GLOBAL RULE INDUCTION FOR INFORMATION EXTRACTION

XIAO JING

NATIONAL UNIVERSITY OF SINGAPORE

2004

GLOBAL RULE INDUCTION FOR INFORMATION EXTRACTION

XIAO JING
(B.S., M.Eng., Wuhan University)

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SINGAPORE
2004

Acknowledgements

There are many people whom I wish to thank for their support and for their contributions to this thesis. First and foremost, I would like to thank my advisor Professor Chua Tat-Seng, who has played a critical role in the completion of this thesis and my PhD study career. Throughout my time at NUS, Prof. Chua was the source of many appealing research ideas. He is always patient to advise me how to write research papers and how to be a good researcher. I will not hesitate to tell new postgraduate students, Prof. Chua is a great supervisor.

Next, I would like to thank Prof. Tan Chew-Lim and Prof. Ng Hwee-Tou who have provided complementary perspectives and suggestions for improving the presentation of ideas. I thank them for their participation.

I also would like to thank all of my friends in Singapore and colleagues in NUS. Especially, I thank Dr. Liu Jimin who gave me many suggestions on the ideas in the information extraction research field. Special acknowledgements are due to Cui Hang and Lekha Chaisorn who helped me working out the experiments in Chapter 5. I would like to thank all of my labmates in the multimedia group, NUS. They are: Dr. Ye Shiren, Dr. Zhao Yunlong, Dr. Ding Chen, Feng Huamin, Wang Jihua, Xu Huaxin, Yang Hui, Wang Gang, Shi Rui, Qiu Long, Steve and Lisa. I thank them for their friendship and support.

Finally, I would like to thank my family members, my parents and my sister who have been supporting me all these years in my student career. Without their love and support, this dissertation would have never happened.

Table of Contents

Summary

Information Extraction (IE) is designed to extract specific data from high volumes of text, using robust means. IE becomes more and more important nowadays as there are huge amount of online documents appearing on the Web everyday. People need efficient methods to manage all kinds of text sources effectively. IE is one kind of such techniques which can extract useful data entries to store in databases for efficient indexing or querying for further purposes. There are two broad approaches for IE. One is the knowledge engineering approach in which a person writes special knowledge to extract information using grammars and rules. This approach requires skill, labor, and familiarity with both domain and tools. The other approach is the automatic training approach. This method collects lots of examples of sentences with data to be extracted, and runs a learning procedure to generate extraction rules. This only requires someone who knows what information to extract and large quantity of example text to markup. In this thesis, we focus on the latter approach, *i.e.* automatic training method for IE. Specifically, we focus on pattern extraction rule induction for IE tasks.

One of the difficulties in some of the current pattern rule induction IE systems is that it is difficult to make the correct decision of the starting point to kick off the rule induction process. Some systems randomly choose one seed instance and generalize pattern rules from it. The shortcoming of doing this is that it may need several trials to find a good seed pattern rule. In this thesis, we first introduce GRID, a Global Rule Induction approach for text Documents, which emphasizes the utilization of the global feature distribution in all of the training examples to start the rule induction process. GRID uses named entities as semantic constraints and uses chunks as contextual units, and incorporates features at lexical, syntactical and semantic levels simultaneously. GRID achieves good performance on both semi-structured and free text corpora.

Second, we show that GRID can be employed as a general classification learner for problems other than IE tasks. It is applied successfully in definitional question answering and video story segmentation tasks.

Third, we introduce two weakly supervised learning paradigms by using GRID as the base learner. One weakly supervised learning scheme is realized by combining co-training GRID with two views and active learning. The other weakly supervised learning paradigm is implemented by cascading use of a soft pattern learner and GRID. From the experimental results, we show that the second scheme is more effective than the first one with less human annotation labor.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Information Extraction

The World Wide Web is swiftly becoming a vast information resource that contains a great variety and quantity of on-line information. People encounter a large amount of fast growing information in the form of structured, semi-structured and free texts. This creates a great need for computing systems with the ability to process those documents to simplify the text information. One type of appropriate processing is called Information Extraction (IE) technology. Generally, an information extraction system takes an unrestricted text as input and "summarizes" the text with respect to a pre-specified topic or domain of interest: it finds useful information about the domain and encodes the information in a structured form, suitable for populating databases [Cardie, 1997]. Different from information retrieval systems, IE systems do not recover from a collection a subset of documents which are hopefully relevant to a query (or query expansion). Instead, the goal of information extraction is to extract from the documents facts about pre-defined types of events, entities and relationships among entities. These extracted facts are usually entered into a database, which may be further processed by standard database technologies. Also the facts can be given to a natural language summarization system or a question answering system for providing the essential entities or relationships of the events which are happening in the text documents.

It has been about twenty years since the first Message Understanding Conference (MUC, the main evaluation event for information extraction technology sponsored by the US government, at first by Navy and later by DARPA [MUC-3 1991; MUC-4 1992; MUC-5 1993; MUC-6 1995; MUC-7 1998]) was held in 1987. The topics of the series of MUCs are listed in Table 1.1.

Message Understanding Conferences	Topics
MUC-1(1987) and MUC-2(1989)	messages about naval operations
MUC-3(1991) and MUC-4(1992)	news articles about terrorist activity
MUC-5(1993)	news articles about joint ventures and microelectronics
MUC-6(1995)	news articles about management changes
MUC-7(1998)	news articles about space vehicle and missile launches

Table 1.1 Topics of the series of Message Understanding Conferences

An example of the information extraction task which was the focus of MUC-3 and MUC-4 is shown in Figure 1.1 and 1.2. The goal is to extract information of Latin American terrorist incidents from news articles. The source message is shown in Figure 1.1 and the filled template is presented in Figure 1.2.

DEV-MUC3-0126 (BELLCORE)

SAN SALVADOR, 15 MAR 89 (AFP) -- [TEXT] URBAN GUERILLAS ATTACKED THE PRESIDENCY IN SAN SALVADOR WITH MORTAR FIRE TONIGHT, CAUSING SOME DAMAGE BUT NO CASUALTIES, ACCORDING TO INITIAL OFFICIAL REPORTS. THE ATTACK OCCURRED AT 1835 (0035 GMT). EIGHT EXPLOSIONS WERE HEARD. IT WAS NOT REPORTED WHETHER PRESIDENT JOSE NAPOLEON DUARTE WAS AT HIS OFFICE AT THE TIME OF THE ATTACK. THE ATTACK WAS PRESUMABLY CARRIED OUT BY FARABUNDO MARTI NATIONAL LIBERATION FRONT URBAN GUERRILLAS.

Figure 1.1 A sample message from MUC-3 and MUC-4 evaluation

0. MESSAGE: ID	DEV-MUC3-0126 (BELLCORE, MITRE)
1. MESSAGE: TEMPLATE	1
2. INCIDENT: DATE	15 MAR 89
3. INCIDENT: LOCATION	EL SALVADOR: SAN SALVADOR (CITY)
4. INCIDENT: TYPE	ATTACK
5. INCIDENT: STAGE OF EXECUTION	ACCOMPLISHED
6. INCIDENT: INSTRUMENT ID	"MORTAR"
7. INCIDENT: INSTRUMENT TYPE	MORTAR: "MORTAR"
8. PERP: INCIDENT CATEGORY	TERRORIST ACT
9. PERP: INDIVIDUAL ID	"URBAN GUERILLAS" / "FARABUNDO MARTI NATIONAL LIBERATION FRONT URBAN GUERRILLAS"
10. PERP: ORGANIZATION ID	"FARABUNDO MARTI NATIONAL LIBERATION FRONT"
11. PERP: ORGANIZATION CONFIDENCE	SUSPECTED OR ACCUSED: "FARABUNDO MARTI NATIONAL LIBERATION FRONT"
12. PHYS TGT: ID	"PRESIDENCY"
13. PHYS TGT: TYPE	GOVERNMENT OFFICE OR RESIDENCE: "PRESIDENCY"
14. PHYS TGT: NUMBER	1: "PRESIDENCY"
15. PHYS TGT: FOREIGN NATION	-
16. PHYS TGT: EFFECT OF INCIDENT	SOME DAMAGE: "PRESIDENCY"
17. PHYS TGT: TOTAL NUMBER	-
18. HUM TGT: NAME	"JOSE NAPOLEON DUARTE"
19. HUM TGT: DESCRIPTION	"PRESIDENT": "JOSE NAPOLEON DUARTE"
20. HUM TGT: TYPE	GOVERNMENT OFFICIAL: "JOSE NAPOLEON DUARTE"
21. HUM TGT: NUMBER	1: "JOSE NAPOLEON DUARTE"
22. HUM TGT: FOREIGN NATION	-
23. HUM TGT: EFFECT OF INCIDENT	NO INJURY OR DEATH: "JOSE NAPOLEON DUARTE"
24. HUM TGT: TOTAL NUMBER	-

Figure 1.2 The filled template corresponding to the message shown in Figure 1.1

There are typically 5 subtasks defined by MUC-6 and MUC-7 for the information extraction task. They are recognized as independent, complicated problems:

(a) Named Entity (NE): Find and categorize proper names appearing in the text. There are 7 classes of NEs defined in MUC-7: person, organization, location, money, percentage, time and date. Usually named entities play important roles for the events appearing in the text documents. The current state-of-the-art performance of named entity recognition achieves an accuracy of around 95% in terms of F_1 measure [Bikel, Schwartz and Weischedel, 1999].

(b) Template Element (TE): find the descriptions of all entities of specified types, *e.g.* for a person, whether it is a civilian or a military official; for an organization, whether it's a commercial entity or a government agency.

(c) Co-reference (CO): find and link together all references to the "same" entity in a given text. For example, given three sentences of "*Computational Linguists* from many different countries attended Dan's EUROLANG tutorial. *The participants* managed to attend the presentation even though they spent all the night in the disco; *they* also managed to follow the presentation without falling asleep and found it very interesting.", co-reference resolution aims to link "computational linguists", "the participants" and "they" to the same entity. The best reported F_1 measure for the co-referencing task in MUC-7 [MUC-7, 1998] is around 62%. But none of the systems in MUC-7 adopted a learning approach to co-reference resolution. The state-of-the-art of machine learning approach to co-reference resolution can achieve a comparable performance to MUC-7 systems of 60% [Soon, Ng and Lim, 2001].

(d) Template Relation (TR): find broader relationships among entities, such as the "employment" relation between persons and companies.

(e) Scenario Template (ST): It is the top-level IE task to find instances of events or facts of specified types. Events are complex relations with multiple arguments, such as a terrorism attack, relating the particular terrorist activity with the date/location/victim of the attack.

Table 1.2 presents the best results reported in the tasks of information extraction in a series of MUC evaluations. In this thesis, we will focus on the top-level task, ST. For example, given a news article related to terrorism, the IE system aims to extract slot information for “perpetrator”, “victim” or “physical target” *etc.* to fill a pre-defined template as shown in Figure 1.2. Note that in order to perform well on ST task the system must be able to perform all the lower-level tasks. On the other hand, for optimal performance on a higher-level task, optimal performance on lower-level tasks may not be necessary: *i.e.*, to find all events (ST), one need not have to find all proper names (NE) in text, just those names that participate in the events that are sought. How to obtain good performance on other tasks is outside the scope of this thesis.

Evaluation	Named Entity	Coreference	Template Element	Template Relation	Scenario Template
MUC-3	╲	╲	╲	╲	R<50% P<70%
MUC-4	╲	╲	╲	╲	F<56%
MUC-5	╲	╲	╲	╲	JV: F<53% ME: F<50%
MUC-6	F<97%	R<63% P<72%	F<80%	╲	F<57%
MUC-7	F<94%	F<62%	F<87%	F<76%	F<51%

Legend: R: recall; P: precision; F: F-measure with recall and precision weighted equally; JV: joint venture; ME: microelectronics

Table 1.2 Best results reported in MUC-3 through MUC-7 by task

From another point of view, the process of information extraction has two major parts [Grishman, 1997]. First, the system extracts individual “facts” from the text of a document through local text analysis. Second, it integrates these facts, producing larger facts or new facts (through inference). As a final step after the facts are integrated, the pertinent facts are translated into the required output format. The overall flow of an information extraction system is presented in Figure 1.3. This thesis is mainly focusing on local text analysis. The discourse analysis in the second phase is outside the scope of this study.

Generally speaking, there are two basic approaches to the design of IE systems, which are called the *Knowledge Engineering Approach* and the *Automatic Training Approach* [Appelt and Israel, 1999]. The Knowledge engineering approach is characterized by the development of the grammars used by a component of the IE system by a “knowledge engineer”, *i.e.* a person who is familiar with the IE system, and the formalism for expressing rules for that system. The knowledge engineering approach requires a fairly arduous test-and-debug cycle, and it is dependent on having linguistic resources at hand,

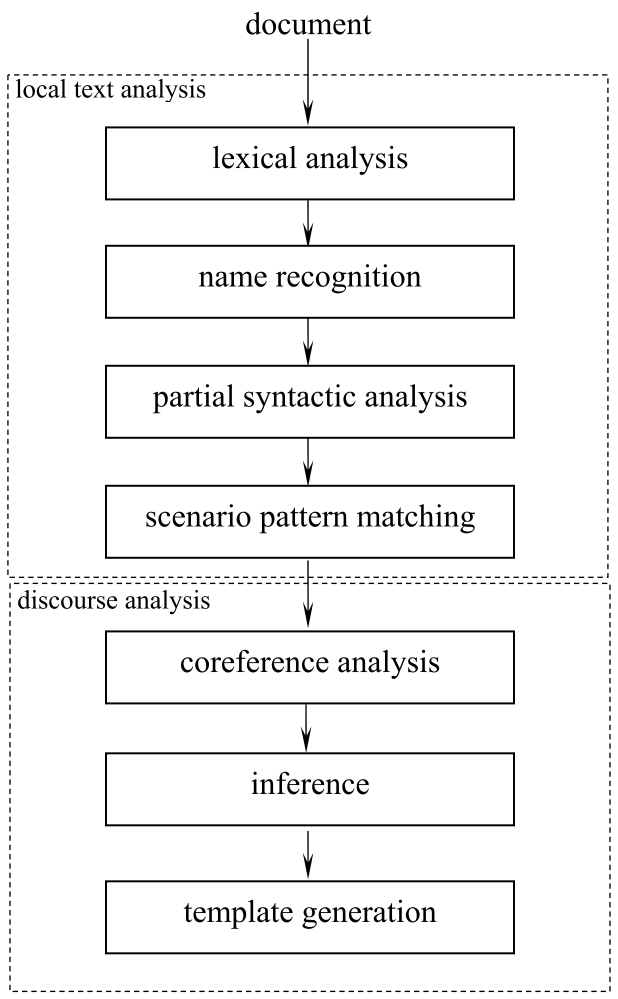


Figure 1.3 Overall flow of an information extraction system

such as appropriate lexicons, as well as someone with the time, inclination, and ability to write rules. If any of these factors are missing, then the knowledge engineering approach becomes problematic. The main problem of the knowledge engineering approach is poor portability. It is very difficult to port IE systems by knowledge engineering approaches to new applications and domains automatically.

The automatic training system is quite different. It is not necessary to have someone on hand with detailed knowledge of how the IE system works and how to write rules for it. Typically, a training algorithm is run based on a suitable annotated training corpus. Rather than focusing on producing rules, the automatic training approach focuses on producing training data. Corpus statistics or rules are then derived automatically from the training data, and used to process novel data. As long as someone familiar with the domain is available to annotate texts, systems can be customized to a specific domain without intervention from any developers. The automatic training approach is favorable when large amounts of training data can be obtained easily. This thesis will focus on the automatic training approach for information extraction.

For the automatic training approach for information extraction tasks, there are many machine learning techniques which can be applied, such as Decision Trees [Sekine, *et al.* 1998; Paliouras, *et al.* 2000], Hidden Markov Models [Freitag and McCallum, 1999; Freitag and McCallum, 2000], Support Vector Machines [Han, *et al.* 2003; Moschitti, Morarescu and Harabagiu, 2003], Maximum Entropy [McCallum, *et al.* 2000; Chieu and Ng, 2002a], Bayesian Networks [Bouckaert, 2003], Finite State Transducers [Kushmerick, Weld and Doorenbos, 1997; Hsu and Dung, 1998]. And other machine learning techniques include Symbolic Relational Learning [Califf, 1998] such as

Inductive Logic Programming (ILP) [Muggleton, 1992], we call symbolic relational learning paradigm as pattern rule induction method in general in this thesis [Muslea, 1999]. This dissertation will focus on the pattern rule induction method for information extraction.

From another point of view, two directions of IE research can be identified: Wrapper Induction (WI) and NLP-based methodologies. WI techniques [Kushmerick, 1997] have historically made scarce use of linguistic information and their application is mainly limited to rigidly structured documents, which contains heavy mark-up, in the form of SGML/HTML/XML *etc.* tags. NLP-based methodologies tend to make full use of all kinds of linguistic information and their main application is for unstructured documents such as news articles. In this thesis we focus more on NLP-based methodologies to extract facts from unstructured and semi-structured text such as seminar announcements with no mark-up tags.

1.2 Motivations

Different from the bag of words approach [Salton and McGill, 1983] employed in most information retrieval and text categorization systems, information extraction systems depend largely on relations of relevant items of surrounding context to find the extracted slots information. Since manually constructing useful extraction pattern rules is time-consuming, error-prone and it is tedious to port them to a new domain, various machine learning algorithms have been used successfully as attractive alternatives in building adaptive information extraction systems [Muslea, 1999]. We consider the following points as the motivations of this thesis:

(a) There are many IE systems which are based on rule-based relational learning methods that target at domains with rich relational structures. Such methods generate rules to extract slots either bottom-up [Califf, 1998; Califf and Mooney, 1999; Ciravegna, 2001] or top-down [Soderland, 1999]. Some methods combine the bottom-up and top-down approaches [Muggleton, 1995; Zelle, Mooney and Konvisser, 1994]. One of the difficulties in rule induction learning systems is that it is difficult to select a good seed instance to start the rule induction process. Some systems simply selected seed instances in an arbitrary order [Soderland, *et al.* 1995; Soderland, 1999]. By doing so, the system often requires to make several false starts in order to learning a high coverage concept definition [Soderland, 1997a]. In general, we expect the choice of good quality prominent features will not only minimize the false starts in inducing rules, it will also ensure that the resulting rules have higher coverage and thus more general. Thus in this thesis, we aim to make use of the global distribution of features to select the good feature in order to kick off the rule induction process. We expect the final learned rule set to be smaller, more optimal and with higher performance as compared to those rules induced from other reported systems on the same domain.

(b) Another problem with some rule induction learning systems for IE is that they perform rule generalization from the order of lexical, syntactic, to semantic level sequentially [Califf and Mooney, 1999]. The main difficulty with fixed order of rule generalization is that current methods often miss good rules that do not have good coverage at the level of lexical level, but may have good coverage at the semantic level. Such rules tend to be discarded early in the rule induction process. This research is concerned with utilizing some global statistical information in the training data to initiate the rule induction process from good starting point and find the appropriate generalization level instead of following the fixed order of generalization [Xiao, Chua and Liu, 2003; Xiao, Chua and Liu, 2004]. In Chapter 4, a supervised covering pattern rule induction algorithm, GRID (Global Rule Induction for text Documents), will be described in detail.

(c) While supervised learning methods need a large amount of manually annotated training instances that are expensive to obtain, there are much research in recent years that focuses on bootstrapping an IE system with a small set of annotated instances or a small set of seed words [Blum and Mitchell, 1998; Collins and Singer, 1999; Agichtein and Gravano, 2000]. Co-training is one such bootstrapping strategy and it begins with a small amount of annotated data and a large amount of un-annotated data. Usually, co-training systems train more than one classifier from the annotated data, use the classifiers to annotate some un-annotated data, train the new classifiers again from all the annotated data, and repeat above process. Co-training with multi-view has been widely applied in natural language learning. It reduces the need for annotated data by exploiting disjoint subsets of features (views) such as contextual view and content view, each of which is sufficient for learning. One of the problems when applying the co-training algorithms for natural language learning from large datasets is the scalability problem. Degradation in the quality of the bootstrapped data arises as an obstacle to further improvement [Pierce and Cardie, 2001]. Thus, in Chapter 6, based on GRID algorithm, a bootstrapping paradigm called GRID_CoTrain which combines co-training with active learning is proposed. Active learning methods attempt to select for annotation and training only the most informative examples and therefore are potentially very useful in natural language applications. In GRID_CoTrain, several active learning strategies in co-training model are investigated.

(d) The best performance in GRID_CoTrain with active learning has to involve a human in the loop to manually annotate some instances or correct some annotation errors. To alleviate the manual labor work, a novel bootstrapping scheme with cascading use of a soft pattern learner (SP) [Cui, Kan and Chua, 2004] and GRID for realizing weakly supervised information extraction is proposed in Chapter 7. The cascaded learners (GRID+SP) can approach the performance of the fully supervised IE system GRID while using much less hand-tagged instances [Xiao, Chua and Cui, 2004]. In our experiments, we also show that GRID+SP performs better than GRID_CoTrain while requiring less human labor.

1.3 Contributions

As discussed earlier, the primary motivations of this thesis involve proposing an effective pattern rule induction algorithm for supervised learning of information extraction tasks and extending it with other machine learning methods to realize weakly supervised information extraction.

