197) | GPE's-minister (67), GPE's-election (59), GPE's-president (50), GPE-say (50), GPE's-party (40) |

Cluster-1250

| | |
|---|---|
| Initial (rows=7) | GPE's-election (5), GPE's-economy (5), GPE's-system (4), GPE's-party (4), GPE's-minister (4) |
| Final (rows=173) | GPE's-election (77), GPE's-president (56), GPE's-minister (51), GPE's-party (44), GPE's-government (32) |

Table 5.2: Snapshots of growing clusters. "Initial" states are taken after 5,000 mappings were processed. "Final" states are taken after all the mappings were processed.

Murder - Murderer and Victim

Obtained patterns (255 events, 65% were correct)

| (for Murderer) | (for Victim) |
|---|---|
| `[PERSON]'s lawyer` | `[PERSON]'s body` |
| `[PERSON]'s attorney` | `[PERSON] is killed` |
| `[PERSON] is charged` | `[PERSON]'s death` |
| `[PERSON]'s lawyer say` | `[PERSON]'s body is found` |
| `[PERSON] is convicted` | `[PERSON]'s family` |

Hand-crafted patterns (93 events, 85% were correct)

| (for Murderer) | (for Victim) |
|---|---|
| `[PERSON] kill` | `[PERSON] is killed` |

Merger - Parent Company and Subsidiary

Obtained patterns (133 events, 63% were correct)

| (for Parent) | (for Subsidiary) |
|---|---|
| `[ORGANIZATION] buy` | `buy [ORGANIZATION]` |
| `[ORGANIZATION]'s bid` | `for [ORGANIZATION] offer` |
| `[ORGANIZATION]'s offer` | `bid for [ORGANIZATION]` |
| `[ORGANIZATION] say that` | `[ORGANIZATION]'s board` |
| `[ORGANIZATION] say in` | `acquire [ORGANIZATION]` |

Hand-crafted patterns (46 events, 85% were correct)

| (for Parent) | (for Subsidiary) |
|---|---|
| `[ORGANIZATION] buy` | `buy [ORGANIZATION]` |

Table 5.3: The extraction results for the "Murder" and "Merger" relation using the patterns obtained from clusters and their hand-crafted counterparts. For the reader's convenience, we rewrote the output in GLARF representation into an ordinary phrase-like denotation.

Election - Country and Candidate

| (for Country) | (for Candidate) |
|---|---|
| [GPE]'s election | [PERSON]'s party |
| [GPE]'s president | [PERSON] win |
| [GPE]'s party | [PERSON]'s government |
| [GPE]'s minister | [PERSON]'s supporter |
| [GPE]'s economy | [PERSON] lead |

Baseball game - Team and its winning Pitcher

| (for Team) | (for Pitcher) |
|---|---|
| [ORGANIZATION]'s rotation | [PERSON] throw |
| join [ORGANIZATION] | [PERSON] allow |
| [ORGANIZATION]'s victory | [PERSON]'s start |
| [ORGANIZATION] win | [PERSON] pitch |
| [ORGANIZATION] need | [PERSON] allow to run |

Sentence - Offender and Judge

| (for Offender) | (for Judge) |
|---|---|
| [PERSON] is sentenced | [PERSON] sentence |
| [PERSON] is sentenced to | [PERSON] sentence to |
| [PERSON]'s lawyer | [PERSON] impose |
| [PERSON] plead in | [PERSON] is told |
| [PERSON] plead | [PERSON] give |

Table 5.4: Typical expressions that appeared in several clusters.