Let us summarize this chapter by explicitly stating our major contributions:

(a) We propose GRID, which utilizes the global feature distribution in training corpus to derive better pattern rules for information extraction tasks. GRID examines all the training instances at the representation levels of lexical, syntactic and semantic simultaneously and selects a global optimal feature to start the rule induction process. GRID also makes full use of linguistic resources such as (shallow or full) parsing and named entity recognition. The features used are general and applicable to a wide variety of domains, ranging from semi-structured corpus to free-text corpus (Chapter 4). The experimental results reveal that the pattern rule set learned by GRID is smaller, more optimal and has higher F_1 performance as compared to the set induced by several systems.

(b) GRID is a general learner and it can be applied to new tasks other than information extraction. We apply GRID successfully to definitional question answering task and story segmentation in news videos (Chapter 5).

(c) In order to alleviate the human annotation labor, we extend GRID to a weakly supervised learning paradigm by combining co-training and active learning technologies. GRID_CoTrain is a weakly supervised learner by co-training classifiers in two views: contextual view and content view. By incorporating active learning strategies, GRID_CoTrain can achieve comparable performance by using a much smaller set of seed words as compared to a fully supervised system (Chapter 6).

(d) Finally, we develop another bootstrapping method (GRID+SP) to automatically annotate the unlabeled examples required by the bootstrapping process. This method is implemented by cascading use of a soft pattern learner (SP) and GRID with less human intervention as compared with the active learning strategies in GRID_CoTrain (Chapter 7).

1.4 Organization

The rest of this dissertation is organized as follows. Chapter 2 presents background knowledge on the pattern rule induction method for information extraction and the basic machine learning paradigms for IE, such as supervised learning, weakly supervised learning and active learning. Chapter 3 surveys related information extraction systems using pattern rule induction for information extraction tasks. Chapter 4 describes the representation of GRID and presents the learning method in detail as well as the experimental evaluations. Chapter 5 presents the application of GRID to other two tasks: definitional question answering and story segmentation in news videos. Chapter 6 describes the application of co-training with multi-view to GRID, GRID_CoTrain, and presents the incorporation of co-training with active learning and discusses the experimental evaluation of GRID_CoTrain using active learning. Chapter 7 introduces another alternative bootstrapping paradigm (GRID+SP) for realizing weakly supervised information extraction by combining GRID with a newly proposed soft pattern learner (SP). Finally Chapter 8 summarizes this thesis and suggests avenues for future research.

Chapter 2

Background

In this Chapter, we introduce background knowledge of pattern rule induction method for information extraction and some related machine learning paradigms such as active learning for information extraction.

2.1 Inductive Learning

Inductive learning has received considerable attention in the machine learning community; see [Mitchell, 1997] Chapters 2 and 3 for surveys. At the highest level, inductive learning is the task of computing, from a set of examples of some unknown target concept, a generalization that (in some domain-specific sense) explains the observations. The idea is that a generalization is good if it explains the observed examples and more importantly makes accurate predictions when additional previously unseen examples are encountered.

For example, given an inductive learning system for information extraction for semantic slot of "victim" in terrorism domain, the system is told that "*Mr. Smith* was killed", and "*Ms. Jordan* was killed". The learner might then hypothesize that the general rule underlying the observations is "*Person* was killed → *Person* is victim". This assertion is reasonable, because it is consistent with the examples seen so far. If asked "*Mr. Hosen* is a victim?" with the fact of "*Mr. Hosen* was killed", the learner would then presumably respond "Yes".

We proceed by presenting inductive learning by bottom-up, top-down and combining these two.

2.1.1 Bottom-up inductive learning

Bottom-up inductive learning is to conduct rule induction learning from specific to general, for example, the generalization example in Section 2.1 is bottom-up where we generalize "person" from "Mr. Smith" and "Ms. Jordan". The AQ algorithm [Michalski, 1983] is a typical covering algorithm that generates rules from specific to general. Covering algorithms aims to generate rules that cover all training examples by learning one rule at a time. Each of the learned rules covers part of the training examples. The examples covered by the last learned rule are removed from the training set before subsequent rules are learned. AQ algorithm begins with a set of labeled training instances and builds a disjunctive set of concept descriptions, which taken together cover all the positive instances and none of the negative ones. Each step of AQ algorithm selects a positive instance not yet covered and derives a general concept description from this seed.

CRYSTAL [Soderland, *et al.*, 1995] is the first system to treat the information extraction task as a supervised learning problem in its own right. CRYSTAL is also a covering algorithm, learning rules from specific to general. Rules in CRYSTAL are generalized sentence fragments. The feature set used by CRYSTAL is implicit in its search operators. It consists of literal terms, syntactic relations, and semantic noun classes (these semantic classes are manually designed input to the algorithm). Thus, one generalization step CRYSTAL can take is to replace a literal term constraint with the semantic class to which it belongs. CRYSTAL is a multi-slot extraction algorithm, which extracts multiple distinct field instances in concert. Webfoot is a modification of CRYSTAL for HTML [Soderland, 1997b]. Instead of sentences, Webfoot trains on text fragments that are the result of a heuristic segmentation based on HTML tags.

Details of CRYSTAL's strategy to find appropriate level of generalization are outlined as following:

CRYSTAL begins by randomly selecting a positive instance of target concept as a seed. It then takes the most specific concept definition that covers this instance and generalizes it. Intuitively, the generalization could be performed by dropping the constraints from the specific concepts gradually. Each proposed generalization is tested on the training set to ensure that the proportion of negative instances does not exceed a user-specified error tolerance. The most general definition within error tolerance is added to the rule base and another seed is selected from positive instance not yet covered by the rule base. This is repeated until all positive instances have been covered or have been selected as seed instances. One problem of generalization is that there are many combinations of term constraints when relaxing the constraints from specific instances. For example, given an instance of "Jack Harper, a company founder", there are 5 term constraints (Jack Harper is treated as one term; comma is also treated as one term). There are 32 (2^5) possible ways to relax this constraint by relaxing a subset of the terms. There are also four (2^2) possible relaxations of the two-word head terms constraints and eight (2^3) for the three-word modifier terms constraints. There are so many possibilities of generalizations for such a simple example. For some initial seed concepts, there are more than one billion ways to generalize it [Soderland, 1997a]. To solve this problem, the key insight of CRYSTAL is to guide the relaxation process by finding the most similar initial concept definition. CRYSTAL performs the proposed generalization by dropping constraints that are not shared by similar definitions. This is equivalent to relaxing constraints just enough to cover the most similar positive instance, since each initial concept definition corresponds to a positive training instance.

RAPIER [Califf, 1998] is another bottom-up IE learner designed to handle informal texts, such as those found in Usenet job postings. Each rule in RAPIER has three parts: a pre-filler pattern that must match the text immediately preceding the filler; a filler pattern that must match the actual slot filler; and a post-filler pattern that must match the text immediately following the filler. First, for each filler slot, most specific patterns are created for each example, specifying word and tag for the filler and its complete context. Given this maximally specific rule-base, RAPIER attempts to compress and generalize the rules for each slot. New rules are created by selecting pairs of existing rules and generalized rules from the pairs. To avoid the extremely large search space of rule generalization, RAPIER starts by computing the generalizations of the filler patterns of each rule pair and creates rules from those generalizations. RAPIER maintains a list of the best *n* rules created and specializes in the rules under consideration by adding pieces of the generalizations of the pre- and post-filler patterns of the seed rules, working outward from the fillers. The rules are ordered using an information value metric [Quinlan, 1990] weighted by the size of the rule (preferring smaller rules). When the best rule under consideration produces no negative examples, specialization ceases; that rule is added to the rule base, and all rules empirically subsumed by it are removed. Note that RAPIER is a compression algorithm not a covering algorithm.

$(LP)^2$ [Ciravegna, 2001] is the most recent covering bottom-up covering algorithm for information extraction tasks. Different from usual pattern rule induction systems, $(LP)^2$ is tag-based learning instead of slot-based, *i.e.* the rules in $(LP)^2$ are to insert one side of tag into the test texts. For example, to extract a semantic slot of "starting time (*stime*)" from a seminar announcement, $(LP)^2$ may have two sets of rules, one for inserting the tag "<*stime*>" to the texts, the other is to insert the other half tag "</*stime*>" to texts. Training in $(LP)^2$ is performed in two steps: initially a set of tagging rules is learned; then additional rules are induced to correct mistakes and imprecision in tagging. Rule induction is performed from specific to general in the training corpus. Generalization consists in the production of a set of rules derived by relaxing constraints in the initial specific rule pattern. Conditions are relaxed both by reducing the pattern in length and by substituting constraints on words with constraints on some parts of the additional knowledge such as the pre-defined dictionary (or gazetteer). Each generalization is tested on the training corpus and an accuracy score *L=wrong/matched* is calculated. For each initial instance, $(LP)^2$ keeps the *k* best generalizations that have better accuracy, or cover more positive examples, or cover different parts of input, or have an error rate that is less than a specified threshold. The other generalizations are discarded.

2.1.2 Top-down inductive learning

FOIL [Quinlan, 1990] is a prototypical example of a top-down covering inductive logic programming algorithm. It learns a function-free, first-order, Horn-clause definition of a target predicate in terms of itself and other background predicates. FOIL learns the rules one clause at a time using a greedy covering algorithm. The clause finding step is implemented by a general-to-specific hill-climbing search that adds antecedents to the developing clause one at a time. At each step, it evaluates possible literals that might be added and selects one that maximizes an information gain heuristic. The algorithm maintains a set of tuples that satisfy the current clause and includes bindings for any new variables introduced in the body.

WHISK [Soderland, 1999] is a top-down rule induction algorithm for information extraction tasks. WHISK is designed to handle text styles ranging from highly structured to free text, including text that is neither rigidly formatted nor composed of grammatical sentences. WHISK induces rules top-down, first finding the most general rule that covers the seed, then extending the rule by adding terms one at a time. The seed instance is randomly selected from the training instance pool. The metric used to select a new term is the *Laplacian* expected error of the rule, *i.e.* the number of errors plus 1 among those extractions by this rule divided by the total number of extractions plus 1. WHISK grows a rule from a seed tagged instance by starting with an empty rule and anchoring the extraction boundaries one slot at a time. To anchor an extraction, WHISK considers a rule with terms added just within the extraction boundary (base rule 1) and a rule with terms added just outside the extraction boundary (base rule 2). In case that these base rules are not constrained enough to make any correct extractions, more terms are added until the rule at least covers the seed. The base rule is selected that covers the greatest number of positive instances among the hand-tagged training set. The best rule is selected from base rules whose *Laplacian* measure is less than the threshold value. WHISK performs a form of hill climbing and cannot guarantee that the rule it grows are optimal, where optimal is defined as having the lowest *Laplacian* expected error on the hand-tagged training instances.

2.1.3 Combining top-down and bottom-up learning

CHILLIN [Zelle, Mooney and Konvisser, 1994] is an example of an ILP algorithm that combines elements of both top-down and bottom-up induction techniques. CHILLIN's input consists of sets of ground facts representing positive and negative examples, and a set of background predicates defined by definite clauses. Basically, CHILLIN tries to construct a small, simple theory covering the positive, but not the negative examples by repeatedly compacting its current version of the program. Compactness is measured as the syntactic size of the theory.

The algorithm starts with a most specific theory, namely the set of all positive examples. Then it generalises the current theory, aiming to find a generalization which allows to remove a maximum number of clauses from the theory while all positive examples remain provable. The generalization algorithm finds a random sampling of pairs of clauses in the current program. These pairs are generalized by constructing their least-general-generalizations. If a generalization covers negative examples, it is specialised by adding antecedents using a FOIL-like algorithm. If the specialization with background predicates is not sufficient for preventing negative examples from being covered, CHILLIN tries to invent new predicates for further specialization of the clause. At each step, CHILLIN considers a number of possible generalizations and implements the one that best compresses the theory. CHILLIN is able to learn recursive predicates. It avoids generating theories leading to endless recursion by imposing syntactic restrictions on recursive predicates. However, CHILLIN may learn recursive predicates covering negative examples.

PROGOL [Muggleton, 1995] also combines bottom-up and top-down search and is a covering algorithm. As in the propositional rule learner AQ, individual clause construction begins by selecting a random seed example. Using mode declarations provided for both the background predicates and the predicate being learned, PROGOL constructs a most specific clause for that random seed example, called the *bottom* clause. The mode declarations specify for each argument of each predicate both the argument's type and whether it should be a constant, a variable bound before the predicate is called, or a variable bound by the predicate. Given the bottom clause, PROGOL employs an A^*-like search through the set of clauses containing up to *k* literals from the bottom clause in order to find the simplest consistent generalization to add to the definition. Advantages of PROGOL are that the constraints on the search make it fairly efficient, especially on some types of tasks for which top-down approaches are particularly inefficient, and that its search is guaranteed to find the simplest consistent generalization if such a clause exists with no more than *k* literals. The primary problems with the system are its need for mode declarations and the fact that too small a *k* may prevent PROGOL from learning correct clauses while too large a *k* may allow the search to explode.

2.2 Learning Methods

This section presents a taxonomy of related machine learning methods for learning pattern rules for information extraction.

2.2.1 Supervised learning for IE

Any situation in which both inputs and outputs of a component of a learning agent can be perceived is called supervised learning. Often, the outputs are provided by a friendly teacher [Russell and Norvig, 2003]. In information extraction tasks, supervised learning methods use labeled or annotated examples for training the learning agents and test them on the remaining unseen examples. The IE systems we mentioned earlier such as CRYSTAL, RAPIER, $(LP)^2$, WHISK are all supervised learning systems. Since annotation is particularly time-consuming, it is not feasible for users to annotate large numbers of documents. However, un-annotated data is fairly plentiful. Thus IE researchers have investigated active learning techniques to automatically identify documents for the user to annotate. In recent years, there are more and more researches that focus on realizing weakly supervised learning with the help of active learning for information extraction.

2.2.2 Active learning

Active learning explores methods that, rather than relying on a benevolent teacher or random sampling, actively participate in the collection of training examples. The primary goal of active learning is to reduce the number of supervised training examples needed to achieve a given level of performance. Active learning systems may construct their own examples, request certain types of examples, or determine which of a set of unsupervised examples are most usefully labeled [Thompson, Califf and Mooney, 1999].

Active learning or selective sampling [Cohn, Atlas and Ladner, 1994] is discussed in this thesis. In this case, learning begins with a small pool of annotated examples and a large pool of un-annotated examples, and the learner attempts to choose the most informative additional examples for annotation. Results on a number of natural language learning tasks have demonstrated that this kind of selective sampling of active learning is effective in reducing the need for labeled examples [Thompson, Califf and Mooney, 1999]. There are two basic approaches to accomplish this task: certainty-based methods [Lewis and Catlett, 1994] and committee-based methods [Freund, *et al.*, 1997].

In the certainty-based paradigm, a system is trained on a small number of annotated examples to learn an initial classifier. Next, the system examines un-annotated examples, and attaches certainties to the predicted annotation of those examples. The k examples with the lowest certainties are then presented to the user for annotation and retraining. Many methods for attaching certainties have been used [Lewis and Catlett, 1994; Thelen and Riloff, 2002] and they typically attempt to estimate the probability that a classifier consistent with the previous training data will classify a new example correctly.

In the committee-based paradigm, a diverse committee of classifiers is created, from a small number of annotated examples. Each committee member attempts to label additional examples. The examples whose annotations result in the most disagreement amongst the committee members are presented to the user for annotation and retraining. A diverse committee, consistent with the previous training data, will produce the highest disagreement on examples whose label is most uncertain with respect to the possible classifiers that could be obtained by training on that data.

For example, [Thompson, Califf and Mooney, 1999] proposed an active learning strategy, RAPIER+Active, for information extraction. RAPIER+Active is a certainty-based sample selection method. The certainty of an individual extraction rule is based on its coverage of the training data: $pos - 5 * neg$, where pos is the number of correct fillers generated by the rule and *neg* is the number of incorrect ones. Given this notion of rule certainty, RAPIER+Active determines the certainty of a filled slot for an example being evaluated for annotation certainty. Once the confidence of each slot has been determined, the confidence of an example is found by summing the confidence of all slots. RAPIER+Active then performs the certainty-based method of selective sampling. The experimental results show that RAPIER+Active outperforms the fully supervised version of RAPIER with about half of the annotated training examples in RAPIER.

2.2.3 Weakly supervised learning by co-training

Co-training [Blum and Mitchell, 1998] is a weakly supervised paradigm that learns a task from a small set of labeled data and a large pool of unlabeled data using separate, but redundant views of the data (*i.e.* using disjoint feature subsets to represent the data). To ensure provable performance guarantees, the co-training algorithm assumes that the views satisfy two fairly strict conditions. First, each view must be sufficient for learning the target concept. Second, the views must be conditionally independent to each other given the class. Co-training has been applied successfully to natural language processing tasks that have a natural view factorization, such as web page classification [Blum and Mitchell, 1998] and named entity classification [Collins and Singer, 1999].

In [Collins and Singer, 1999], the authors proposed a co-training algorithm for named entity classification using two views: one is called contextual view and the other is content view. Contextual view considers words surrounding the string in the sentence in which it appears (an example of a contextual rule is that it states that any proper name modified by an appositive whose head is president is a person). Content view describes the actual item to be extracted. It might be a simple look-up for the string (an example of a rule is "*Honduras* is a location") or a rule that looks at words within a string (an example of such a rule is that any string containing *Mr.* is a person). The key to using co-training with multi-view for named entity recognition is the redundancy of the unlabeled data. In many cases, inspection of either the content or context information alone is sufficient to classify an example. For example, in "…, says Mr. Cooper, a vice president of …", both a content feature (that the string contains *Mr.*) and a contextual feature (that *president* modifies the string) are strong indications that *Mr. Cooper* is an entity of type *Person*. Even if an example like this is not labeled, it can be interpreted as a "hint" that *Mr.* and *president* imply the same category. This idiosyncrasy enables the co-training of two classifiers (one is contextual rule, the other is content rule) using a small set of seed rules and a large set of unlabeled data for named entity recognition. The authors presented a typical co-training algorithm (DL_CoTrain) with contextual and content rules using decision list for named entity classification as follows:

(a) Given a small set of hand-crafted initial seed rules, such as "full-string=New York→Location".

(b) Set the content decision list equal to the set of seed rules.

(c) Label the training set using the current set of content rules. Examples where no rule applies are left unlabeled.

(d) Use the labeled examples to induce a decision list of contextual rules. The detail of learning a decision list is described in [Yarowsky, 1995].

(e) Label the training set using the current set of contextual rules. Examples where no rule applies are left unlabeled.

(f) On this new labeled set, select *k* content rules. Set the content rules to be the seed set plus the rules selected.

(g) If the number of rules is less than the pre-specified number, return to step (c). Otherwise, label the training data with the combined content/contextual decision list, then induce a final decision list from the labeled examples where all rules are added to the decision list.

2.3 Summary

Inductive learning is well-studied for analyzing and building systems that improve over time or performing generalization from the training examples. The framework provides a rich variety of analytical techniques and algorithmic ideas.

In this Chapter, we showed the background of basic rule induction methods for information extraction tasks, and also discussed some basic machine learning paradigms for information extraction. In the next Chapter, we will introduce more information extraction systems using the pattern rule induction methods.

Chapter 3

Related Work

Pattern rule induction is widely applied in information extraction research. A key component of an IE system is its set of pattern extraction rules that is used to extract from each document the information relevant to a particular extraction task. As manually constructing useful pattern rules needs a linguistic expert who is familiar with the IE system and the formalism for expressing rules for that system, a number of research efforts in recent years have focused on learning the pattern extraction rules from training examples provided by the common user. In this Chapter, we review several IE systems based on pattern rule induction techniques. We begin by analyzing pattern rule induction systems designed for free text documents, followed by those designed to handle the more structured types of online documents. Lastly, we introduce the wrapper induction systems which are designed to extract and integrate data from multiple Web-based sources. For each system, we focus on the following 5 aspects: (a) working domain; (b) pattern rule representation; (c) extraction granularity; (d) syntactic or semantic constraints; and (e) generalization and/or specialization approaches.

3.1 Information Extraction Systems for Free Text

In this section, we review pattern rule induction systems designed to process documents that contain grammatical, plain text. Their pattern extraction rules are based on syntactic and semantic constraints that help identify the relevant information within a document. Consequently, in order to apply the pattern extraction rules, one has to pre-process the original text with a syntactic analyzer and a semantic tagger. A typical processing of learning pattern extraction rules for free texts is described as following:

Sentence Splitting → Tokenization → Training Instances Selection → PoS Tagging → Named Entity Extraction → Parsing (shallow/full) → Pattern Rule Induction

Basically, the pattern extraction rules in IE are categorized into two types: single-slot rules and multi-slot rules. In some cases, the target is uniquely identifiable (single-slot rules), while in other cases, the targets are linked together in multi-slot association frames. Multi-slot rules can extract the multi-target simultaneously.

(1) AutoSlog/AutoSlog-TS [Riloff, 1993; Riloff, 1996]

AutoSlog generates extraction patterns using annotated tests and a set of heuristic linguistic patterns. AutoSlog-TS is based on the AutoSlog system and eliminates its dependency on annotated texts and only requires the pre-classified texts as input.