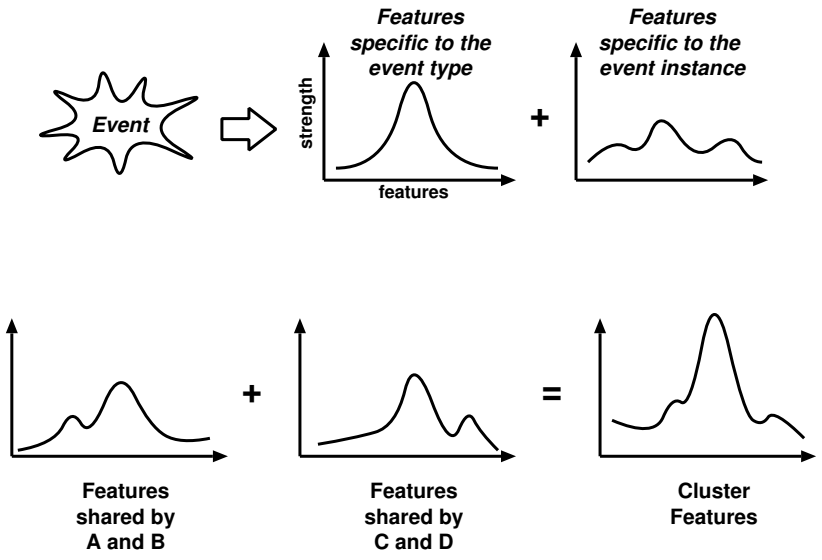


Figure 5.1: How the clustering procedure works. The feature are split into two sets: features specific to the event type and features specific to the event instance (above.) The clustering proceeds in a way that the salient features from both mapping get strengthened by each other, making those features more strong (below.)

### 5.3.1 Why Useful Expressions are Obtained?

Now, we try to explain why these "telling" patterns can emerge during the clustering procedure. First, let us assume that a feature set obtained from a certain event (article) is made up of two different sets of features: features that are specific to its event *type*, and features that are specific to its event *instance*. For example, an expression "`PERSON is killed`" is specific to, say, the "Murder" event type, whereas other expressions such as "`PERSON 's boyfriend`" are specific to the particular event instance. We further assume that these instance-specific features are varied enough, they behave more like noise. So after combining the features from several events, these features eventually start canceling each other, leaving the type-specific features even more salient. This process is shown figuratively in Figure 5.1. Since our clustering algorithm picks the strongest mapping between two event instances first, each initial cluster in the clustering procedure is likely to have several salient features, which attract more features of the same kind. Although this procedure does not guarantee the optimal results, it is still likely that some clusters can gather useful expressions in this way.

# Chapter 6

# Related Work

Our research is sitting at the crossing point of several major research trends in Information Extraction. In this chapter, we briefly look at these related works. Figure 6.1 shows a rough sketch of the connection between our research and prior work.

There have been two major trends that are directly connected to our research. First, IE has been long focusing on scenario customization. Since traditional IE systems have to rely on manually crafted patterns, many attempts for reducing this cost have been made. This is closely tied to another trend of research, which is automatic pattern acquisition. Our work is heavily motivated by these two trends and the idea of Preemptive IE is an attempt to indicate another direction for these problems.

## 6.1 Scenario Customization

Traditionally, an IE system has been created for a particular scenario. To adjust the IE system for a different scenario, the major parts of the system had to be rebuilt or tuned manually. Research about scenario customization has been conducted since the idea of IE was first conceived. The main goal of this research is to find a better system structure that facilitates its reuse and reduces the cost of redesign. It also tried to provide a better mechanism to tune or control an existing IE system to adapt to a new scenario.

In 1995, the notion of Named Entity was first introduced in the MUC-6 evaluation. The notion of NE was useful to separate a layer which is somewhat independent of each task and facilitated the reuse of its components [10]. Most IE systems these days still use these notions and separate pattern recognition and NE recognition layers. In 1997, Yangarber et al. built a pattern construction tool for domain experts [27]. The system provides a graphical user interface that helps a user to create patterns for a new scenario. It uses an NE as one of the primary building blocks of IE patterns, and allows a user to search articles, find salient expressions, and try applying a manually created pattern to articles. Aone et al. tried to design a system structure that can be tuned to various scenarios and expanded the number of scenarios with manually crafted patterns [3].

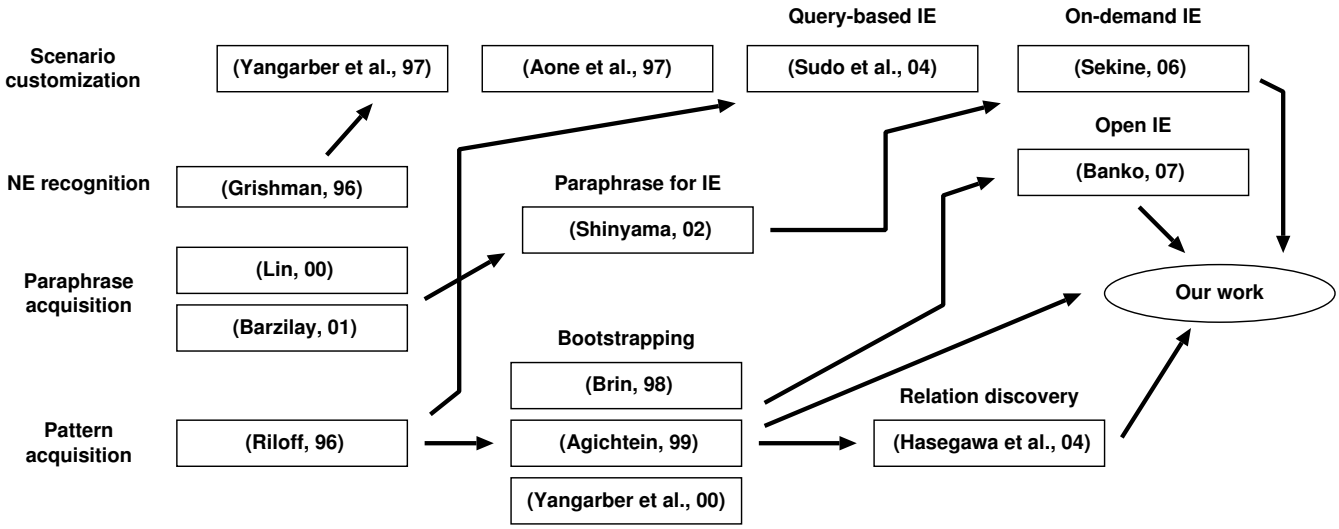


Figure 6.1: Related Works

## 6.2 Pattern Acquisition

The research on scenario customization greatly benefited from the research on pattern acquisition. Automatic pattern acquisition from annotated/unannotated corpora has been another major research trend in IE. Traditionally, IE patterns were either manually crafted or extracted from annotated corpora. However, both approaches suffered from large costs of human labor, so obtaining patterns with minimal human labor was demanded. In 1996, Riloff proposed to use preclassified (but not annotated) documents for pattern acquisition [19]. "Preclassified" means that documents were classified as either relevant or irrelevant for a certain scenario, and the system tried to learn positive or negative examples of expressions from these documents. This idea is later expanded by using an IR system for document selection.

After Riloff's work, many researchers have tried pattern acquisition from unannotated corpora with bootstrapping or co-training. These systems tried to exploit the duality of the pattern representation. They trained two mutually dependent learning systems that train each other. In a typical bootstrapping system, a pattern representation is used for learning its surrounding context, and a context representation is used for learning the pattern it matches. Yangarber et al. used the class of documents [28], and Brin and others used the arguments (NE tuples) of patterns [6, 2] as the alternative contexts to learn. They used some "seeds" in order to initiate the learning process. The idea of bootstrapping spawned an active area of machine learning research represented by Collins et al [8]. The main drawback of these approaches is the selection of the seed patterns/tuples provided by a user and its stopping criteria. Some researchers later tried to solve this by introducing various heuristics. Yangarber et al. proposed the idea of counter-learning [26]. In counter-learning, multiple acquisition processes are run parallelly until the results from multiple processes start overlapping with each other.