- Working Domain: Terrorism attacks in MUC-4 [MUC-4 proceedings, 1992];
- Pattern Rule Representation: (only single-slot rules)

  Patterns are represented as concept nodes. Given a sentence of "The Parliament was bombed by the guerrillas", the concept node is represented as:

  Name: target-subject-passive-verb-bombed

  Trigger: bombed (the trigger words could be verbs or nouns)

Variable Slots: (target (*S* 1)) --- S denotes for "subject"

Constraints: (class phys-target *S*)

Constant Slots: (type bombing)

Enabling Conditions: ((passive))

Below are some of the pre-defined linguistic patterns used by AutoSlog:

<subject> passive-verb; *e.g.* <victim> was murdered

<subject> active-verb; *e.g.* <perpetrator> bombed

<subject> verb infinitive; *e.g.* <victim> attempt to kill

- Extraction Granularity:

  The granularity of extraction in AutoSlog/AutoSlog-TS is the syntactic field that contains the target phrase, such as subject, object *etc.*.

- Syntactic/Semantic Constraints:

  AutoSlog/AutoSlog-TS utilizes syntactic constraints such as the subject, object *etc.* obtained from parsing the sentences.

- Generalization/Specialization Approach

  No obvious generalization or specialization scheme is applied.

(2) CRYSTAL [Soderland, *et. al.*, 1995]

CRYSTAL is an IE system that automatically induces a dictionary of "concept-node definitions" that are sufficient to identify relevant information from a training corpus. Each of these concept-node definitions is generalized as far as possible without producing errors, so that a minimum number of dictionary entries cover all of the positive training instances.

- Working Domain: Hospital discharge reports;

- Pattern Rule Representation: (both multi-slot and single-slot rules)

  Given a sentence of "The patient denies any episodes of nausea", the concept node by CRYSTAL is represented as following:

  Concept Node Type: Sign or Symptom

  Subtype: Absent

  Extract from: Direct Object

  Active Voice Verb: deny

  Subject Constraints: words include "patient"; head class: <patient or disabled group>

  Verb Constraints: words include "denies"

  Direct Object Constraints: head class <sign or symptom>

- Extraction Granularity:

  The granularity of extraction in CRYSTAL is the syntactic field that contains the target phrase. Both exact word and semantic class are used.

- Semantic/Syntactic Constraints:

  Syntactic analysis, semantic lexicon and semantic hierarchy are employed.

- Generalization/Specialization:

  CRYSTAL employs a semantic hierarchy to perform generalization/specialization. Unifying two class constraints may involve moving up the semantic hierarchy to find a common ancestor of classes in the two constraints. Class constraints are dropped entirely when they reach the root of the semantic hierarchy. If a constraint on a particular syntactic buffer is missing from one of the two definitions, that constraint is dropped from the unified constraints.

(3) LIEP [Huffman, 1995]

LIEP is another IE system that can learn dictionaries of extraction patterns directly from the annotated user-provided examples of texts and extract the events. It learns patterns that recognize relationships between key constituents based on local syntax.

- Working Domain: newswire articles about business management change
- Pattern Rule Representation: (multi-slot rules only)

  Pattern rules are represented as forms of paths through a finite-state machine. Given a sentence of "Bob Smith was named CEO by Foo. Inc.", the LIEP pattern is described as following:

  n_was_named_t_by_c:

  noun-group(PNG, head(isa(person-name))), "Bob Smith"

  noun-group(TNG, head(isa(title))), "CEO"

  noun-group(CNG, head(isa(company-name))), "Foo. Inc."

  verb-group(VG, type(passive), head(named or elected or appointed)), "named"

  preposition(PREP, head(of or at or by)), "by"

  Subject(PNG, VG),

  object(VG, TNG),

  post_nominal_prep(TNG, PREP),

  prep_object(PREP, CNG)

  ➔ management_appointment(M, person(PNG), title(TNG), company(CNG)).

- Extraction Granularity:

  Entities are generally expressed by noun phrases. Noun groups are divided into PNG (person), TNG (title) and CNG (company). LIEP identifies the extracted phrase of interest.

- Syntactic/Semantic constraints:

  LIEP uses both the syntactic and semantic information. Syntactic information such as subject, object *etc.* is obtained from a parser. Noun groups are divided into semantic groups such as persons, companies *etc.*.

- Generalization/Specialization

  LIEP employs pattern generalization when the later training examples have the same syntactic relationships as a previously learned pattern, but with different constituent head words or properties. For example, if two training examples have the same syntactic structures but with different verb phrase head word; one is "named", the other is "appointed"; LIEP will generate a "*genclass*" for the learned pattern. The "*genclass*" is equal to "named, appointed". Once such a generalized pattern is formed, LIEP tests it by computing its F-measure and comparing it to the F-measure of the original pattern. If the generalized pattern's F-measure is better, it is added and the old pattern is thrown away; otherwise the generalization is thrown away, and a fully new pattern rule is learned from later training examples.

(4) PALKA [Kim and Moldovan, 1995]

PALKA (Parallel Automatic Linguistic Knowledge Acquisition) system automatically acquires linguistic patterns from a set of domain-specific training texts and desired outputs. Patterns are constructed in the form of FP-structures (Frame-Phrasal patterns) from training texts, and the acquired patterns are tuned further through the generalization of semantic constraints.

- Working Domain: Terrorism attacks in MUC-4

- Pattern Rule Representation: (both single-slot and multi-slot rules)

  Pattern rules in PALKA are represented as FP-structure, where FP-structure = MeaningFrame + PhrasalPattern. A meaning frame is a pre-defined information type. FP-structure can express exact word constraints only on verbs.

  Given a sentence of "The Parliament was bombed by guerrillas.", the FP-structure is described as follows:

  ```
  MeaningFrame: (BOMBING  agent:       ANIMATE
                          target:      PHYS-OBJ
                          instrument:  PHYS-OBJ
                          effect:      STATE)
  PhrasalPattern: ((PHYS-OBJ) was bombed by (PERP))
  FP-structure:
  (BOMBING    target: PHYS-OBJ
              agent:  PERP
              pattern: ((target) was bombed by (agent))
  ```

- Extraction Granularity:

  The syntactic field that contains the target phrase.

- Syntactic/Semantic Constraints:

  Syntactic information is obtained by a parser and semantic information is obtained by a pre-defined semantic concept hierarchy.

- Generalization/Specialization:

  Both generalization and specialization approaches are employed in PALKA. When a positive example is encountered, the semantic constraint is generalized, and when a negative example is encountered, the semantic constraint is specialized. The generalization is performed by moving up the semantic concept hierarchy tree and the specialization is performed by going down the tree.

(5) HASTEN [Krupka, 1995]

The extraction patterns generated by HASTEN are called *Egraphs*, and they can be seen as lists of (*SemanticLabel, StructuralElement*) pairs. HASTEN uses a similarity metric to compare an *Egraph* with the input text.

- Working Domain: Management succession of MUC-6.
- Pattern Rule Representation: (both single-slot and multi-slot rules)

  Pattern rules in HASTEN are represented as *Egraphs*. Given a sentence of "The Parliament was bombed by guerillas.", the *Egraph* is represented as following:

```
BOMBING:
      TARGET:        NP "semantic = physical-object"
      ANCHOR:        VG "root = bomb"
      PERPETRATOR:   NP "semantic = terrorist-group"
```

  "TARGET, ANCHOR, PERPETRATOR" are *SemanticLabels*, and the values following the labels are the *StructuralElement*. HASTEN uses a similarity metric to compare an *Egraph* with the input text. In the first step, the system matches the structural elements and binds the semantic labels of the successfully matched structural elements. It then uses a set of fixed weight factors to compute the percentage of the matched *Egraph*, and it compares the final score with a pre-defined threshold value.

- Extraction Granularity:

HASTEN identifies the exact phrase of interest.

- Syntactic/Semantic Constraints:

  Both syntactic structural and semantic classes are used in HASTEN.

- Generalization/Specialization:

  No obvious generalization/specialization approaches are discussed in the reference paper.

All the systems discussed above used induced pattern rules to extract relevant data from grammatical, free text. Even though all of them use syntactic and semantic constraints to identify the items of interest, there are several important differences among them. First, the granularity of the extraction is different: LIEP and HASTEN identify the exact phrase of interest, while AutoSlog/AutoSlog-TS, PALKA, and CRYSTAL determine only the syntactic field that contains the target phrase. Second, except for CRYSTAL, all of the other systems allow semantic constraints only on the slots to be extracted. Lastly, PALKA, CRYSTAL, and HASTEN can generate both single- and multi-slot rules, while AutoSlog/AutoSlog-TS learns only single-slot rules and LIEP can only induce multi-slot rules.

3.2 Information Extraction from Semi-structured Documents

With the expansion of the Web, users can access collections of documents that consist of a mixture of grammatical, telegraphic and/or ungrammatical text. Semi-structured text has the characteristics that the patterns of occurrences are quite repeatable and the syntactic cues are quite minimal. Typical semi-structured texts include: rental ads, job postings, product pages *etc.*. Development of systems to perform IE tasks on corpora of such semi-structured texts has immediate practical application. However, the IE techniques developed for free texts cannot be applied directly to semi-structured documents since semi-structured texts have less linguistic patterns. Here we discuss four representative systems designed to handle semi-structured documents. The systems extract pattern rules that combine syntactic/semantic constraints with delimiters that "bound" the text to be extracted. Although it seems useful to extract pattern rules that take advantage of repeated HTML tags, the four systems do not utilize constraints of HTML tags.

(1) WHISK [Soderland, 1999]

WHISK is designed to handle text styles ranging from highly structured to free text, including text that is neither rigidly formatted nor composed of grammatical sentences. When used in conjunction with a syntactic analyzer and semantic tagging, WHISK can also handle extraction from free text such as news stories.

- Working Domains: Rental advertisement and management succession (MUC-6)
- Pattern Rule Representation: (both single-slot and multi-slot rules)

  Pattern rules in WHISK are represented as Perl-like regular expressions. For example, given a sentence of "Mr. A succeeds Mr. B, Chairman of XYZ Inc.", the pattern rule in WHISK is described as following:

  Pattern: * (person) * succeeds * (person) * (corp)

  Output: Succession {PersonIn $1} {PersonOut $2} {Org $3}
- Extraction Granularity:

  WHISK rules specify exact delimiters on the target phrase, *i.e.* the extract phrase of interest.
- Syntactic/Semantic Constraints:

WHISK does not need to perform syntactic analysis for handling semi-structured texts. For free text, syntactic information is obtained through parsing result. Pre-defined semantic classes are employed in both semi-structured and free texts. For example, the semantic class of "Bedroom" is defined as following:

Bedroom ::= (br || brs || bdrm || bedrooms || bedroom)

- Generalization/Specialization:

  WHISK induces pattern rules top-down, first finding the most general rule that covers the seed, then extending the rule by adding terms one at a time. The metric used to select a new term is the *Laplacian* expected error of the rule. The WHISK rules could be generalized by substituting predefined semantic classes ("Bedroom") for original lexical tokens ("br").

(2) RAPIER [Califf and Mooney, 1997]

RAPIER employs a bottom-up learning algorithm. It incorporates techniques from several inductive logic programming systems and acquires unbounded patterns that include constraints on the words, part-of-speech tags, and semantic classes present in the filler and the surrounding text.

- Working Domains: Job postings and seminar announcements.
- Pattern Rule Representation: (single-slot rule only)

  The RAPIER extraction patterns consist of three distinct slots: the Pre- and Post- "filler patterns" play the role of left and right delimiters, while the "Filler Pattern" describes the structure of the information to be extracted.

  Given an excerpt of a job posting as follows:

AI. C Programmer. 38-44K. Leading AI firm in need of an energetic individual to fill the following position:……

The extracted data are:

Computer-science-job: title: C programmer; salary: 38-44K; area: AI

The pattern rule in RAPIER for extracting the “area” slot is represented as following:

Pre-filler pattern:	word: leading
Filler pattern:	list: len:2
	tags: [nn, nns]
Post-filler pattern:	word: [firm, company]

The pre- and post- filler patterns specify that information to be extracted is immediately preceded by the word “leading” and is immediately followed either by “firm” or by “company”. The “Filler pattern” imposes constraints on the structure of the information to be extracted: it consists of at most two words that were labeled “nn” or “nns” by the Part-of-Speech (PoS) tagger.

- Extraction Granularity:

  As we can see from the above example, RAPIER rules enforce phrase length constraints. The extraction granularity by RAPIER rules is the phrase whose length (the number of words) is less than the rule length constraint.

- Syntactic/Semantic Constraints:

  RAPIER does not perform syntactic analysis for handling the semi-structured texts. It obtains the semantic class information from WordNet [Miller, et. al., 1995] from which the hypernym links are used.

- Generalization/Specialization:

  The generalization scheme in RAPIER is realized from PoS tagging as PoS is more general than word itself and the patterns with PoS taggers can cover more examples.

(3) SRV [Freitag, 1998]

SRV is a top down relational algorithm for information extraction from a class of pages that contain one or more pages devoted to single entities. The SRV system takes the documents to be used for extraction; extracts the individual terms or tokens and classifies them into one of the core features. The features can be simple (mapping token to a value) or relational (token to another token). The system learns over an explicitly provided set of such features. A set of rules are developed from the training set. A part of this training set is used for validation. The SRV is tested on web site pages from four well-known universities.

- Working Domain: University web pages.
- Pattern Rule Representation: (single-slot rules only)

  Pattern extraction rules in SRV are represented as first-order logic extraction patterns that are based on attribute-value tests and the relational structure of the documents.

  Given two instances of "… to purchase 4.5 mln Trilogy shares at …" and "… acquire another 2.4 mln Roach shares …", SRV has a pattern rule to extract the company name as following:

  ```
  Acquisition:- length (<2),
                some (?A [] capitalized true),
                some (?A [next-token] all-lower-case true),
                some (?A [right-AN] wn-word 'stock').
  ```

  The "right-AN" construct refers to the "right AN link" in a link grammar, which connects a noun modifier with the noun it modifies. "wn-word" is the WordNet synset.

- Extraction Granularity:

Similar to RAPIER system, SRV extracts the phrase whose length is less than the length constraint in the pattern rule.

- Syntactic/Semantic Constraints:
  SRV takes advantage of orthographic features, token length and link grammars. Furthermore, it imposes constraints based on the WordNet semantic classes.
- Generalization/Specialization:
  We can consider the semantic class used in SRV is a kind of generalization. Pattern rules with the semantic class can cover more examples.

(4) $(LP)^2$ [Ciravegna, 2001]

$(LP)^2$ is a covering algorithm for adaptive IE from text documents. It induces symbolic rules that insert SGML tag into texts by learning from examples found in a user-defined tagged corpus. In $(LP)^2$, training is performed in two steps: first a set of tagging rules is learned to identify the boundaries of slots; next, additional rules are induced to correct mistakes in the first step of tagging.

- Working Domain: Seminar announcements and Job postings.
- Pattern Rule Representation: (single-slot rules only)
  $(LP)^2$ learns symbolic rules through user tagged training examples. For example, given a user tagged example of "…the seminar at <*stime*> 4 pm will…", (LP)2 learns a rule for inserting <*stime*> to test instance as following:
  at digit *timeid* $\rightarrow$ insert <*stime*> between "at" and digit
  where *timeid* could be "*am*", "*A.M.*" or "*pm*" *etc.*. We can see the rules in $(LP)^2$ is not slot-based but tagged-based. Every rule in $(LP)^2$ inserts either opening tag "< >" or closing tag "< />".
- Extraction Granularity:
  $(LP)^2$ extracts the desired information from texts based on word tokens.
- Syntactic/Semantic Constraints:
  $(LP)^2$ employs the PoS information and the pre-defined semantic classes for rule generalization.
- Generalization/Specialization:
  Generalization in $(LP)^2$ consists in the production of a set of rules derived by relaxing constraints in the initial rule pattern. Conditions are relaxed both by

reducing the pattern in length and by substituting constraints on words with constraints on some parts of the additional knowledge. Each generalization is tested on the training corpus and an accuracy score *L=wrong/matched* is calculated. For each initial instance the *k* best generalizations are kept that: (1) report better accuracy; (2) cover more positive examples; (3) cover different parts of input; (4) have an error rate that is less than a specified threshold. The other generalizations are discarded.

The four types of extraction rules presented above differ in several ways. First, RAPIER, SRV and $(LP)^2$ can generate only single-slot rules while WHISK generates multi-slot rules. Single-slot learners require less training examples than multi-slot learners [Ciravegna, 2000] while single-slot learners need more rules for extracting a multi-target instance than multi-slot learners do. Next, RAPIER, SRV and $(LP)^2$ are capable of imposing a richer set of constraints than WHISK: RAPIER and $(LP)^2$ make use of a part-of-speech tagger, while SRV takes advantage of orthographic features, token's length and link grammars. Furthermore, RAPIER and SRV can impose constraints based on WordNet semantic classes.

3.3 Wrapper Induction Systems

Different from the traditional IE community, the wrapper systems aim to extract and integrate data from multiple structured Web-based texts. Structured text uses markup to represent an ordered hierarchy of content objects. The structure elements describe the structure of text without describing how the content is presented. Examples of such structure markup languages are SGML and XML. They include a schema or document type declaration that defines and restricts the component elements. In this context, the aim of information extraction (IE) techniques is to select pertinent sentences within a text and to extract from these sentences structured facts which can be stored in databases. A typical wrapper application extracts data from Web pages based on predefined HTML templates (*e.g.*, electronic commerce, weather, or restaurant review pages). The wrapper induction systems generate delimiter-based rules that do not use linguistic constraints. Most wrappers utilize the HTML/XML tags as pattern rule constraint elements, such as STALKER [Muslea, Minton and Knoblock, 1999].

(1) HLRT [Kushmerick, Weld and Doorenbos, 1997]

HLRT is the first wrapper induction system and it generates extraction rules similar to those of WHISK, except that it uses only delimiters that immediately precede and follow the actual data.

- Working Domain: Email services and restaurant information.
- Pattern Rule Representation: (multi-slot rules only)
  Pattern rules in HLRT are similar to WHISK which are based on Perl-like regular expressions. Given two excerpts from a restaurant webpage of
  "D1: 1. Joe's: (313)323-5545 2. Li's:(406)545-2020"
  "D2: 1. KFC: 818-224-4000 2. Rome: (656)987-1212";
  HLRT generates the rule as following:
  * '.' (*) ':' * '(' (*) ')'
  Output: Restaurant{Name @1} {AreaCode @2}
  Apparently, the above rule fails on D2 because of the different phone number formatting.
- Extraction Granularity:
  The extractions in HLRT are based on words or tokens.
- Syntactic/Semantic Constraints:
  HLRT does not apply the syntactic or semantic information for wrapper rule induction.
- Generalization/Specialization:
  No generalization/specialization approaches are applied in HLRT.

(2) SoftMealy [Hsu and Dung, 1998]

SoftMealy is a wrapper induction algorithm that generates pattern rules expressed as finite-state transducers. It can be induced from a handful of labeled examples.

- Working Domain: University Computer Science faculty webpages
- Pattern Rule Representation: (multi-slot rules only)

Pattern rules in SoftMealy are expressed through finite-state transducers and contextual rules. Rules allow both the use of semantic classes and disjunctions. Given the same two excerpts as those in HLRT examples from a restaurant webpage of

"D1: 1. Joe's: (313)323-5545 2. Li's:(406)545-2020"

"D2: 1. KFC: 818-224-4000 2. Rome: (656)987-1212";

SoftMealy generates a rule as following:

'.' (*) EITHER ':' (*Nmb*) '_'

OR ':' * '(' (*Nmb*) ')'

Output: Restaurant {Name @1} {AreaCode @2}

SoftMealy's extraction patterns are more expressive than the HLRT ones.

Limitations of both SoftMealy and HLRT consist of their inability to use delimiters that do not immediately precede and follow the relevant items.

- Extraction Granularity:

  The extractions in SoftMealy are based on words or tokens.

- Syntactic/Semantic Constraints:

  Pattern rules allow semantic constraints. No syntactic analysis is applied.

- Generalization/Specialization:

  The generalization algorithm in SoftMealy induces contextual rules by taxonomy tree climbing. The algorithm generalizes each rule element by replacing each token with their least common ancestor with other tokens in the same taxonomy tree. After the generalization, duplicated instances will be removed and the remaining instances constitute the output contextual rules.

(3) STALKER

STALKER is a wrapper induction system which performs hierarchical information extraction. It introduces Embedded Catalog Tree (ECT) formalism to describe the hierarchical organization of the documents. The ECT specifies the output schema for the extraction task, and it is also used to guide the hierarchical information extraction process.

- Working Domain: Email services and restaurant information (the same test sets that are used in HLRT).
- Pattern Rule Representation: (single-slot rules only)

Pattern rules in STALKER are represented as Embeded Catalog Trees (ECT). For example, given a sample document as following:

```
Name: Taco Bell <br> <p> <br>
  -  LA: 400 Pico; (213) 323-5545, (800) 222-1111.
           211 Flower; (213) 424-7654. <p>
  -  Venice: 20 Vernon; (310) 888-1010. <p> <hr>
```

The Embedded Catalog Tree is represented as following:

```
Document := Restaurant LIST(City)
City     := CityName LIST(Location)
Location := Number Street LIST (Phone)
Phone    := AreaCode PhoneNumber
```

where the pattern extraction rules for Restaurant, LIST(City) and CityName are:

```
Restaurant extraction rule:   * 'Name:' (*) '<br>'
LIST(City) extraction rule:   * '<br>' * '<br>' (*) '<hr>'
LIST(City) iteration rule:    * '_' (*) '<p>'
CityName extraction rule:     * (*) ':'
```

Although STALKER can only present single-slot rules, it uses the ECT to group together the individual items that are extracted from the same multi-slot template (*i.e.*, from the same ECT parent).

- Extraction Granularity:

  Extractions in STALKER are based on words or tokens.
- Syntactic/Semantic Constraints:

  STALKER applies the semantic constraints. No syntactic analysis is used.
- Generalization/Specialization:

  No obvious generalization/specialization approaches are applied in STALKER.

STALKER is different from WHISK in two ways. First, even though STALKER uses semantic constraints, it does not enforce any linguistic constraints. Second, the STALKER rules are single-slot which can be grouped together using the ECTs.

In addition to the NLP-based information extraction systems that we have reviewed, there are also modeling-based and ontology-based approaches to Web data extraction [Laender, *et.al.*, 2002b]. Given a target structure for objects of interest, modeling-based systems try to locate in Web pages portions of data that implicitly conform to that structure. The structure is provided according to a set of modeling primitives (e.g. tuples, lists) that conform to an underlying data model. The systems used algorithms similar to those used by the wrapper induction systems to identify objects with the given structure in the target pages. Systems that adopt this approach are NoDoSE [Adelberg, 1998] and DEByE [Laender, *et.al.*, 2002a]. An ontology-based system performs extraction directly on the data. Given a specific domain application, an ontology can be used to locate constants present in the page and to construct objects with them. The most representative ontology-based approach is the one developed by [Embley, *et.al.*, 1999].

3.4 Summary

In this Chapter, we mainly reviewed three categories of NLP-based IE systems: IE systems for free texts, IE systems for semi-structured texts and the wrapper induction systems for structured texts. We analyzed these related systems based on five aspects: their working domains, their pattern rule representations, the extraction granularities, whether to use syntactic and/or semantic constraints and how to generalize and/or specialize the pattern rules. In the next Chapter, we will introduce GRID, a Global Rule Induction approach to text Documents. GRID learns single-slot pattern rules and it works on both semi-structured documents and free texts. GRID extracts information based on the noun phrase boundaries. It applies lexical chaining to generalize pattern rules with the help of WordNet. The semantic constraints in GRID are obtained from a rule-based named entity recognizer. Shallow/full parsing is also used in GRID.

Chapter 4

GRID: Global Rule Induction for text Documents

As discussed earlier, most of the rule induction algorithms for information extraction randomly select a seed instance to start the rule induction or generalization. The search for a good seed often uses only local information and is often arbitrary. Thus it may happen that some false starts are needed to select a good seed in order to learn good quality, high coverage pattern rules, and often such seeds are not found. In this chapter, we introduce GRID (Global Rule Induction for text Documents) which emphasizes the use of global feature distribution in all of the training examples in order to make better decision on pattern rule induction. The main contributions and innovative features of GRID are:

- GRID emphasizes the use of global feature distribution information on the whole set of training examples in order to make better decisions on pattern rule extraction. It examines all training instances at the representation levels of lexical, syntactic and semantic simultaneously and selects a global optimal feature to start the rule induction. The features used by GRID are general and applicable to a wide variety of domains, ranging from semi-structured to free-text corpora.

- GRID adopts chunks (noun or verb phrases) information provided by a shallow parser as units to determine the context of the rules. As chunk is of higher syntactic level than word or token, it provides a more appropriate unit to model context.
- GRID incorporates named entity recognition to provide semantic constraints for pattern rule induction and uses a novel statistics-based lexical chaining method to generalize pattern rules.

4.1 Pre-processing Of Training and Test Documents

GRID is a supervised covering rule induction algorithm that learns from a training corpus where the users have tagged the sentences containing information of specific slot type such as the name of speaker, venue in a seminar announcement *etc.*. For each slot type, the tagged instances of that type are regarded as positive examples, while the remaining sentences in the documents are regarded as negative examples. GRID uses chunking information derived from shallow parsing as the basic granularity of context information. This is to avoid the difficulties in deciding slot boundaries if words were to be used as units for context [Ciravegna, 2001]. Other approaches use higher syntactic units compared with chunks such as subjects and objects as context [Riloff, 1993; Riloff, 1996; Soderland, *et al.*, 1995]. But it is not easy to find a robust parser to obtain the subject and object information.

Before learning may commence, both training and testing documents are pre-processed by the same basic NLP modules such as sentence splitter, tokenization, morphological analysis, shallow parsing and named entities extraction. We use the NLProcessor (a shallow parser) from Infogistics company[1] to perform the syntactic analysis to generate information on Part-of-Speech (PoS), noun group and verb group chunking. For example, given a sentence "A bomb was thrown near the house"; after the shallow parsing by NLProcessor, we will get the result "[A_DT bomb_NN] <was_VBD thrown_VBN> near_IN [the_DT house_NN]" in which the "[ ]"s are noun phrases and "( )"s are verb phrases. DT, NN VBD, VBN and IN are the PoS tags for delimiter, noun, verb past, verb past participle and preposition respectively[2]. We use [ ] or ( ) as the chunk units in our later experiments. We also employ a rule-based named entity recognition module similar to that used in [Chua and Liu, 2002] to derive the semantic classes of some noun groups, such as person, organization, location, and time *etc.*. The named entity recognition module uses rules which are based on both local sentence-level and global context information from the same document. Sometimes we are unable to identity the semantic type of a noun phrase if we consider only local sentence-level context information. For example, in the sentence of "Herminio Para announced a new system", "Herminio Parra" could be a person or an organization name. Thus we need to employ global information from the whole document to resolve the above ambiguity [Chieu and Ng, 2002b]. The types of global information we used are described as follows:

(a) Acronyms: Words made up of all capitalized letters will be stored as acronyms (*e.g. PCD*). The system will then look for sequence of initial capitalized words that match the acronyms found in the whole document. For example, if *PCD* and *Party of Christian Democratization* are both found in the same document and if *Party of Christian Democratization* can be identified as an organization name,

[1] http://www.infogistics.com/textanalysis.html
[2] http://www.infogistics.com/tagset.html

then "*PCD*" will also be identified as an organization name and they are linked to the same entity.

(b) Sequence of Initial Capitals: In the sentence of "*Also Panama Defense Forces is the target of this attack.*", "*Also Panama Defense Forces*" may be identified as an organization name while in fact only "*Panama Defense Forces*" is the organization name. It is unlikely that other occurrences of "*Panama Defense Forces*" in the same document also co-occur with "*Also*". This feature attempts to capture such knowledge. For every sequence of initial capitalized words, its longest substring that also occurs in the same document as a sequence of initial capitals is identified. For this example, "*Panama Defense Forces*" is the longest substring of "*Also Panama Defense Forces*" in the same document. In this case, the named entity recognizer identifies "*Panama Defense Forces*" as the organization name instead of identifying "*Also Panama Defense Forces*" as the organization name.

(c) Initial Capitals of Other Occurrences: The capitalization of the initial letter of a word may be due to its position rather than its meaning (first word of a sentence; in headline). In these cases, the case information of other occurrences of the same word might be used to confirm its type. For example, in the sentence that starts with "*Leon was killed in the attack ...*", because Leon is the first word, the initial capital might be due to its position (as in "*They were killed in the attack…*"). If however somewhere in the same document we see "*Until March 1987 Leon distributed leaflets on…*", then we can be surer that Leon is a person.

(d) Organization Suffixes and Person Prefixes of Other Occurrences: Sometimes we cannot distinguish whether a phrase is a person or an organization name according to the local sentence-level feature. For example, in the sentence "*Herminio Parra announced a new system…*", we do not know whether "*Herminio Parra*" is a person or an organization name. On the other hand, if we found "*Professor Herminio Parra*" in the same document, then "*person*" will be more probable.

In order to find more relevant instances to learn pattern rules, we also employ a pronominal anaphora resolution algorithm[3] [Lappin and Leass, 1994] to substitute some pronouns. For example, if we have "Mr. A is a general in the army. He was killed by terrorists", "He" will be substituted by "Mr. A". Given that "Mr. A" appears in the template answer keys, the instance of "Mr. A was killed by terrorists" will be considered as a positive training instance. Likewise, for test documents, we also do such substitutions in order to find more possible extraction patterns.

4.2 The Context Feature Vector

For every tagged training instance, GRID generates a context feature vector centered around the tagged slot (such as the "starting time" in a seminar announcement) from which to generate the pattern rules. The context feature vector is of the general form:

$$<c_{-k}> \ldots <c_{-2}> <c_{-1}> <c_0> (tagged\_slot) <c_{+1}> <c_{+2}> \ldots <c_{+k}> \quad (4.1)$$

Here $<c_i>$ $\{i=-k \text{ to } +k;\ i \neq 0\}$ represents the context units of the tagged slot, and k is the number of context units considered. $<c_0>$ represents the central tagged slot itself. $<c_i>$ can be a token, a noun or a verb phrase or even a syntactic unit such as subject or object

[3] http://www.comp.nus.edu.sg/~qiul/NLPTools/JavaRAP.html

and it can be of various feature types, including: words, PoS (if it's a single token), various types of verbs and noun chunks, and semantic classes.

One key characteristic of GRID is its representation of context feature vector, in which we code all elements (including both the tagged slot and the context elements) at their appropriate lexical, syntactic and semantic representations simultaneously. The context feature vector for a single tagged instance can therefore be represented as follows:

$$<(-k,f_{-k}^{1}), ..., (-k,f_{-k}^{m}),..., (-1, f_{-1}^{1}), ..., (-1, f_{-1}^{m}), (0,f_{0}^{1}),..., (0,f_{0}^{m}), (1,f_{1}^{1}), ..., (1,f_{1}^{m}), ..., (k,f_{k}^{1}),..., (k,f_{k}^{m})> \quad (4.2)$$

where *m* is the number of linguistic features for each element.

As shown in Equation (4.2), each element is represented as a tuple (g, f_g^i). The first part of the tuple, *i.e.* *g*, indicates the position of the element within the tagged instance. $g=0$ gives the position of tagged slot, and positive *g* (or negative *g*) gives the g^{th} right (or left) hand context element from the tagged slot. If there are *m* features, and *k* context elements, then we have a context vector of size $(2k+1)\times m$.

The second part of the tuple, *i.e.* f_g^i, gives the possible appropriate linguistic representation for each element. The overall feature set consists of 12 (*i.e.* *m*=12) lexical, syntactic and semantic features are given in Table 4.1. The first two representations (Lex. String and PoS) respectively give the original lexical form, and the Part-of-Speech information of the element if it is a single token. The two features are of string type. The next 8 representations (NP_Person, NP_Org., NP_Loc., NP_Date, NP_Time, NP_Perc., NP_Mon., and NP_Num.) cover the general named entities (NE) of type Person, Organization, Location, Date, Time, Percentage, Money and Number ("NP" stands for "Noun Phrase" and "VP" stands for "Verb Phrase"). The first 7 types are the standard named entities defined in MUC [MUC-7, 1998]. Here we added the NP type of "number" to capture all numbers. The last 2 representations (VP_Act. and VP_Pass.) indicate the active and passive voice of VP. The values from the third feature to the twelfth feature are stored as "true" or "false". We also store these representations as string type and for NP and VP, we also store the head noun and root verb for f_g^1, f_g^{11} and f_g^{12}. The set of representations is selected to model all possible patterns used in rule induction. They are selected to capture all essential syntactic and semantic types, and are based partly on related works [Riloff, 1996; Soderland, 1999] that were demonstrated to be effective. These features are not specific to any domain.

Feature	Description	Feature	Description
f_g^1	Lex. String	f_g^2	PoS
f_g^3	NP_Person	f_g^4	NP_Org.
f_g^5	NP_Loc	f_g^6	NP_Date
f_g^7	NP_Time	f_g^8	NP_Perc.
f_g^9	NP_Mon.	f_g^{10}	NP_Num.
f_g^{11}	VP_Pass.	f_g^{12}	VP_Act.

Table 4.1 Features that GRID employed

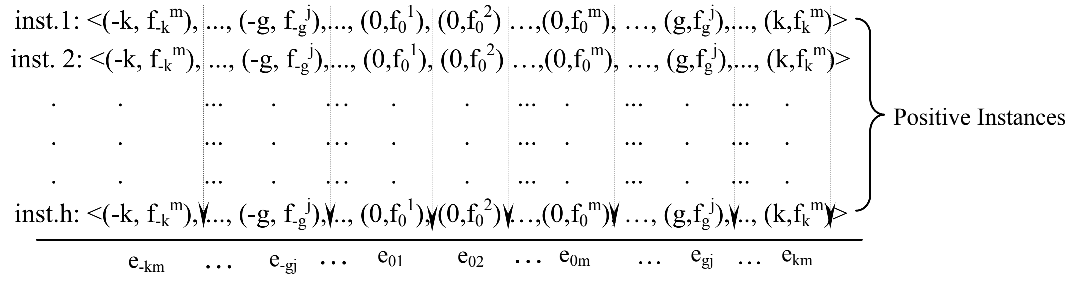
4.3 Global Representation of Training Examples

In order to find a good seed instance to start the rule induction process, GRID utilizes a global approach to finding pattern rules by making full use of the feature statistics in the tagged examples. It does not generalize the rule from one single instance like some rule induction IE systems do, such as $(LP)^2$ [Ciravegna, 2001]. Instead, it incorporates the global information in all positive training examples and selects the most prominent generalized/non-generalized feature to construct the rule. The initial rule generated for each slot type in GRID will thus have the highest coverage in the current active training instance pool.

Given the cluster of training instances of a specific slot type, GRID generates a context feature vector for each instance using Equation (4.2).

By arranging all the instances modeled using Equation (4.2) in the same table, we obtain the global context feature representation for the whole training corpus as shown in Figure 4.1. We align these elements according to their corresponding context positions. Note that not all of the feature representations are present for each element. The occurrences of the common element features at a specific position *g* (*g* is positive number) are cumulated as e_{gi}. From Figure 4.1, we can easily obtain the global distribution frequency of any element feature and at any position, and derive the set of instances covered by any feature set *f*. We consider different features play different importance in various domains, for example, in the free text terrorist attacks corpus, verb features play crucial roles. Thus for each feature, we give it a weight coefficient β_{gi} empirically.

It is important to select a good feature to kick off the rule induction for a covering algorithm. One intuition is to select element feature that has the highest value of $\beta_{gi} \times e_{gi}$ in the active positive training set. By adding this element feature f_g^i into an active feature


inst.1: <(-k, f_{-k}^m), ..., (-g, f_{-g}^j),..., (0,f_0^1), (0,f_0^2) …,(0,f_0^m), …, (g,f_g^j),..., (k,f_k^m)>
inst. 2: <(-k, f_{-k}^m), ..., (-g, f_{-g}^j),..., (0,f_0^1), (0,f_0^2) …,(0,f_0^m), …, (g,f_g^j),..., (k,f_k^m)>
inst.h: <(-k, f_{-k}^m), ..., (-g, f_{-g}^j),..., (0,f_0^1), (0,f_0^2) …,(0,f_0^m), …, (g,f_g^j),..., (k,f_k^m)>
Positive Instances
e_{-km} … e_{-gj} … e_{01} e_{02} … e_{0m} … e_{gj} … e_{km}
Figure 4.1 Global distribution of instances and representations

set *f*, we can generate a pattern $r_c(f)$ in terms of the feature set *f* including current f_g^i so that $r_c(f)$ covers a number of active training instances which have the most prominent feature f_g^i. However, the quality of $r_c(f)$ is determined not only by its coverage in the positive training set but also by the number of instances in the negative set that it covers which would be regarded as errors. Let n_k denote the number of both positive and negative examples covered by the rule $r_c(\underline{f})$, and m_k be the number of negative examples or errors covered by that rule. A good measure of the quality of the rule is the *Laplacian* expected error [Soderland, 1999] defined as:

$$Laplacian(r_c(\underline{f})) = \frac{m_k + 1}{n_k + 1} \tag{4.3}$$

The element feature that has the highest $\beta_{gi} \times e_{gi}$ value does not necessarily lead to a rule with the lowest *Laplacian* measure. On the other hand, it is too costly to evaluate the *Laplacian* measures of all possible element features. As a compromise, we evaluate the *Laplacian* measure of the top *w* element features with high $\beta_{gi} \times e_{gi}$ values in the active positive training set. Our ultimate aim is to select rule that has prominent feature f_g^i with high $\beta_{gi} \times e_{gi}$ value and whose *Laplacian(r* $Laplacian(r_c(\underline{f}))$ satisfies the pre-defined error tolerance value. It is worth noting that adding more features into $\underline{f}$ helps to constrain the rule, and ideally lead to improvement in rule precision.

4.4 The Overall Rule Induction Algorithm

The pattern rules in GRID are represented as follows:

$$\text{constraint}_1\ \text{constraint}_2\ \ldots\ \text{constraint}_i\ \ldots\ \text{constraint}_n \rightarrow \text{insert SGML tags} \tag{4.4}$$

At the left hand side, there are some pattern constraint conditions which could be any feature element in Table (4.1). At the right hand side, there is rule action which is to insert opening tag "< >" and closing tag "< />" for each semantic slot. Usually, the SGML tags are inserted beside noun phrase boundaries. For example, a pattern rule of "*start at NP_Time* → *NP_Time is starting time*" is to insert the tags of "<*stime*>" and "<*/stime*>" beside the boundaries of the noun phrase whose named entity type is "*Time*" when the left side of the pattern rule is matched with the instance.

We now present the overall algorithm for GRID to induce pattern rules as follows:

```
a) Group tagged instances of the same slot type into one cluster.
b) Generate context feature vectors for all positive instances in every cluster. The
   resulting $k^{th}$ cluster is $\underline{C}_k$, with the positive instance set $\underline{P}_k$ and negative instance set $\underline{N}_k$.
   Let $\underline{r}_k$ be the set of rules extracted so far to cover $\underline{P}_k$; and set $\underline{r}_k = null$.
c) For every cluster $\underline{C}_k$, perform the followings:
    ($c_1$) Loop-1:  // to generate new rules
        Let $\underline{f}_c = null$ be the current feature set;
        $r_c(\underline{f}_c)$ be the current rule; and
        $\underline{P}_c$, $\underline{N}_c$ be the set of instances covered by $r_c(\underline{f}_c)$.
        Initially, set: $\underline{P}_c = \underline{P}_k$, $\underline{N}_c = \underline{N}_k$
        RuleAttempt = 0;
    ($c_2$) Loop-2:  // to refine current rule $r_c(\underline{f}_c)$
        Find top $w$ element features $\{f_g^i\}$ (based on $\beta_{gi} \times e_{gi}$ values) that covers at least
        one instance in $\underline{P}_c$;
        Select the $f_i^j$ that minimizes the Laplacian measure of the current rule $r_c(\underline{f}_c \cup f_i^j)$;
        Add $f_i^j$ to $\underline{f}_c$, i.e. $\underline{f}_c = \underline{f}_c \cup f_i^j$
        RuleAttempt++;
    ($c_3$) IF $Laplacian(r_c(\underline{f}_c)) < \sigma$ (error tolerance)
        THEN  // the quality of resulting rule is good
            Add rule $r_c$ to rule set $\underline{r}_k$; or $\underline{r}_k = \underline{r}_k \cup r_c$;
            Update $\underline{P}_k = \underline{P}_k - \{$all instances covered by rule $r_c\}$;
            Go to Loop-1 to generate another rule.
        ELSE  // more work is needed to constraint rule $r_c$
            Update $\underline{P}_c$ by removing those instances that are not covered by $r_c$;
    IF RuleAttempt $\geq \lambda$ (max. rule attempt for constraining rules)
    THEN // relaxing error tolerance;
        Increase $\sigma$;
        Go to Loop-1 to generate new rule with bigger error tolerance;
    ELSE
        Go to Loop-2 to find new feature $f'^j_i$ to refine rule $r_c$.
Repeat until $\underline{P}_k$ is empty.
```

The "*RuleAttempt*" is related to the length of the generated rule which the user could pre-specify. For example, if we set the rule length to "4", then "*RuleAttempt*" could be 8. That is to say, we constrain the rule to the maximum size of 4 contextual units (4 for left side and 4 for right side around the tagged slot respectively). Based on the above algorithm, GRID will generate rules that incorporate the most prominent features. If using a single feature cannot satisfy the error tolerance for quality, then more features will be added to tighten the constraints until the quality of the resulting rule is good enough. GRID is a covering algorithm and each instance in the positive training pool is involved to induce one rule. We can also see that GRID is a local search algorithm. It performs a form of hill climbing and once the rule with current features satisfies the error tolerance it will be output even though adding even more features would result in a lower *Laplacian* value. In case there is noise in the positive training examples, we can apply some "post-pruning" strategies to control the whole quality of the learned rules. For example, after the entire rule set has been generated, some of the rules may have low coverage on the training set. A post-pruning step that discards all rules with *Laplacian* expected error greater than a threshold has the effect of removing the least reliable rules.