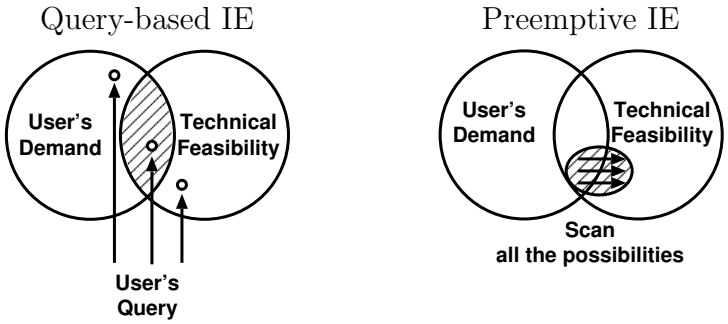


Figure 6.2: Query-based IE and Preemptive IE

## 6.3 Query-based IE

Pattern acquisition by Riloff required manually classified documents for training. Sudo et al. proposed to automate this classification as a part of the scenario customization process. This idea is called Query-based IE to combine scenario customization with pattern acquisition. They tried to control the pattern acquisition process with a user's queries, using an Information Retrieval technique to obtain relevant training documents [24]. This idea eventually grew into On-demand IE by Sekine [21]. In a On-demand IE system, a user first types in a query that triggers documents which can be used for automatic pattern acquisition. Then the system detects patterns that are equivalent to each other and use the obtained patterns to conduct actual extraction. The original system that Sudo et al. has created still requires a user's manual annotation of obtained patterns as postprocessing, but Sekine's system has completely eliminated this need by exploiting other language resources and heuristics.

Both ideas, Query-based IE and On-demand IE, can be considered as customizing scenarios based on a user's query. The assumption behind these attempts is that a user can always control document selection perfectly, which is necessary for pattern acquisition at a later stage. However, since a user does not have direct control over obtained IE patterns, it is not always easy to describe a desired relation by just selecting training documents with a list of keywords, even if s/he understands precisely how the pattern acquisition process works.

Query-based systems also impose a difficulty on evaluation. Since all the results can be changed by a user's query, an evaluation of these systems has to measure how easy it is to craft a good query that returns good results. However, this is greatly affected by its user interface, so various factors of the system need to be considered in the evaluation.

### 6.3.1 Query-based IE and Preemptive IE

Our work, the idea of Preemptive IE was originally conceived to answer the above issues. The biggest problem for a query-based system is that a user does not get any direct feedback and does not know how to improve the query to get better results. Especially, it is extremely difficult for a

user to improve the coverage (recall) of the results unless s/he comes up with a very clever (but often counter-intuitive) query. Furthermore, as we speculated in Section 2.1.2, an average user might not even have a clear idea about what kind of relations can be, or should be, extracted in the first place, because there are so many relations from various viewpoints. Therefore, some sort of "probing" mechanism is desired. However it is not easy to systematically "scan" all the IR queries. In a query-based system, a user has to resort to a lot of trial-and-error without any clue.

Figure 6.2 shows the schematic difference between Query-based IE and Preemptive IE. A successful IE application has to sit in the middle where a user's demand and technical feasibility meet. Query-based IE lets a user create arbitrary points in the right circle based on keywords, which may or may not give satisfactory results. In contrast, Preemptive IE "scans" or "crawls" a certain area of the feasible scenarios, performs extraction, and then lets a user choose the best result.