During the test phase, we apply the learned GRID pattern rules to unseen test instances that are also preprocessed by the series of NLP modules as we do for training instances (see Section 4.1). When the left side constraints (see Equation (4.4)) are matched with the test instances, then the opening tag "< >" and the closing tag "< />" of a slot will be inserted beside a noun phrase boundaries to indicate a detected entity. However, we observe that sometimes there may be additional tokens or adverbs within the constraints in the unseen instances. In such cases, the left side constraints of the pattern rules will not be matched and the entity will be missed. To overcome this problem, we perform a flexible matching between the learned pattern rules and the test instances. We allow up to one shift in context of new test instances when matching against the learned pattern rules. For example, the rule of "NP VP_active (kill) → NP is *perpetrator*" will match the instance of "FLMN also killed another three persons.", where there is an extra term "also" between NP and VP_active in the "correct" test instance.

We also try the idea of applying edit distance [Sankoff and Kruskal, 1999] to perform inexact matching of rules. However, the substitution scheme used in edit distance is not appropriate for information extraction task. For example, if we allow one element substitution in the pattern rule, the rule of "NP VP_active (kill) → NP is perpetrator" will also match the instance of "NP VP_passive (kill)" since there is only one different element but the NP in the instance is not a "perpetrator" but "victim". Thus we do not use the edit distance to perform partial matching of rules. Instead we found that the simple one shift matching is more effective.

4.5 Rule Generalization

Lexical chaining is a process of placing the individual words into chains of other words of similar meaning. Lexical chaining has been successfully applied in a variety of text retrieval applications, such as text categorization task. In this research, we employ the lexical chaining technology for pattern rule generalization.

At the end of rule induction learning, we extract a set of pattern rules. In general, the rule set obtained is not optimal as it did not consider the lexical and semantic relationships between features used in different rules. For example, for the <*victim*> slot in the terrorism attack domain, we may generate similar pattern rules but with one different slot element of same semantic types, such as the "*murder of* <*victim*>" and the "*assassination of* <*victim*>". As these rules share similar semantic meaning, they should be merged into a more general rule where the root noun is of the semantic class of {*murder, assassination*}. The generalized rule's score is then re-evaluated by the *Laplacian* measure.

Thus we apply a lexical chaining algorithm in terms of corpus statistics to group element features of NP and VP that share the same synsets in WordNet [Miller, *et al.*, 1999] or semantic types. This step helps to generalize the rules by generalizing some verbs or nouns to their semantic classes and improve the rule performance in information extraction. The detailed algorithm of the lexical chaining for verb or noun is as follows:

1) Initialize:
   a) $\underline{W}_S \leftarrow \{(w_1,f_1), (w_2,f_2), \cdots,(w_n,f_n)\}$,

      where f_i is the frequency of word w_i in a given corpus containing n unique words.
   b) Set the output group $\underline{G}_{out} \leftarrow \phi$

2) Generate all possible semantic groups:

   a) For each word w_i in $\underline{\text{W}}_\text{S}$, use WordNet to find its synonyms, *i.e.* Syn(w_i);

   b) Generate all possible groupings of words as:

   $\underline{\text{G}}_{\text{all}} \leftarrow \{(\underline{\text{G}}_1, c_1), (\underline{\text{G}}_2, c_2), \cdots, (\underline{\text{G}}_\text{n}, c_n)\}$

   where $\underline{\text{G}}_\text{i}$ contains all words that have the common synonym set; *i.e.* Syn(w_i) $\cap$ Syn(w_j) $\neq \varnothing$; (the synonyms include the ones in all word senses)

   and $c_i = \sum_{w_k \in \underline{G}_i} f_k$ the prominence measure of $\underline{\text{G}}_\text{i}$.

3) Select the prominent groups as the semantic groups:

   a) From $\underline{\text{G}}_{\text{all}}$, select the group with maximum c_i as:

   $(\underline{\text{G}}_{\text{max}}, c_{max}) \leftarrow \arg\max_{\underline{G}_i \in \underline{G}_{all}} \{c_i\}$

   b) If $c_{max} < \tau$ then terminate.

   c) Else move $\underline{\text{G}}_{\text{max}}$ to $\underline{\text{G}}_{\text{out}}$, ie

   $\underline{\text{G}}_{\text{out}} \leftarrow \underline{\text{G}}_{\text{out}} \text{ U } (\underline{\text{G}}_{\text{max}}, \text{c}_{\text{max}})$, $\underline{\text{G}}_{\text{all}} \leftarrow \underline{\text{G}}_{\text{all}} - \underline{\text{G}}_{\text{max}}$

   d) For each remaining group $\underline{\text{G}}_\text{j}$ in $\underline{\text{G}}_{\text{all}}$, perform the followings:

   $\forall \underline{G}_j \in \underline{G}_{all}$, set $\underline{\text{G}}_\text{j} \leftarrow \underline{\text{G}}_\text{j} - \underline{\text{G}}_{\text{max}}$, $c_j \leftarrow c_j - \sum_{w \in \underline{G}_j \cap \underline{G}_{\text{max}}} c$,

   and if $\underline{\text{G}}_\text{j} \neq \varnothing$, then $\underline{\text{G}}_{\text{all}} \leftarrow \underline{\text{G}}_{\text{all}} - \underline{\text{G}}_\text{j}$

   f) Repeat the process from Step (3a) until $\underline{\text{G}}_{\text{all}}$ = *null*.

We can see that the assumption of this lexical chaining algorithm is that words tend to have one sense per discourse and one sense per collocation [Yarowsky, 1995]. At the end of applying the above lexical chaining algorithm, we obtain a set of semantic groups, each containing a cluster of related words. These semantic groups are used as the basis to generalize the elements in the learned pattern rules.

As the WordNet dictionary is a general lexicon knowledge base for any domain, the semantic group defined using WordNet may have different meanings in different contexts. In order to improve the quality of the lexical chaining process in deriving better semantic groups, we may want to apply more specific domain knowledge such as the ontological knowledge of the terrorist attack domain. The ontology of a problem domain concerns the entities and their relationships in that domain. It includes a vocabulary of terms, definitions and indications of how concepts are inter-related which collectively impose a structure on the domain. Kavalec and Svatek [2002] suggested that it is a promising approach to combine information extraction with ontologies. On the one hand, ontologies can help to improve the quality of information extraction and, on the other hand, the extracted information can in turn be used to improve and extend the ontology. In this thesis, we only consider the first aspect of using ontology knowledge to improve the quality of information extraction. We employ an existing ontology knowledge base from SUMO ontology [www_1] (ontologies of terrorism) and use them as conceptual hierarchy dictionaries for the terrorism domain data of MUC-4.

The ontology of terrorism includes the descriptions of terrorist groups, the types of terrorist attacks, terrorist targets and the ontology of terrorist actions. We built conceptual hierarchy dictionaries based on the ontological knowledge. Figure 4.2 and Figure 4.3 show examples of the hierarchy of terrorist groups and the terrorist attacks edited and presented using Protégé format [www_2].

Many classes in the ontologies have sub-classes. For example, in Figure 4.3, class "*Building*" has several sub-classes such as "*Garage*", "*Hotel*", "*Store*" *etc.*. We may utilize these class hierarchies to generalize our pattern rules. Take an example, if we have "*grocery store*" as an element in a pattern rule; we can expand that element to all of the sub-classes of "*Building*" as a semantic class based on definition in Figure 4.3. In this way, we can extract more relevant and useful instances when applying this generalized rule to other unseen instances.

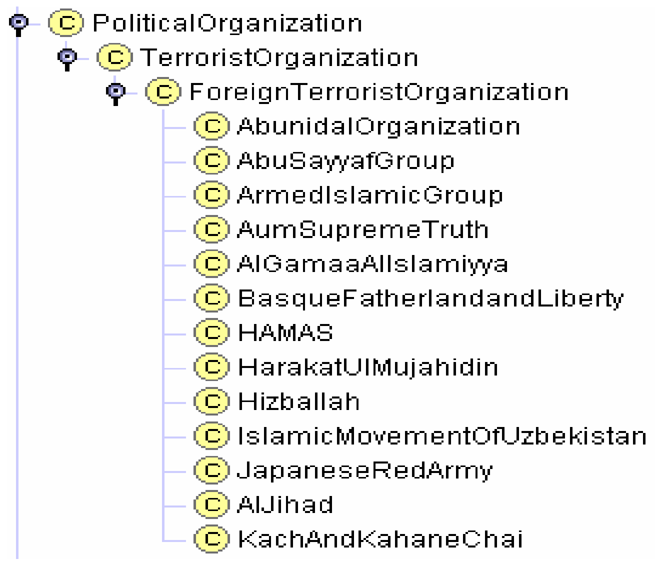


Figure 4.2 Excerpt of the terrorist group ontology

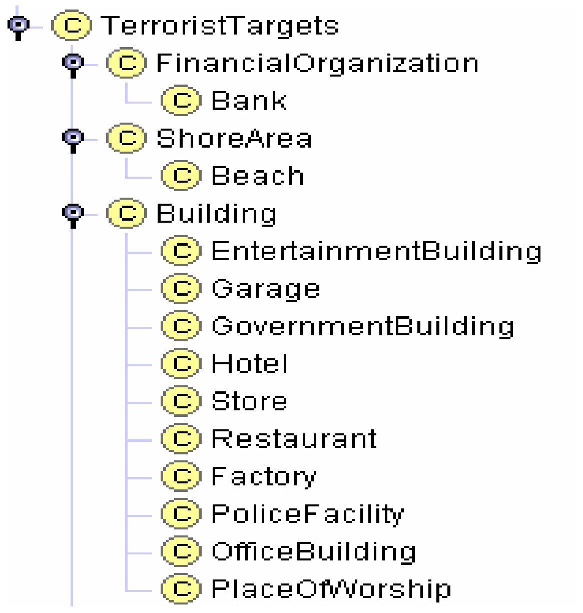


Figure 4.3: Excerpt of the terrorist targets ontology

The domain ontology can also serve as knowledge source to verify the correctness of the rules. In the ontology of terrorist groups, each group is depicted by its long name, short name, alias names, location country, leader, target countries and member number. These facts can be used to verify a rule's correctness. For example, if a noun phrase is classified as "*victim*" while we find that the noun phrase is among the terrorist group list, we will discard that rule and consider it as a wrong classification.

4.6 An Example of GRID Learning

In this section, we present a simple example to illustrate how GRID learns pattern extraction rules. For simplicity, we use an example in a semi-structured domain, and present only a subset of feature representations and context elements. Suppose we want to extract the semantic slot <*stime*>, which indicates the "starting time" in a seminar announcement. Table 4.2 shows the 5 positive instances where the desired slots are tagged. The example instances are selected from the CMU seminar announcement corpus [www_3].

We employ GRID with *w*=1 (*i.e.* start with the most frequent feature). By examining the feature frequency for the context elements at every position around the tagged slot for the 5 positive instance in Table 4.2, we can infer that the tagged slot "NP_Time" (the named entity module can identify this feature) appears most frequently (it occurs 5 times) and thus has the highest coverage in the training example pool. This feature is then being selected, and the generated pattern rule is:

"NP_Time → NP_Time is *starting time*" (4.5)

This rule, however, does not satisfy the *Laplacian* measure as we can see that there are many "NP_Time"s in the corpus that belong to other semantic type such as the "*ending* *time*". So the rule has to be constrained further. Next we examine the context information beside the tagged slot. We see that the token of ":" at the 1st left context position appears 3 times, and is therefore being selected next. The pattern rule is now constrained as:

": NP_Time → NP_Time is *starting time*" (4.6)

context position	-2	-1	0
instance 1	Time	:	<*stime*> 3:30 PM </*stime*>
instance 2	Time	:	<*stime*> 2 p.m. </*stime*>
instance 3	Time	:	<*stime*> 4 p.m </*stime*>
instance 4	start	at	<*stime*> 10 am </*stime*>
instance 5	begin	from	<*stime*> 11:30 AM </*stime*>

Table 4.2 An example for extracting slot <*stime*>

For the CMU seminar announcement corpus, this rule is sufficient to meet our *Laplacian* measure, *i.e.* 0.1, and thus the first rule learned is as following:

": NP_Time → NP_Time is *starting time*" (4.7)

Once we obtain this rule, we remove the first 3 instances which are covered by rule (4.6) from the positive training example pool. We iterate the above process on the remaining two positive examples and finally obtain another two rules as follows:

"start at NP_Time → NP_Time is *starting time*" (4.8)

"begin from NP_Time → NP_Time is *starting time*" (4.9)

During test phase, if any of these learned pattern rule applies, the opening tag <*stime*> and the closing tag </*stime*> will be inserted beside the NP_Time's boundaries. We can see from the above example that the first rule GRID generated covers the most number of positive examples in current active training instances and satisfies the *Laplacian* measure.

4.7 Experimental Results

To verify the generality and effectiveness of GRID, we test GRID on a number of IE tasks including the semi-structured web page corpora, and the free text corpus such as the MUC-4 corpus [MUC-4, 1992]. In each experiment, GRID is trained on a subset of the corpus and the learned rules are tested on the remaining unseen texts, as defined in the respective corpus. The test documents are pre-processed by the same set of NLP tools as described in Section 4.1 and 4.2.

4.7.1 Performance of GRID on free-text corpus

As discussed earlier, GRID is designed to overcome some of the major shortcomings of existing rule-induction-based IE systems. In particular, GRID is designed with the following features:

(a) It takes advantage of the global statistics of all training instances at the representational levels of lexical, syntactic and semantic simultaneously in selecting good starting points to commence pattern rule induction. In particular, it considers top *w* features with high coverage as candidate seeds for rule induction.

(b) It adopts chunk, instead of tokens, as units for the context unit of rules. In free text domain, we also examine the effect of full parsing for GRID.

(c) It employs a named entities recognizer to get the semantic information and it performs lexical chaining to generalize the VPs and NPs that share the same semantics in WordNet and utilizes the specific ontology knowledge as additional dictionary for rule generalization.

We use the free-text MUC-4 corpus to evaluate the effectiveness of each or combinations of the above features. We also present the results of the overall testing on the semi-structured web-based corpus in Section 4.6.2. There are 1,500 training documents and two official test sets, *i.e.* TST3 and TST4, containing 100 documents each. There are 700 training documents which are relevant to terrorism attacks. We train GRID using the 700 documents with relevant templates, and test it on the two official test sets. The output templates are scored using the scorer provided by MUC-4. We perform shallow parsing of the sentences to extract the feature set as listed in Table 4.1 for each context information unit. We perform tests by varying the following parameters: (a) number of top context features, *w*, to be considered in selecting starting seed; (b) the context unit type and size; (c) whether the use of full parsing is effective; and (d) whether lexical chaining based on ontology knowledge for rule generalization is useful. Finally, we compare the performance of our system with other reported systems.

4.7.1.1 Effect of different w

One important parameter in GRID algorithm is *w*, which defines the number of high support features to be considered during the rule induction process. In this section, we present the GRID performance on the MUC-4 terrorism attacks corpus using various *w* values. The task is to extract the perpetrator name, victim, weapon and location *etc.* from the terrorism attack documents according to the template slots defined by MUC-4. The context length *k* (see Equation (4.1)) is set to 4 empirically according to the experiments in the next subsection. We vary the value of *w* from 1 to 8. The results are presented in F_1 measure via MUC-4 scorer. Figure 4.4 presents the average F_1 measure of TST3 and TST4 under various *w* values. It shows that the performance of GRID improves steadily until *w* reaches 3. Thus in the following experiments, we set w equal to 3.

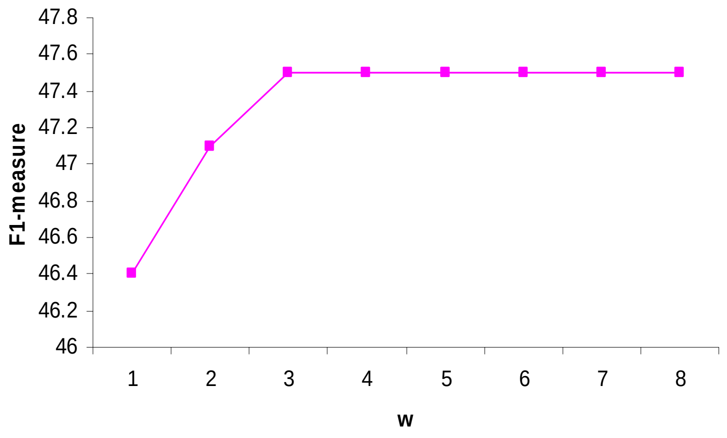


Figure 4.4 Effect of *w* on performance of GRID

4.7.1.2 Effect of the context unit

In order to investigate the effect of context unit using chunks and the performance of different context length, we conduct a series of experiments based on different context length of words or chunks. Chunking information is obtained by a shallow parser from Infogistics company; normally, a chunk could be a noun phrase or verb phrase. Figure 4.5 shows the performance of GRID based on different context length and context unit. We try various context length *k* (left *k* and right *k* word/chunk) from 1 to 5 based on word or chunk unit. From the figure, it is clear that chunk-based system performs better than the word-based ones. The difference is significant for the free text corpus. The experiment shows that when the context length reaches 4, the performance becomes steady. Hence for subsequent experiments, we adopt chunk as the context unit, and set the context length (*k*) to 4.

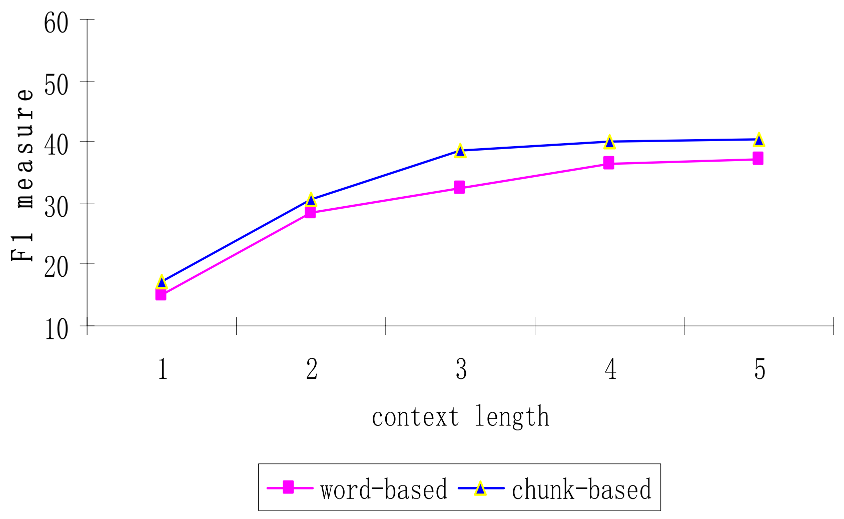


Figure 4.5 Performance of various word/chunk-based context unit on MUC-4 corpus

4.7.1.3 Effect of performing full parsing vs shallow parsing

In this experiment, we employ a full parser [www4] and incorporate four more features to the feature set as listed in Table 4.1. The four features are: NP_Agent, NP_Patient, VAg and Vpa. NP_Agent and NP_Patient stand for agent noun and patient noun phrase respectively. VAg and VPa stand for the associated verbs of agent and patient respectively. For example, we have two sentences: “Members of that security group are combing the area.” and “A bomb was thrown near the house”. In the first sentence, the NP_Agent is “members” and its associated VAg is “comb”; while in the second sentence, the NP_Patient is “bomb” and its associated Vpa is “threw”. These four features can be obtained through traversing the full parsing tree. Table 4.3 compares the performance between systems with shallow parsing and full parsing. The results are based on lexical chaining by WordNet which is the standard approach used in most systems. We can see that for free text documents, full parsing leads to significantly better overall performance.

In particular, with full parsing, the F_1 measure increase significantly from 40.3% to over 47%. However, for semi-structured text such as 'CMU Seminar' and 'Austin Job Listing', full parsing does not produce any improvement in results, This is because for semi-structured corpora, there are less linguistic variations between sentences and thus the use of full parsing does not bring in much more information than shallow parsing.

	MUC-4 (averaged TST3 & TST4)	CMU Seminars	Austin Job Listings
without full parsing (shallow parsing)	40.3	89.2	80.8
with full parsing	47.5	89.3	80.9

Table 4.3 Performance on different domains with and without full parsing

4.7.1.4 Effect of rule generalization with lexical chaining

To illustrate the benefit of rule generalization by performing lexical chaining using the corpus statistics and the synset of WordNet, we conduct experiments to compare the performance difference between systems with and without lexical chaining. We use full parsing for MUC-4 domain. Table 4.4 shows the performance in terms of F_1 measure of GRID on different domains with and without lexical chaining. It is shown that rule generalization by lexical chaining is effective for the free text corpus. One of the reasons is that verb plays an important role in free text corpus, such as MUC-4 corpus. By performing rule generalization, we can obtain more general semantic classes for verbs, thus leading to better rule performance.

The results also indicate that lexical chaining using WordNet plus the domain specific ontology knowledge performs better than using WordNet alone for the MUC-4 task. The ontology dictionary can provide additional knowledge for better rule generalization as we discussed in Section 4.5 and can identify more entities during the test phase. For example, according to the terrorist organizations from the ontology dictionary, we can identify the "perpetrator" organization in a test document even it does not appear in the training examples. The improvement is moderate in this case because the terrorism ontology dictionary we use is general and not tuned for MUC-4. The use of a more specific domain knowledge is expected to perform better.

	MUC-4 (averaged TST3&TST4)	CMU Seminars	Austin Job Listings
without lexical chaining	44.1	87.7	79.5
with lexical chaining using WordNet	47.5	89.3	80.9
lexical chaining by WordNet + ontology	48.5	---	---

Table 4.4 Performance of GRID on different domains with and without lexical chaining

4.7.1.5 Comparison of performance of GRID with other reported systems on MUC-4

We evaluate GRID's performance on MUC-4 corpus in two ways. First is to use the evaluation measures in AutoSlog-TS [Riloff, 1996] and second is to use the standard evaluation method in MUC-4. Although AutoSlog-TS uses an unsupervised approach, which differs from ours, we nevertheless compare ours with it for two reasons. First, its results are openly published as have been shown to perform better than its supervised version called AutoSlog [Riloff, 1993]. Second, its methodology is well documented. We used 1,500 texts (the standard training documents of MUC-4 plus TST1 and TST2 tasks) for training, in which about 50% of the texts are relevant with their associated answer keys given in the MUC-4 corpus. Our target slots are perpetrator, victim and physical target. During testing, we use 100 texts composing 25 relevant texts and 25 irrelevant texts from the TST3 test set, plus 25 relevant texts and 25 irrelevant texts from the TST4 test set. We also perform the same scoring scheme as that in AutoSlog-TS. We score the output by assigning each extracted item to one of the five categories of: correct, missed, mislabeled, duplicate, or spurious. We compute three performance metrics on the test data in terms of: recall, precision and F_1 measure. Table 4.5 and Table 4.6 respectively give the detailed results of GRID and AutoSlog-TS on MUC-4 based on AutoSlog-TS evaluation criteria; while Table 4.7 presents the comparison between GRID and AutoSlog-TS.

Slot	Correct	Missed	Mislabeled	Duplicated	Spurious
Perp.	31	26	3	15	43
Victim	42	23	5	23	40
Target	33	22	15	17	31
Total	106	71	23	55	114

Table 4.5 Results of GRID on MUC-4 corpus

Slot	Correct	Missed	Mislabeled	Duplicated	Spurious
Perp.	30	27	2	12	97
Victim	40	25	7	19	85
Target	32	23	17	16	58
Total	102	75	26	47	240

Table 4.6 Results of AutoSlog-TS on MUC-4 corpus

Slot	AutoSlog-TS			GRID		
	Recall	Precision	F_1	Recall	Precision	F_1
Perp.	53	30	38	54	50	52
Victim	62	39	48	65	59	62
Target	58	39	47	60	52	56
Average	58	36	44	60	54	57

Recall (R) = correct/(correct+missing)

Prec. (P) = (correct+duplicate)/(correct+duplicate+mislabeled+spurious)

*F_1 = 2*P*R/(P+R).*

Table 4.7 Comparison between GRID and AutoSlog-TS

From Table 4.7, we can see that GRID performs much better than AutoSlog-TS. Further analysis of the rule sets generated by both methods reveals that GRID generates much fewer pattern extraction rules than AutoSlog-TS. AutoSlog-TS generated 11,225 rules and after human inspection, retained about 210 rules. In contrast, GRID generated only about 180 pattern rules, and no manual re-evaluation is needed to achieve superior performance. The results are encouraging and indicate that our global approach to rule induction is effective.

We can draw further observations on the quality of the rules generated by GRID system as compared to AutoSlog-TS as follows:

(a) It was found that some pattern rules in AutoSlog-TS are combined as one rule in GRID via lexical chaining. For example, two patterns in AutoSlog-TS: "murder of <NP>" and "assassination of <NP>" are combined as one extraction pattern in which murder and assassination were replaced by a semantic class (synset) on killing action.

(b) Without a ranking scheme and human intervention, GRID generates similar pattern rules as the 25 top-ranked rules reported in AutoSlog-TS system. For example, the 5 top pattern rules generated by GRID for identifying "victim" are as follows (NP denotes "noun phrase" and VP denotes "verb phrase"):

SemanticMurder (including "murder, killing, assassination") of NP → NP is a victim

NP_Person VP_Passive_Past with semantic class "kill" → NP_Person is a victim

NP_Person VP_Passive_Past with semantic class "wound" → NP_Person is a victim

NP_Person VP_Passive_Past with semantic class "kidnap" → NP_Person is a victim

VP_Active_Past with semantic class "kill" NP_Person → NP_Person is a victim

They are similar to those found in the 25 top-ranked rules found in AutoSlog-TS.

TST3	Recall	Precision	F_1	TST4	Recall	Precision	F_1
GE	58	54	56	GE	62	53	57
GE-CMU	48	55	51	GE-CMU	53	53	53
UMASS	45	56	50	*GRID+ontology*	46	51	48
GRID+ontology	46	52	49	SRI	44	51	47
GRID	45	53	49	Alice-ME	46	46	46
Alice-ME	46	51	48	*GRID*	45	47	46
SRI	43	54	48	NYU	46	46	46

Table 4.8 Comparisons of GRID with other systems on the TST3 and TST3 test sets

Table 4.8 shows the performance of GRID on MUC-4 corpus based on MUC-4 evaluations. The standard templates in MUC-4 are more complicated than those in AutoSlog systems. There are string slots, text conversion slots and set fill slots. For text conversion slots and set fill slots, they can be inferred from strings in the documents [Chieu, Ng and Lee, 2003]. All experiments of GRID are based on full parsing and the parameters described earlier. From Table 4.8 we can see that GRID can achieve the performance of the state-of-the-art machine learning system called ALICE [Chieu, Ng and Lee, 2003]. Furthermore, the use of domain-specific ontology (GRID+ontology) can further improve the performance of GRID by about 1%. Notably, the top performing systems listed in Table 4.8 are "GE" and "GE-CMU". However, both systems are "manual" systems that involved "10½ person months" manual efforts on MUC-4 evaluations using the GE NLTOOLSET. This is in addition to the "15 person months" manual efforts they spent on MUC-3 evaluations. For a fully automated learning approach, such as GRID or Alice, the resulting IE system is more portable across domains.

4.7.2 Results on semi-structured text corpora

We report the results of GRID compared to other reported systems on two publicly available corpora: the CMU seminar announcements and the Austin job listings [www_3]. Based on the experiments in the last section regarding the effects of different performance of the four settings, we also perform similar experiments on the semi-structured corpora. We find that the employment of full-parsing cannot improve systems' performance (only about 0.1% improvement in average, see Table 4.3). Thus in this section, we only use shallow parsing for the documents in these semi-structured domains. The utilization of lexical chaining can achieve 1.5% improvement (see Table 4.4); then thus we employ lexical chaining on these two tasks. Finally, we find that the best performance can be obtained with context chunk size equal to 4 and *w* equal to 3.

For these two tasks, we perform 5 trials validation experiments. In each trial, we randomly partition the data into two halves, using one half for training and the other half for testing. Our results in Table 4.9 and Table 4.10 are the average of these 5 trials. We use MUC-7 scorer [Douthat, 1998] to score each slot. The results of the first task in terms of F_1 measure are summarized in Table 4.9, along with the results of other state-of-the-art IE systems. We extract results for ME_2 and SNoW systems from [Chieu and Ng, 2002a], and those of the other systems from [Ciravegna, 2001]. Considering the average accuracy for all the slots, it can be seen that GRID outperforms other reported systems on the same task. In this domain, GRID performs worse on slot of <location>. The main reason is that the named entity recognizer we used is designed to extract the general location types,

	speaker	location	stime	etime	All(averaged)
GRID	85.7	76.2	99.6	96.0	89.3
ME_2	72.6	82.6	99.6	94.2	87.3
$(LP)^2$	77.6	75.0	99.0	95.5	86.8
SNoW	73.8	75.2	99.6	96.3	86.2
BWI	67.7	76.7	99.6	93.9	84.5
HMM	76.6	78.6	98.5	62.1	79.0
Rapier	53.0	72.7	93.4	96.2	78.8
SRV	56.3	72.3	98.5	77.9	76.3
Whisk	18.3	66.4	92.6	86.0	65.8

Table 4.9 F_1 measure obtained by GRID on CMU seminars

slot	GRID	$(LP)^2$	Rapier	BWI
Id	100	100	97.5	100
Title	45.3	43.9	40.5	50.1
Company	79.1	71.9	69.5	78.2
Salary	80.7	62.8	67.4	
Recruiter	81.2	80.6	68.4	
State	90.1	84.7	90.2	
City	93.5	93.0	90.4	
Country	94.8	81.0	93.2	
Language	88.1	91.0	80.6	
Platform	78.2	80.5	72.5	
Application	76.7	78.4	69.3	
Area	65.1	66.9	42.4	
Req-yeas-e	70.2	68.8	67.1	
Des-years-e	71.1	60.4	87.5	
Req-degree	86.3	84.7	81.5	
Des-degree	73.4	65.1	72.2	
Post date	99.5	99.5	99.5	
AllSlots(averaged)	80.8	77.2	75.9	

Table 4.10 F_1 measure obtained by GRID on job listings

such as the countries, cities *etc.*, whereas the locations in this corpus are not in general location forms. Most locations refer to meeting room numbers such as “PH 223D”. Thus GRID misses out some of these locations.

The second task performs IE on 300 job announcements. The task consists of identifying for each job listing: message id, job title, salary offered, company offering the job, recruiter, state, city and country where the job is offered, programming language, platform, application area, required and desired years of experience, required and desired degree, and posting date. The results are presented in Table 4.10. The results of other reported systems are taken from [Ciravegna, 2001]. As BWI [Freitag and Kushmerick, 2000] was only tested on a very limited subset of slots, we do not compare GRID with it.

Again, it can be shown that GRID outperforms $(LP)^2$ and Rapier system with respect to the overall average F_1 measure. From the experimental results on these two webpage corpora, we observe that the named entity identification module can improve the accuracy especially in slots of <speaker>, <stime>, <etime>, <salary>, <company> and <recruiter> since it can recognize persons, time, money and organizations. GRID performs worse on those slots that do not conform to general named entity types, such as <title>, <application>. Some specific domain knowledge can help to improve the precision on these types of slots.

4.8 Discussion

We can see that the features employed in GRID are general and effective for both the semi-structured corpora and free-text corpus. We believe that the excellent performance of GRID is mainly due to the following reasons: (a) the use of global feature distribution at various lexical, syntactic and semantic representations to determine the best element feature to start the rule induction process; (b) the introduction of semantic constraints by the named entity recognizer; (c) the use of chunking analysis to delimit the boundaries of the slots; and (d) the use of lexical chaining to perform rule generalization. This is demonstrated in the free text corpus in which the rules induced by GRID are more general and cover more cases in the test set. We believe that GRID can perform even better on semi-structured corpora if we have a proper domain specific named entity recognizer that can identity semantics of slots such as titles, applications *etc.*, as discussed earlier.

After observing the errors on the free-text corpus that GRID may occur, we make the following error analysis: (a) Some errors are accumulated from the previous basic NLP modules, such as PoS tagging, chunking and named entity recognition. (b) Most of errors are incurred due to problems in pattern rule matching because of insufficient training examples and the limitation of rule representation. (c) The lexical chaining and WordNet-based rule generalization can overcome part of problem (b), but a number of errors are incurred in rule generalization using WordNet. For example, “kidnap” and “seize” are in the same synset in WordNet dictionary, thus we group “kidnap” and “seize” into one semantic class during the lexical chaining process. This gives rise to a rule of “NP VP_passive (semantic_kidnap) → NP is a *victim*”. When applying this rule to the instance of “the cocaine was seized…”, “the cocaine” will be wrongly tagged as “victim”. Fortunately this “false positive” error occur only in a small number of instances, and the error incurred is small in proportion to the effectiveness of the overall generalization by synset. We believe that the use of good ontology will help in this respect. (d) The other source of errors is attributed to the use of limited NLP understanding on free text document. GRID fails in cases where deep discourse understanding is needed. For example, in the sentence: “Mr. Pastrana said the bomb exploded outside the building of the secret police, known from the Spanish initials as DAS, in western Bogota at 7:35 A. M..”. In MUC-4 answer keys, “DAS” is “target”. However, GRID fails to identify it because it cannot associate “DAS” with “the secret police”. To overcome this problem, we need to explore discourse analysis and knowledge based method.

For semi-structured and structured texts, the error analysis of (c) and (d) are not so crucial. The more specific named entity recognizer is needed for the seminar announcement documents, such as for the identification of the seminar locations. The other limitation is the insufficient training example from which we cannot learn enough pattern rules to cover all of the unseen test instances. One of the future works is that we might need to utilize the HTML tags to learn more structured pattern rules.

4.9 Summary

The ability to extract the desired pieces of information from natural language texts is an important task with a various number of potential applications. This chapter presents a global rule induction algorithm, GRID, which makes use of global feature distribution in the whole training examples. A major difference between GRID and other pattern rule learning systems is that GRID learns the rule by starting from a global optimal feature in current active training examples. The first rule generated by GRID thus covers the most number of positive training examples in current active training instances set and satisfies the pre-defined error tolerance. The main contribution of this research is in employing global information to extract high quality pattern rules. It extracts rule at the global level by examining the global statistics of all instances represented at the lexical, syntactic and semantic levels simultaneously. The features used in GRID are not specific to any domain. Our tests on both semi-structured corpora and free text corpus indicate that our approach is effective.

In the next Chapter, we will describe the applications of GRID in two tasks other than information extraction. We will then present two bootstrapping paradigms which employ GRID as the basic learner in Chapter 6 and 7.

Chapter 5

Applications of GRID on Other Tasks

Although GRID is originally designed for information extraction tasks, it is a general pattern rule learner that can be applied to other problems. In this chapter, we introduce two applications of GRID on tasks other than text-based information extraction. One task is to use GRID to learn the pattern rules for identifying the definitional sentences in definitional question answering. The other task is to apply GRID to story segmentation in news videos.

5.1 GRID for Definitional Question Answering

Nowadays, there are more and more new terms and personalities introduced in popular media everyday, such as Clay Aiken, SARS, which are of great interest to the public. Definitional questions, *i.e.* questions like "What is a *taikonaut*?" or "Who is *Clay Aiken*?", have recently drawn much attention in research community [Voorhees, 2003a]. The new terms and personalities, though they appear in mass media, cannot be found in the authoritative sources of definitions, such as dictionaries or encyclopedias. We focus on identifying definitional sentences from new articles in this section. A definitional sentence contains descriptive information that can be included in an extended definition of the term. Definitional sentences usually present some lexical or linguistic patterns. These patterns can be hand-crafted or automatically machine learned. For example, we can identify pattern rules such as: "*Gunter Bloebel*, who is known as…" as typical definitional sentence pattern. However, it is time-consuming to manually construct definitional sentence patterns which tend to result in low recall. In this section, we present a supervised learning approach to identifying definitional sentences using GRID.

5.1.1 Data Preparation

Given a group of training sentences, GRID learns local contextual patterns surrounding the given search term. We do not handle long-distance dependencies, as our observations show that definition sentences are identified mainly by adjacent words and punctuations. The process of generating pattern instances is presented in Figure 5.1. The labeled definition sentences are first processed with Part-of-Speech (PoS) tagging and chunking by a natural language tagger and chunker, NLProcessor, which we use in Section 4.1. We then perform selective substitution of certain lexical items by their syntactic classes in order to generate representative patterns. The substitution attempts to replace words that are specifically related to the search term with more general tags so that the patterns can be applied to other sentences. The substitution rules that we use and some examples are listed in Table 5.1.

In Table 5.1, *centroid words* are those words that are highly correlated to the search term, as judged by mutual information. We adopt a local centrality metric of words with respect to the search term based on their co-occurrences with the search term within sentences. The rationale is that the search terms tend to appear with their descriptive sentences within news articles. As a news article usually describes multiple terms and persons, descriptive sentences are likely to repeat a term rather than using other forms of

1) Definition sentences (bold terms are search terms)
…… galaxies, **quasars**, the brightest lights in distant universe ……
…… according to **Nostradamms**, a 16th century French apothecary …….
…… severance package, known as **golden parachutes**, included ……
…… A **battery** is a cell which can provide electricity.
……
2) Reduced pattern instances (capitalized tags are chunks and syntactic classes):
NN, <Search_Term>, DT$ NN
according to <Search_Term>, DT$ NNP
known as <Search_Term>, VB
<Search_Term> BE$ DT$
……

Figure 5.1 Illustration of generating pattern instances

Token	**Substitution**	**Examples (from the example sentence in this section)**
Any part of the search term	<SCH_TERM>	“Iqra” → <SCH_TERM>
Centroid Words: (Topical words related to the search term)	Corresponding syntactic classes	“channel” → NN
Noun phrases by chunking	NP	“Arab Radio and Television company” → NP
Adjectival and adverbial modifiers	*To be deleted*	
is, am, are, was, were	BE$	is → BE$
a, an, the	DT$	“the” → DT$
(all numeric values)	CD$	
All other words and punctuations	*no substitution*	“Owned”, “by”, “of”, *etc.* are unchanged.

Table 5.1 Substitution heuristics for definitional question answering

reference. Our sentences also have been processed by an anaphora resolution module as in Section 4.1. As such, co-occurrence based metric is able to capture the local importance of words to the search terms without losing recall.

The co-occurrences of words can be measured by using the metrics described in [Lin and Hovy, 2000], which constructs topic signatures for document summarization. We employ mutual information as the measurement of co-occurrences for simplicity. All the words, after removing stop words, are stemmed before calculating their centrality. The equation for calculating the centrality *Centrality*$_{sch\_term}$*(w*$_i$*)* of a word w_i is as follows:

$$Centrality_{sch\_term}(w_i) = \frac{\log(Co(w_i, sch\_term)+1)}{\log(sf(w_i)+1)+\log(sf(sch\_term)+1)} \times idf(w_i) \qquad (5.1)$$

where *Co(*w_i*, sch_term)* denotes the number of sentences where w_i co-occurs with the search term *sch_term*; and *sf(*w_i*)* gives the number of sentences containing the word w_i. We also use the inverted document frequency of w_i, *idf(*w_i*),* as a measurement of its global importance[4]. Centrality scores for all words appearing in the input sentences are calculated and those words whose scores exceed the average plus a standard deviation form a set of centroid words.

The scheme of substitution by general information such as PoS and syntactic classes helps to capture obscure patterns. To demonstrate this, we present an example of a definition pattern that is not likely to be covered by previous work. The example does not describe a direct definition but indicates some important properties of the search term, which should be included in its extended definition. Given a definition sentence for "*Iqra*":

[4] We use the statistics from Web Term Document Frequency and Rank site (http://elib.cs.berkeley.edu/docfreq/) to approximate words' IDF.

The channel Iqra is owned by the Arab Radio and Television company and is the brainchild of the Saudi millionaire, Saleh Kamel.

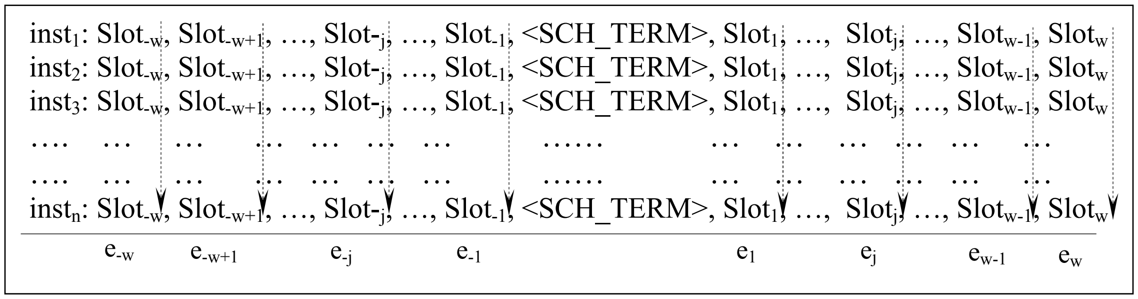
After substitution, the sentence is transformed into a token sequence comprising syntactic tags, words and punctuations as follows:

DT$ NN <SCH_TERM> BE$ owned by DT$ NP and BE$ DT$ brainchild of NP.

In order to generate general patterns, we need to consider the "context" around the <SCH_TERM>. The context is modeled as a window centered on <SCH_TERM> according to the pre-defined window size *w*, *i.e.* the number of slots (or tokens) on both sides of <SCH_TERM>. Thus we get fragments of size *2w+1* including the search term. We refer to such fragments as pattern instances on which the general GRID-learned pattern rules are to be generated. For example, the pattern instance from the above sentence is (*w*=3):

DT$ NN <SCH_TERM> BE$ owned by

Accumulating all the pattern instances extracted from the training definitional sentences and aligning them according to the positions of <SCH_TERM> (as shown in Figure 5.2), we can easily apply GRID algorithm to them and generate pattern rules automatically.