## 6.4 Relation Discovery and Open IE

As the availability of (both linguistic and computational) resources has grown and the techniques of automatic pattern acquisition has gotten sophisticated, some researchers started trying to expand the number of IE scenarios automatically. Unlike a previous attempt by Aone, this new research trend relies on minimal human intervention. In 2004, Hasegawa et al. proposed a technique to discover relations in an unsupervised manner [11]. They first extracted a pair of adjacent Named Entities and the words between the two NEs, and then tried to cluster them using its contexts. Banko et al. tried to discover relations from a much larger document set such as Web [4]. They used a similar method for extracting adjacent entity pairs and the in-between texts from a corpus by using a chunker, and tried to identify the correct relations with frequency counting. Since these attempts are somewhat similar to our research, it is worth mentioning the difference between their works and ours. There are at least three major differences between these approaches and our research:

1. First, both approaches rely on rather surface features. Hasegawa takes a word sequence and Banko uses a shallow parser to extract the words between two entities in order to identify the relation. This makes it difficult to take into account a richer feature set that appears in a more global context. Thus it is hard to capture a relation that spans multiple sentences. Also, both approaches assume the entities involved with a certain relation must be adjacent to each other and the relation is always of binary form $R(X, Y)$.
2. Both attempts focused on finding somewhat well-known relations that can be stated concisely and explicitly, such as "`A is the president of B`" or "`P was born in Q`." This is understandable when they use surface features and they have to rely on high-frequency relations to ensure its accuracy. However, since relations in news articles are often described in an obscure way, these approaches are not suitable for finding relations for news events reliably from a small number of articles.
3. Both approaches employed simpler relation identification mechanism. Hasegawa et al. clustered the obtained relations by using surrounding words to group the "equivalent" relations. Banko et al. tried to reduce the obtained expressions to increase its uniformity by removing stop words or using stemming. Both approaches suffer from the variety of expressions.

Furthermore, because of the lack of rich features, these approaches cannot distinguish the relations that have almost identical expressions but still semantically different, such as "`PER beat PER`" in election and "`PER beat PER`" in sport events, as we presented in Section 2.1.2.

## 6.5 Handling Varied Expressions

The research of Information Extraction would become much easier if all newspapers always used the same expression for the same type of events. However, this has never been the case. A creative reporter or editor always tries to use various expressions to express certain things. Research about paraphrasing has been active since around 2000. Lin et al. tried to acquire similar expressions [13] from monolingual corpora by comparing arguments of predicates. They took a set of expressions whose arguments are highly correlated. Barzilay et al. tried to acquire paraphrases from parallel corpora [5], using the word and part-of-speech tag alignment. An attempt to apply these ideas for acquiring IE patterns was done by Shinyama et al. [23]. They used comparable corpora and tried to take the expressions that share the same Named Entities from articles that report the same event. However, these approaches do not scale well for varied expressions reliably.

# Chapter 7

# Conclusion

In this thesis, we presented an approach in order to explore the possibility of Information Extraction scenarios. We have proposed a framework called "Preemptive Information Extraction," which discovers various relations with entities from news articles in an unsupervised manner. Two important ideas were introduced: one is to separate the notion of relation detection and relation identification, and the other is to use clustering technique to identify the type of newly discovered relations. We built a preliminary system that uses news sources on the Web and performs Preemptive IE in a reasonable amount of time, and then evaluated the system in terms of the performance and the relation coverage. We also discussed the various aspects of the system, including its usability and the possible use of its by-products, such as obtained expressions.

## 7.1 Future Work (System Wide)

In this section, we suggest a couple of improvements on the performance of a Preemptive IE system as a future direction.

### 7.1.1 More Named Entity Categories

The current system uses six different categories for Named Entities. Since covering more NE types will increase the coverage of entities within one article, we can expect that using more NE types will increase the number of obtainable features we can use for clustering. Also, the current system does not recognize numerical or temporal expressions as an entity. Sekine et al. has proposed hundreds of NE categories that are hierarchically organized [22]. However, increasing the variety of features might also cause a data sparseness problem. We might be able to utilize the hierarchical nature of the NE categories to provide some back-off for infrequent features.

### 7.1.2 Improvement on Features and its Scoring Metrics

In the current system, we have used two different types of features: global features (a bag of words) and local features (a GLARF structure of an expression). Obviously, we could introduce more features other than these. One of the major drawbacks of using many features is that it would become more difficult to create an association of two relation instances (a mapping object) because

there will be more parameters to take into account. Currently, we are taking a simple unsupervised approach: thresholding with the weights of all the features. However, we could use a more complex binary classifier that can be trained in a supervised manner.