$inst_1$: $Slot_{-w}$, $Slot_{-w+1}$, …, $Slot_{-j}$, …, $Slot_{-1}$, <SCH_TERM>, $Slot_1$, …, $Slot_j$, …, $Slot_{w-1}$, $Slot_w$
$inst_2$: $Slot_{-w}$, $Slot_{-w+1}$, …, $Slot_{-j}$, …, $Slot_{-1}$, <SCH_TERM>, $Slot_1$, …, $Slot_j$, …, $Slot_{w-1}$, $Slot_w$
$inst_3$: $Slot_{-w}$, $Slot_{-w+1}$, …, $Slot_{-j}$, …, $Slot_{-1}$, <SCH_TERM>, $Slot_1$, …, $Slot_j$, …, $Slot_{w-1}$, $Slot_w$
$inst_n$: $Slot_{-w}$, $Slot_{-w+1}$, …, $Slot_{-j}$, …, $Slot_{-1}$, <SCH_TERM>, $Slot_1$, …, $Slot_j$, …, $Slot_{w-1}$, $Slot_w$
e_{-w} e_{-w+1} e_{-j} e_{-1} e_1 e_j e_{w-1} e_w

Figure 5.2 Global distribution of positive training pattern instances

5.1.2 Experimental Results

In order to evaluate the effectiveness of GRID in learning pattern rules for definitional question answering problem, we use the community standard TREC-12 definitional QA data set [Voorhees, 2003b]. This "TREC corpus" or the TREC QA corpus[5], consists of over one million news articles. A total of 50 definitional questions, along with answers in the form of answer nuggets, are provided with the corpus. Among the questions, there are 30 questions about people, 10 about organizations and 10 about other terms.

In order to provide additional training data for rule generation outside of TREC corpus, an auxiliary corpus of web documents are collected based on questions from Lycos. This "Lycos corpus" comprises 26 questions on people and other terms most frequently searched for in Lycos (http://50.lycos.com). For each question, we use Google's site search to get up to 200 news articles from each eight prominent news sites (*e.g.* CNN and BBC). The text body of each news page, embedded between HTML paragraph tags, is extracted. We asked seven subjects to label all definition sentences. The labeled sentences are processed into 596 positive and 15,442 negative training instances. We used these labeled training instances to learn pattern rules using GRID.

In order to get comparable evaluation results, we adopt the same evaluation metrics as used in the TREC-12 task. For each question, TREC gives a list of essential nuggets and acceptable nuggets for answering this question. An individual definition question is scored using nugget recall (NR) and an approximation to nugget precision (NP) based on length. These scores are combined using the F_5 (β=5) measure, where recall is five times as important as precision.

[5] The AQUAINT Corpus of English News Text.
http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002T31

For convenience of comparison, we use a set of hand-crafted pattern rules (listed in Table 5.2) as our baseline. These rules were employed in [Yang, *et. al.*, 2003], a system which tied for second place in the TREC-12 definitional QA task. The hand-crafted pattern rules are quite exhaustive which took one person several months of reading the news articles and refining the pattern rules. Table 5.3 shows the performance comparison between GRID-learned pattern rules and the hand-crafted rules. We can see an improvement of 6.56% in the F_5 measure over the hand-crafted rules when using GRID-generated pattern rules.

ID	**Regular expressions of rules**
1	<SCH_TERM> (who | which | that)* (is | are) (called | known as)*
2	<SCH_TERM> , (a | an | the)
3	<SCH_TERM> (is | are) (a | an | the)
4	<SCH_TERM> , or
5	<SCH_TERM> (- | :)
6	<SCH_TERM> (is | are) (used to | referred to | employed to | defined as | described as)
7	" (.+) " by <SCH_TERM>
8	(called | known as | referred to) <SCH_TERM>

Legend:
|*: Any one of the elements within the round brackets. *: Optional field. (.+): Any characters.*

Table 5.2 List of hand-crafted rules for TREC12

Use of Patterns	NR	NP	F_5 measure (% improvement)
Baseline (hand-crafted rules)	0.5100	0.1953	0.4669
GRID-learned rules	0.5361	0.2216	0.4975 (+6.56%)

Table 5.3 Comparison of definition patterns

From the experimental results, we observe that GRID can capture most typical pattern rules for definitional sentences. For instance, the rule "<SCH_TERM>, DT NN" generated by GRID can match the sentence "*Goth, a subculture ……*". Figure 5.3 shows an excerpt of the learned rules for definitional sentences identification by GRID.

Since definition sentences exhibit more syntactic flexibility that is difficult to capture by a "strict" matching scheme such as GRID, we do not obtain the best performance in definitional question answering task. [Cui, Kan and Chua, 2004] presents a novel soft matching pattern learner which achieves better performance than GRID.

```
<SCH_TERM> , DT NN
<SCH_TERM> , DT NNP
<SCH_TERM> , who won
<SCH_TERM> , (known | listed) as
who BE <SCH_TERM> 's
<SCH_TERM> BE DT NN
```

Figure 5.3 Sample rules generated by GRID for definitional sentences idenfication

5.2 GRID for Video Story Segmentation Task

The ever-increasing amount of broadcast news video from the internet requires people to manage the video effectively. One effective way to organize video is to segment it into small, single-story units and classify these units according to their semantics. In this section, we introduce the application of GRID learning to the story segmentation of new videos.

5.2.1 Two-level Framework

Our experiment is based on the multi-modal two-level news story segmentation framework as presented in [Chaisorn, *et.al.* 2004]. The hierarchical structure for our story segmentation scheme is in Figure 5.4. The two levels are: shot classification level and story segmentation level. The basic unit of analysis is the shot, and we model each shot using a combination of high-level object-based features, temporal features and low-level visual features. At the shot level, we employ a decision tree to classify the shots into the pre-defined shot types. At the story level, we use GRID to detect story boundaries using the shot genre information, as well as time-dependent features based on scene change and cue-phrases.

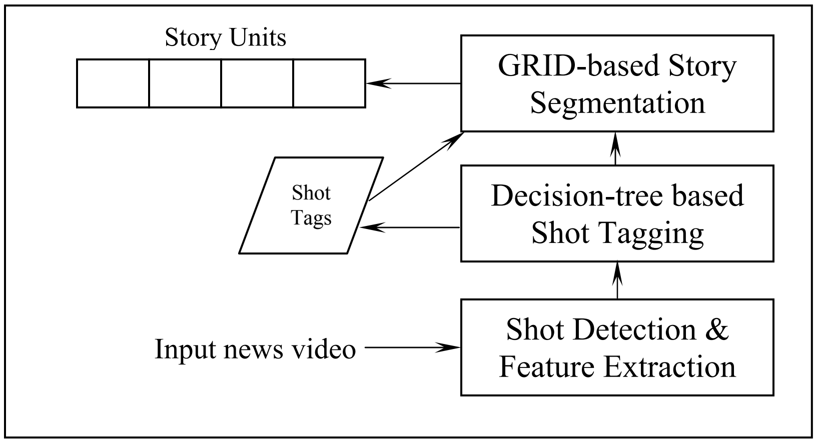

Story Units
GRID-based Story Segmentation
Shot Tags
Decision-tree based Shot Tagging
Input news video
Shot Detection & Feature Extraction

Figure 5.4 Overall system components

5.2.2 The News Video Model and Shot Classification

The details of slot classification and story segmentation can be found in [Chaisorn, *et. al.*, 2004]. Here we summarize the approach. We aim to propose an appropriate model for new video, and to determine the complete set of categories to cover all shot types. The categories must be meaningful so that the category tag assigned to each shot is reflective of its content and facilitates the subsequent stage of segmenting story units. To achieve this, we use the class taxonomy of TV Any-Time mode as the guide. We arrive at the following set of 12 shot categories: *Intro/Highlight*, *Anchor*, *2Anchor*, *Meeting/Gathering*, *Speech/Interview*, *Live-reporting*, *Sports*, *Text-scene*, *Special*, *Finance*, *Weather*, and

Commercial. In addition to these categories, we introduce five additional categories which are: "*LEDS*": to represent lead-in/out shots; "*TOP*": to model top story logo shots; "*SPORT*": to capture sport logo shots; "*PLAY*": to represent play of the day logo shots; and "*HEALTH*": to model health logo shots. Thus, the total number of shot categories is 17 which cover all essential types of shots in this collection. Some categories are quite specific such as the Anchor or Speech categories. Others are more general like Sports or Live-reporting categories. Each shot category is represented by unique *Tag_ID* (1, 2, …, 17).

News is a rather structured media with regular structures. It consists of a wide variety of shot types arranged in a well-defined sequence designed to convey the information clearly to a wide variety of audiences. Certain shot types like commercials, studio anchor person, finance and weather shots *etc.*, have well-defined and rather fixed temporal-visual characteristics. They can best be detected using specific detectors. Thus we perform the classification for the 17 shot categories using a range of techniques. Specifically, we first classify the commercial, anchor/2anchor shots, visual-based shots; and for the rest of shots, we employ a decision-tree based learning method to classify them. The details of the shot classification are as following:

(a) Commercial Detection: Commercial blocks and individual commercials are usually preceded and ended with a sequence of black frames and audio silence. Also, the ASR (Automatic Speech Recognition) recognition rate during the commercials is usually low, as there is more background music/noise. Hence, commercials tend not to have any recognized ASR outputs. The process of commercials detection is therefore accomplished in the following two steps: (i) black frames detection using color histogram; and (ii) commercials block detection using clustering technique based on a combination of black frames, silence and low ASR confidence level.

(b) Anchor/2Anchor Shot Detection: For most news video, we observe that anchor persons always appear in three different positions, *i.e.* left, center, or right position. Thus, in order to eliminate those shots with face detected but are unlikely to be *Anchor* shots, we use the number of faces detected, their sizes and positions to identify the *Anchor* and *2Anchor* shots.

(c) Visual-based Shot Detection: Visual-based shots are the shots that have distinct visual characteristics depending on their program categories and broadcast stations. They are regularly aired in certain time slots within the broadcast news. Examples of these visual-based shot categories are: “Finance”, “Weather”, LEDS, “health” logo, “SPORT” logo, and “TOP” (Top stories) logo. We use the 176-Luv-color-histogram as the feature, and employ image matching and video sequencing techniques developed in our lab to perform the detection.

(d) Rule-based Shots Detector using Decision Tree: The remaining shots are classified using Decision Tree. The feature vector used for each shot is of the form:

$$S_i = (a, m, d, f, s, t, c) \tag{5.2}$$

where a is the class of audio, $a \in$ {t = speech, m = music, s = silence, n = noise, tn = speech + noise, tm = speech + music, mn = music + noise}; m is the motion activity level, $m \in$ {l = low, m = medium, h = high}; d is the shot duration, $d \in$ {s = short, m = medium, l = long}; f is the number of faces, $f \geq 0$; s is the shot type, $s \in$ {c =

closed-up, m = medium, l = long, u = unknown}; t is the number of faces, $t \geq 0$; and c is set to "true" if the videotexts present are centralized, $c \in$ {t = true, f = false}.

5.2.3 Story Segmentation

After the shot classification, we perform story segmentation based on the combination of video, audio and ASR features. For each shot, we use 3 features to model it. The 3 features are as following:

(i) Shot categories: As we discussed earlier, we defined 17 shot categories to cover all type of news video. The value of this feature would be the Tag_ID of the shot (1, 2, …, 17).

(ii) Scene change: This feature indicates whether there is a change of scene between the previous and current shots. We represent with "c" for a change and "u" otherwise.

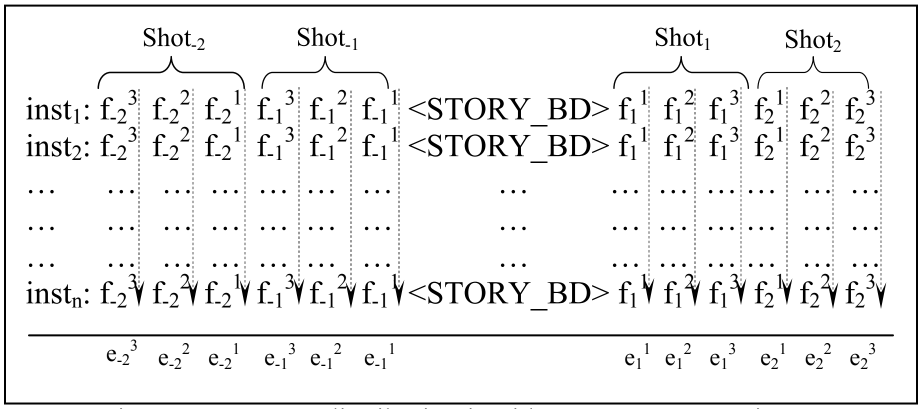
(iii) Cue-phrase: From the ASR results of the speech track in videos; we analyzed the statistics of cue-phrases that typically appear at the beginning of news stories (Begin-Cue). For each shot, we represent Begin-Cue as 1 (for presence of Begin-Cue) or 0 otherwise. An example of Begin-Cue is "good evening I'm <person_name>". An example of Misc-Cue is "when we come back".

We summarize the input features to GRID as listed in Table 5.4. We use left two and right two shots as the context information for GRID. Each shot has 3 features as listed in Table 5.4. We collect the video shots which there are story boundaries as GRID's positive training instances. The video shots which there are not story boundaries are used as negative instances. We align the positive instances according to the story boundaries positions (STORY_BD) as presented in Figure 5.5.

Feature	Description	Possible Values
f_g^1	Shot categories (Tag_ID)	1, 2, 3, …, 17
f_g^2	Scene change	"c" or "u"
f_g^3	Cue-phrase	"1" or "0"

Legend: g is the context position as in Section 4.3.

Table 5.4 Features used in GRID for video story segmentation



Shot_{-2}
Shot_{-1}
Shot_1
Shot_2
inst_1: f_{-2}^3 f_{-2}^2 f_{-2}^1 f_{-1}^3 f_{-1}^2 f_{-1}^1 <STORY_BD> f_1^1 f_1^2 f_1^3 f_2^1 f_2^2 f_2^3
inst_2: f_{-2}^3 f_{-2}^2 f_{-2}^1 f_{-1}^3 f_{-1}^2 f_{-1}^1 <STORY_BD> f_1^1 f_1^2 f_1^3 f_2^1 f_2^2 f_2^3
inst_n: f_{-2}^3 f_{-2}^2 f_{-2}^1 f_{-1}^3 f_{-1}^2 f_{-1}^1 <STORY_BD> f_1^1 f_1^2 f_1^3 f_2^1 f_2^2 f_2^3
e_{-2}^3 e_{-2}^2 e_{-2}^1 e_{-1}^3 e_{-1}^2 e_{-1}^1 e_1^1 e_1^2 e_1^3 e_2^1 e_2^2 e_2^3

Figure 5.5 Feature distribution in video story segmentation

5.2.4 Experimental Result

We test GRID's application on story segmentation task using the TRECVID 2003 [www_5] corpus. The training and test data are CNN and ABC news video of year 1998. Altogether, there are about 120 hours. About 60 hours of the videos is used for training, and the rest for testing. The shot classification is tested on a subset of development set (~12 hours of video). The average performance of shot classification in terms of F_1 value is 92.95 [Chaisorn, *et.al.*, 2004]. The result of the story segmentation is presented in Table 5.5. From Table 5.5, we can see that GRID performs comparably with the top ranked system which is implemented by NUS in TRECVID 2003 segmentation task which can achieve 77.5% F_1-measure [Chua, *et.al.* 2003]. Figure 5.6 shows an excerpt of GRID rules for video story segmentation task.

News video	Precision	Recall	F_1
ABC	71.95	84.51	77.72
CNN	76.76	68.47	72.38
Average	74.36	76.49	75.05

Table 5.5 Results of GRID application on video story segmentation

```
<story_boundary> ANCHOR
<story_boundary> 2ANCHOR
LEDS <story_boundary> SPORT
ADV <story_boundary> LEDS
LEDS <story_boundary> ANCHOR 2ANCHOR
```

Figure 5.6 An excerpt of GRID rules for video story segmentation

Observing the learned pattern rules by GRID in TRECVID 2003 task, we find that GRID can obtain many rules that humans can summarize through watching videos. For example, usually there is a story segmentation between two weather reports; GRID can learn a pattern rule of "*LEDS <story_boundary> sport*" which means if the previous shot is lead in/out shot and the following shot is a sport logo shot, then between these two shots, there is a story boundary. Different from the training examples for information extraction, in video story segmentation experiments, we find that there are many common patterns which appear both in positive training examples and negative training examples. GRID misses some of useful patterns which occur many times in negative training examples (too many times in negative examples will result in high *Laplacian* value, thus it is not a good pattern for GRID). Further research can be done to modify GRID to make it more appropriate for video story segmentation task, for example, use another suitable measurement for evaluating rules instead of *Laplacian* metric.

5.3 Summary

In this Chapter, we applied GRID as a general learner to definitional question answering problem and video story segmentation task. In both tasks, GRID achieved good performance and we believe that GRID can be employed as a general classification learner in other fields. In the following two Chapters, we will extend GRID from supervised learning paradigm to two weakly supervised paradigms to alleviate the human labor of annotation.

Chapter 6

Bootstrapping GRID with Co-Training and Active Learning

In Chapter 4, we introduced the supervised rule induction algorithm GRID for information extraction and presented its applications on tasks other than information extraction in Chapter 5. As supervised learning requires a large amount of manually annotated training data which are usually expensive to obtain, weakly supervised learning has drawn much attention in the IE community in recent years. In this Chapter, we present GRID_CoTrain, a weakly supervised paradigm by bootstrapping GRID with co-training and active learning. We also utilize external knowledge resource such as WordNet and existing ontology knowledge to optimize the learned pattern rules.

6.1 Introduction

As we discussed earlier, the mentioned IE systems such as GRID, WHISK and $(LP)^2$ are all supervised learning systems. Since it is time-consuming and error prone to manually annotate training data for supervised systems, there are many research efforts in recent years that focus on bootstrapping an IE system using a small set of annotated data and plentiful un-annotated data to implement weakly supervised learning [Xiao, *et. al.* 2002; Agichtein and Gravano, 2000; Collins and Singer, 1999; Blum and Mitchell, 1998]. Co-training is one such bootstrapping strategy. Co-training begins with a small number of labeled data and a large number of unlabelled data. It trains more than one classifier from the labeled data, uses the classifiers to label some unlabelled data, trains again the new classifiers from all the labeled data, and repeats the above process. Co-training with multiple views learners reduce the need for labeled data by exploiting disjoint subsets of features (views) such as contextual and content views, each of which is sufficient for learning. Initial studies of co-training focused on the applicability of co-training on clarifying the assumptions needed to ensure its effectiveness. Blum and Mitchell (1998) presented a PAC-style analysis of co-training and made two important conclusions: first, each view of the data should itself be sufficient for learning the classification task; and second, the views should be conditionally independent of each other. More formally, given that X is the training feature set and Y is the classification, we assume that $X=X_1 \times X_2$, where there exist functions $g_1: X_1 \rightarrow Y$ and $g_2: X_2 \rightarrow Y$ such that $f(X) = g_1(X_1) = g_2(X_2)$ for all $X=X_1|X_2$. In the real world domain, this ideal assumption is not fully satisfied as there are some ambiguities in the classes of noun phrases [Jones, *et al.* 2003]. For example, the noun phrase "*Columbia*" could be a "*location*" (in the context of "*headquartered in Columbia*") or an "*organization*" (in the context of "*Columbia published …*"). Although the tight constraints are not fully satisfied in most information extraction tasks, Nigam and Ghani [2000] found that co-training with separate feature sets still performed better than the ones which do not split the feature sets. [Jones, *et al.* 2003] showed that combination of active learning and bootstrapping (using CoEM algorithm) made the information extractor robust although the ambiguities of noun phrase classes exist.

One of the problems when applying co-training algorithms for natural learning from large datasets is the scalability problem. Degradation in the quality of the bootstrapped data arises as an obstacle to further improvement [Pierce and Cardie, 2001] in the learning process. In this Chapter, we combine co-training with active learning to overcome this problem. Active learning methods attempt to select only the most informative examples for annotation and training and therefore are potentially very useful in natural language applications. In the experiments, we include a human to annotate some instances after every few iterations in an active learning framework. We investigate several active learning strategies in the co-training model to determine which instances should be annotated by the human. The strategies considered include: uniform random selection, density selection, committee-based sampling and confidence-based sampling.

On the other hand, during the course of bootstrapping, the quality of the learned pattern rules is crucial for the effectiveness of the final learner. One of the major issues in learning rules concerns the tradeoff between specific, unambiguous extraction rules and the need for general rules that can be applied widely. While the concept hierarchy defined in WordNet is too general, specific domain knowledge is useful for better rule generalization. We need an effective rule generalization strategy using existing domain ontology knowledge in learning of the pattern rules from a small set of labeled instances. Another issue is that the errors incurred by the learned rules in the earlier iterations will be accumulated in the later iterations. Existing ontology knowledge may also help us to verify the correctness of some of the automatically learned rules.

In this Chapter, we describe a bootstrapping pattern rule-based IE system, called GRID-CoTrain, which co-trains the context and content views with active learning using an existing pattern rule learner GRID. In Section 6.2, we describe some related research of bootstrapping for IE tasks. Section 6.3 describes the bootstrapping algorithm in detail. In Section 6.4, the strategies of rule optimization using the general WordNet lexicon plus the domain ontology knowledge base is described. Section 6.5 presents our experimental results on the terrorism domain. Finally, we conclude this Chapter in Section 6.6.