### 7.1.3 Evaluations of Relation Coverage

We have used the ACE event types in this thesis for evaluation. Although they cover most of the major Information Extraction tasks that are currently being tried, there is still a question if the coverage of the obtained relations is sufficient. We hope that there will be research in the future that tries to answer this question more extensively by estimating the coverage of other popular categories of events. Especially, we are interested in what kind of news events can (or should) be represented as a table, and what kind of events can (or should) *not*. However, great effort may be needed to establish an acceptable agreement on the categorization of events.

## 7.2 Future Work (Broader Directions)

In this thesis, we focused on the idea of Preemptive IE for news articles only. However, we can expect that our basic idea can be applied to more varied types of texts such as technical documents or more general documents on the Web, or even to a different application other than IE such as summarization. At the same time, we might be able to use our system as a basic tool for richer research efforts about semantics. Is it possible to quantify the generality of relations in some meaningful way? To what extent a does person recognize a relation as a "solid" or "familiar" one? Also, relations with various degrees of generality might form some hierarchical structure, but what would it look like? We hope our ideas provide some useful footholds for future research in Natural Language Processing.

# Bibliography


[1] APPENDIX A: EVALUATION TASK DESCRIPTION. In *THIRD MESSAGE UNDERSTANDING CONFERENCE (MUC-3): Proceedings of a Conference Held in San Diego, California*, 1991.

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

[3] Chinatsu Aone and Mila Ramos-Santacruz. A Large-Scale Relation and Event Extraction System. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP-00)*, 2000.

[4] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, January June–December 2007.

[5] Regina Barzilay and Kathleen R. McKeown. Extracting Paraphrases from a Parallel Corpus. In *Proceedings of the ACL/EACL*, 2001.

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

[7] Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of NAACL-2000*, 2000.

[8] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of EMNLP 1999*, 1999.

[9] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, 1998.

[10] Ralph Grishman and Beth Sundheim. Message Understanding Conference - 6: A Brief History. In *Proceedings of the COLING*, 1996.

[11] 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-04)*, 2004.

[12] LDC. ACE (Automatic Context Extraction) English Annotation Guidelines for Events. 2005.

[13] Dekang Lin and Patrick Pantel. Discovery of Inference Rules for Question Answering. *Natural Language Engineering*, 7(4):343–360, 2001.

[14] Adam Meyers, Ralph Grishman, and Michiko Kosaka. Formal Mechanisms for Capturing Regularizations. In *Proceedings of LREC-2002*, Las Palmas, Spain, 2002.

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

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

[17] NIST. The ACE 2005 Evaluation Plan. 2005.

[18] M. F. Porter. An algorithm for suffix stripping. pages 313–316, 1997.

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

[20] Jerome H. Saltzer, David P. Reed, and David D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[21] Satoshi Sekine. On-demand information extraction. In *ACL*. The Association for Computer Linguistics, 2006.

[22] Satoshi Sekine, Kiyoshi sudo, and Chikashi Nobata. Extended Named Entity Hierarchy. In *Proceedings of the LREC*, 2002.

[23] Yusuke Shinyama, Satoshi Sekine, Kiyoshi Sudo, and Ralph Grishman. Automatic paraphrase acquisition from news articles. In *Proceedings of HLT 2002*, 2002.

[24] Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. pre-codie: crosslingual on-demand information extraction. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 25–26, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[25] Charles L. Wayne. Topic Detection & Tracking: A Case Study in Corpus Creation & Evaluation Methodologies. In *Proceedings of the LREC*, 1998.

[26] Roman Yangarber. Counter-Training in Discovery of Semantic Patterns. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, 2003.

[27] Roman Yangarber and Ralph Grishman. Customization of Information Extraction Systems. In *Proceedings of the International Workshop on Lexically Driven Information Extraction*, Frascati, Italy, 1997.

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