6.2 Related Bootstrapping Systems for IE Tasks

In recent years, many researchers have used bootstrapping technology for various information extraction tasks. The *Snowball* system [Agichtein and Gravano, 2000] introduces the strategies for generating patterns and extracting tuples from plain-text documents that require only a handful of training examples from users. They use a simple relation which is “organization-location” (where is the organization located) for evaluation. Multi-level bootstrapping is used in [Riloff and Jones, 1999] for generating both the semantic lexicon and extraction patterns simultaneously. To alleviate the deterioration in performance due to non-category words entering the semantic lexicon, the outer bootstrapping mechanism compiles the results from the inner bootstrapping process and identifies the most reliable lexicon entries. As input, it requires only un-annotated training texts and a handful of seed words for a category. [Collins and Singer, 1999] described two bootstrapping algorithms, DL_CoTrain (DL stands for decision list) and AdaBoost for named entity classification. In [Pierce and Cardie, 2001], the authors pointed out that there are some limitations to co-training for natural language learning from large datasets. They proposed a “corrected co-training” to solve the scalability problem during the course of bootstrapping and suggested the combination of weakly supervised learning (such as co-training) with active learning. [Muslea, *et al.* 2003] described an active learning model with a strong view and a weak view, Aggressive Co-Testing, for wrapper induction. A strong view consists of features that are adequate for learning the target concept (such as contextual view); in contrast, in a weak view, one can only learn a concept that is more general or specific than the target concept (such as the content view). Take for an example, to extract the telephone number from a webpage, a strong view rule could be "* Phone: (number)" which indicates "skip any token until "phone" appears followed by numbers". This strong view rule is sufficient to extract the telephone numbers from structured web pages. A weak view rule, such as "(number) – number number", is not sufficient to identify whether it's a fax number or a telephone number. Aggressive Co-Testing exploits both strong and weak views and uses the weak views both for detecting the most informative examples in the domain and for improving the accuracy of the predictions. [Yangarber, 2003] introduced competition among several scenarios simultaneously. This provides natural stopping criteria for the unsupervised learners, while maintaining good precision levels at termination.

In GRID_CoTrain, we plan to combine co-training with active learning. Co-training is performed by training content rules and contextual rules. After several iterations, a human being is in charge of labeling several instances that are selected by several active learning strategies. We will investigate the performances of various sampling selection methods in the experimental evaluation section.

6.3 Pattern Rule Generalization and Optimization

Usually, in IE systems, the pattern rules need to be generalized to cover more unseen instances and to be specialized to remove false extractions by too general rules. [Kim and Moldovan, 1995] introduced a generalization and specialization strategy through a pre-defined concept hierarchy tree. Other IE systems such as WHISK [Soderland, 1999] and CRYSTAL [Soderland, *et. al.*, 1995] used pre-defined semantic classes to generalize pattern rules. In [Chai and Biermann, 1997], the authors proposed a corpus based statistical generalization tree model in rule optimization. The rule optimization was implemented by the generalized noun entities in WordNet. The degree of generalization is adjusted to fit the user's needs by the use of statistical generalization tree model. As the conceptual definition in WordNet is domain independent and generic, we utilize the synset in WordNet for rule generalization. In addition, we deploy a specific existing domain dependent ontology for better rule generalization. We use MUC-4 corpus as our evaluation corpus and utilize existing ontology knowledge on terrorism from Teknowledge Company [www_1] as the conceptual hierarchy thesaurus.

6.4 Bootstrapping Algorithm GRID_CoTrain

We use GRID as the base learner in the bootstrapping scheme. The bootstrapping scheme is co-trained by two views, one is the content view and the other is the contextual view.

6.4.1 Bootstrapping GRID Using Co-training with Two Views

Co-training with multiple views is a weakly supervised paradigm for learning a classification task from a small set of labeled data and a large set of unlabeled data, using separate, but redundant, views of the data. The main task of information extraction is to extract specific semantic entities from the text documents and to determine their relationship in filling in a template. The extraction of semantic entities requires the use of context, which can be expressed in the general form as:

$$<c_{-k}>\ldots<c_{-2}><c_{-1}>\ <c_0>\ (sem\_entity)\ <c_{+1}><c_{+2}>\ldots<c_{+k}> \tag{6.1}$$

where $<c_i>$ $\{i = -k$ to $+k;\ i\neq 0\}$ refers to the context token at position i of the semantic entity, and k is the number of context tokens considered. $<c_0>$ represents the central semantic entity itself.

In this Chapter, we consider the $<c_0>$ *(sem_entity)* as the content view X_1; the left and right context information (*i.e.* $<c_i>$ $\{i = -k$ to $+k;\ i\neq 0\}$) are considered as contextual view X_2. Figure 6.1 indicates the basic procedure of GRID-CoTrain with two views.

```
Given instance space: $X$
  Distribution $D$ over $X$:  $D = \{(x_i, y_i)\}_{i=1}^{m}$ ; $Y$ is the classifications.
  $X = X_1 \times X_2$ (two views: content view $X_1$ and contextual view $X_2$)
  GRID ($G$) is the base learner;
  Apply $G$ to the two views, $X_1(G)$, $X_2(G)$ to create two classifiers: $g_1$, $g_2$;
  Combine co-training (two classifiers $g_1$ and $g_2$) with active learning
    to get mapping $F$: $X \rightarrow Y$
```

Figure 6.1 Overall paradigm of GRID-CoTrain

We present the detailed algorithm of GRID_CoTrain as following:

(a) Initialization: We define two pools in this algorithm. One is *lexicon pool* which contains list of words/phrases for content view for each category. The other is *pattern pool* which contains list of patterns for contextual view derived so far for different category.

Initially, we set the *lexicon pool* to the set of user selected seed words for each category. The initial set of content rules equal to the set of "*seed*" words for each type. For example, "*FMLN*" is one of the seed words for the category of "*perpetrator*" in the MUC-4 corpus. The corresponding content rule is "*full string = FMLN* → *FMLN is a perpetrator*". The *pattern pool* is initially set to null.

(b) Label the training set using the current set of content rules. Instances where no rule can be applied are left unlabeled.

(c) Use the labeled examples to induce the pattern rules by employing the GRID algorithm. In the weakly supervised learning model of GRID_CoTrain, we do not have enough positive and negative examples to evaluate the pattern rules in which it is different from what we do in the supervised model of GRID. Instead of using the *Laplacian* measure in the supervised version, we use an *RlogF* metric [Riloff and Jones, 1999] to score each pattern rule. The score for each pattern is computed as:

$$R\log F(pattern_i) = \frac{F_i}{N_i} * \log_2(F_i) \qquad (6.2)$$

where F_i is the number of category members extracted by $pattern_i$ and N_i is the total number of noun phrases extracted by $pattern_i$.

We select the top n=3 extraction patterns for each category and put them in the *pattern pool* accordingly. Intuitively, the *RlogF* measure is a weighted conditional probability; a pattern receives a high score if a high percentage of its extractions are category members.

(d) Label the training set using the current set of pattern rules in the pattern pools for each category. Examples where no rule applies are left unlabeled. For each extracted noun phrase or word, we use the average logarithm metric used in [Thelen and Riloff, 2002] to score it. The score for each phrase or word is computed as:

$$AvgLog(phrase_i) = \frac{\sum_{j=1}^{P_i} \log_2(F_j + 1)}{P_i} \qquad (6.3)$$

where P_i is the number of patterns that extract $phrase_i$ and F_j is the number of distinct category members extracted by $pattern_j$.

We select the top 3 phrases or words for each category and put them in the *lexicon pool* accordingly. Generate the content rules for the noun phrases or words added to the lexicon pools as in step (a). Set the content rules to be the seed set plus the words or the noun phrases added to the lexicon pools and go to step (b). The algorithm can be stopped when it runs a fixed number of iterations or when there is no new entries added to the lexicon pools and pattern pools.

6.4.2 Active Learning Strategies in GRID_CoTrain

As discussed in Section 6.1, one problem in co-training paradigm is the degradation in quality of automatically bootstrapped data which presents an obstacle to further improvement in performance. In GRID_CoTrain, we combine the active learning strategies with co-training to tackle this problem. Active learning is to determine which unlabeled instances to label next in order to maximize the learning objective with least labeling effort. We restrict our study to selective sampling of active learning here. Sample frequency is usually used in selective sampling strategies. Given a training example of "<c_{-k}>…<c_{-2}><c_{-1}> <c_0> (*sem_entity)* <c_{+1}><c_{+2}>…<c_{+k}>", we consider the frequency of the <c_0> (*sem_entity)* as our criteria for selective sampling. Some researchers used the frequency of occurrences of context information as the criteria for selective sampling [Jones, *et. al.*, 2003]. We investigate several active learning strategies for GRID-CoTrain.

The active learning strategies we investigate are as following:

(a) Uniform random selection: It selects the <*sem_entity*> that appear in the training examples at least once randomly with equal probability. The actual sample frequency is ignored. This sampling selection could be considered as baseline among the various strategies.

(b) Density selection: This sampling selection considers the actual sample frequency. The most frequent <*sem_entity*> in the unlabeled examples set is selected for annotation first. This method is based on the assumption that labeling frequent occurring samples would be beneficial for the learner.

(c) Certainty-based selection [Lewis and Catlett, 1994]: This sampling selection selects samples with lowest certainties and presents them to the user for annotation. We use the Equation (6.3) as the certainty metric.

(d) Committee-based selection [Freund *et al.* 1997]: In this paradigm, we regard the committee-based sampling as feature set disagreement. Since we learn two classifiers based on content view and context view, one way is to select samples where a human can provide useful information to identify samples where these two classifiers disagree. If there are such instances, we present them to the human annotator.

We trust the classification of the samples annotated by the human annotator. So the samples that are labeled by the human are considered as correctly labeled training samples and the tagged noun phrases are put into their according lexicon pools. The labeled samples are put into the labeled training pool for the base learner in the later iterations. In Section 6.6, we will present the performances of different active learning strategies in the terrorism domain. In our study, after every 5 iterations, we ask a human user to annotate up to 20 samples selected by the active learning strategies.

6.5 Rule Generalization Using External Knowledge

At the end of every 10 iterations in the course of bootstrapping, we utilize some rule optimization strategies such as rule generalization. We employ two levels of rule generalization strategies to improve information extraction performance. One is based on the general lexical knowledge base WordNet. The other is the domain specific ontology knowledge for fine-grained rule generalization.

6.5.1 Rule Generalization Using WordNet

As discussed in Chapter 4, at the end of the rule induction learning, we extract a set of satisfied pattern extraction rules. In general, the rule set obtained is not optimal as it did not consider the lexical and semantic relationships between features used in different rules. For example, for the <victim> slot in the terrorism attack domain, we may generate similar rules but with one different slot element, such as the "*murder of <victim>*" and "*assassination of <victim>*". As these rules share similar semantic meaning, they should be merged into a more general rule where the root noun is of the semantic class *{murder, assassination}*. The generalized rule's score is re-computed according to Equation (6.2).

To achieve this, we aim to perform lexical chaining on those rules that contain feature representations of verb phrases or noun phrases. We employ a lexical chaining algorithm as described in Chapter 4 (Section 4.4) to determine if the root verbs or head nouns can be replaced by their semantic groups. The process uses synsets in WordNet along with corpus statistics to find the common semantic group of different lexical tokens. At the end of lexical chaining process, we obtain a set of semantic groups, each containing a cluster of related words. These semantic groups are used as the basis to generalize some features related to nouns and verbs.

6.5.2 Fine-grained Rule Generalization Using Specific Ontology Knowledge

Similar to the rule generalization using specific ontology knowledge as discussed in Section 4.5, we use the same terrorism ontology knowledge [www_1] as the source for fine-grained rule generalization for GRID_CoTrain. For example, class "building" has several sub-classes such as "Garage", "Hotel", "Store" *etc.* in the ontology knowledge dictionary (refer to Figure 4.3). We may utilize these class hierarchies to generalize our pattern rules in content view. For example, if we have "grocery store" as "target" in the content view, we can expand it to all of the sub-classes of "Building" based on definition in Figure 4.3. In this way, we can extract more relevant and useful sample during the bootstrapping process.

6.6 Experimental Evaluation

To evaluate the active learning and rule generalization strategies in GRID_CoTrain, we perform experiments on terrorism news articles from the MUC-4 corpus [MUC-4 Proceedings, 1992]. For training, we use the training corpus and the TST1 and TST2 documents. Altogether, there are 1500 documents of the terrorism texts of which 50% are relevant to terrorism attacks. The TST3 and TST4 documents are used for testing the learned rules. We compare our bootstrapping algorithm of GRID_CoTrain with the supervised results of GRID in Chapter 4. We run the two algorithms on three semantic categories (perpetrator, victim, and target). The seed word lists use for the bootstrapping experiments are presented in Figure 6.4.

Perpetrator: fmln, armed forces, shining path, armed men, eln, guerrilla
Victim: peasants, jesuit priests, mayor of achi, carlos julio torrado, enrique lopez albujar
Target: government house, stores, electric towers, headquarters, electricity facilities

Figure 6.4 Seed word list for bootstrapping

First, we evaluate the different active learning strategies discussed in Section 6.4.2. We score the results the same as we did in AutoSlog-TS and Section 4.6.1.5 in which each extracted item is assigned to one of the five categories of: *correct, missed, mislabeled, duplicate,* or *spurious*. We compute the F_1-measure as following:

Recall(R) = *correct* / (*correct* + *missing*)

Precision (P) = (*correct* + *duplicate*) / (*correct* + *duplicate* + *mislabeled* + *spurious*)

$F_1 = 2 \times P \times R / (P + R)$

We run GRID_CoTrain algorithm 100 times and ask a human to annotate up to 20 samples manually after every 5 iterations where the samples are selected using different strategies of active learning. The learned rule sets are re-evaluated every 10 iterations. Figures 6.5, 6.6 and 6.7 show the F_1 value of the three concept slots for GRID_CoTrain based on different active learning strategies respectively. All of the results are obtained after rule generalization by WordNet plus domain ontology knowledge.

From the Figures we can draw the following conclusions:

(a) Bootstrapping with co-training by content view and context view without active learning performs well in the first 50 iterations. However, as the automatically annotated samples become larger, the performance decreases. This is partly due to the errors accumulated by automatically labeled samples.

(b) The use of the active learning helps GRID_CoTrain to improve its performance steadily as we perform more bootstrapping iterations, as compared to the non-active learning system. Among the active learning strategies, we found the certainty-based and committee-based strategies to be the most effective.

(c) GRID_CoTrain performs almost comparable to the supervised learning by using the active learning strategies. Thus the human-involved active learning maintains the quality of the learned rules, and yet the effort on the part of human annotator remains small (needs only about 20% in our experiments) in proportion to the amount of labeled data needed for training the fully supervised version of GRID.

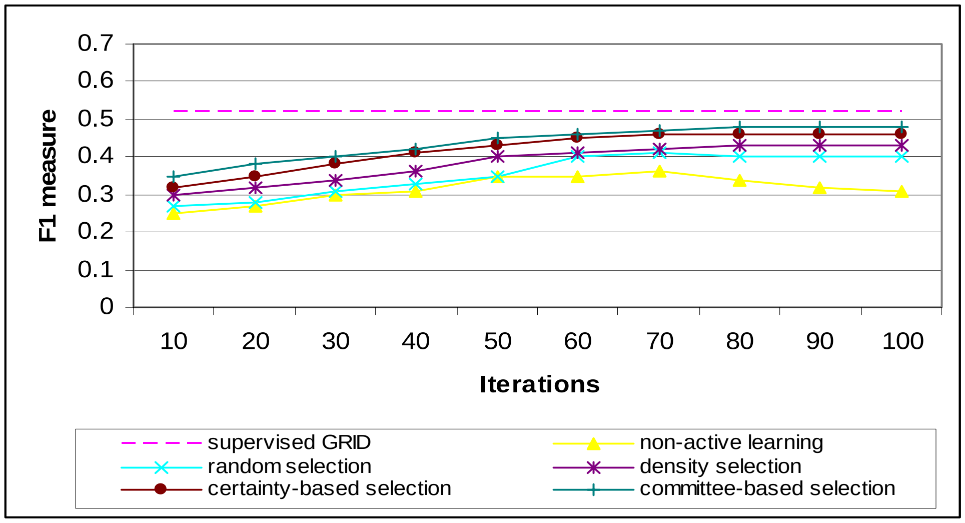


Figure 6.5 Performances of different active learning strategies for slot “perpetrator”

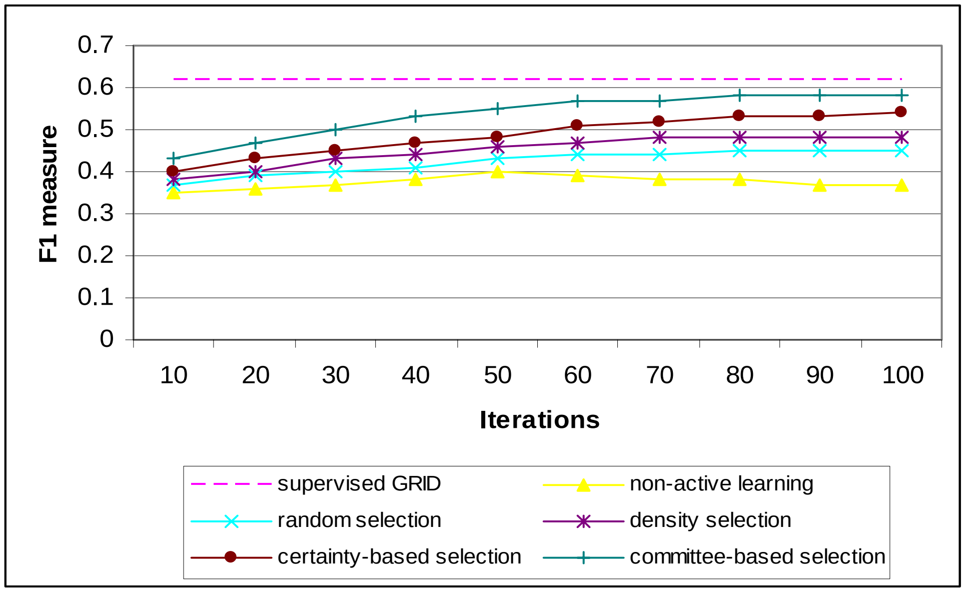


Figure 6.6 Performances of different active learning strategies for slot “victim”

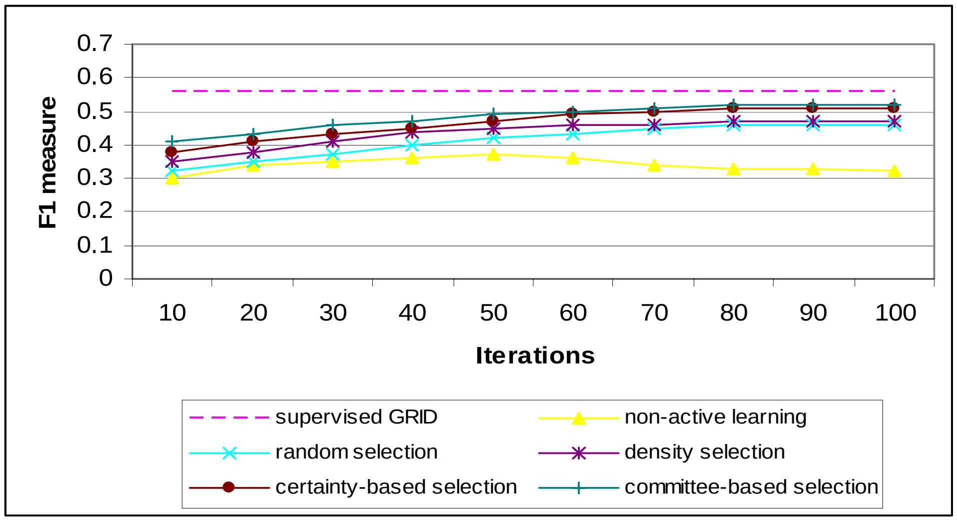


Figure 6.7 Performances of different active learning strategies for slot “target”

To evaluate the effect of specific ontology knowledge on rule generalization, we perform further experiment to compare rule generalization using WordNet only and using WordNet plus the domain ontology knowledge. We use the committee-based active learning strategy (averaged F_1 for the three concept slots) as the experimental setting for evaluating the rule generalization method with/without domain knowledge. Figure 6.8

shows that the specific domain ontology knowledge can improve the performance of IE by 2%.

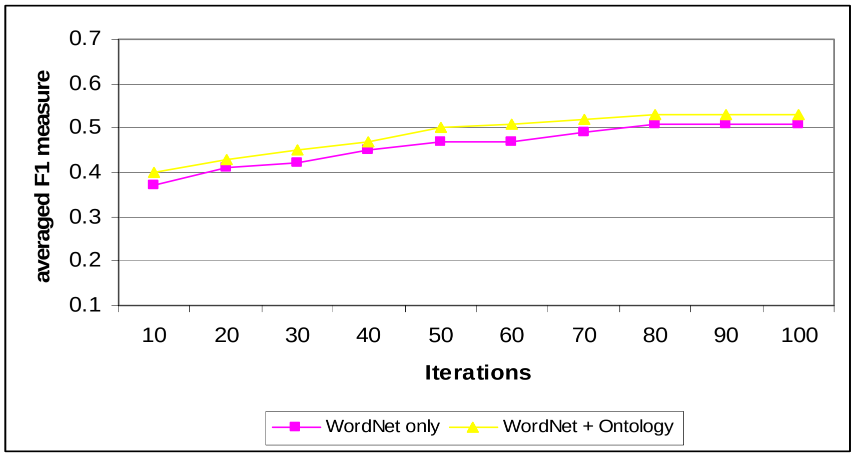


Figure 6.8 Comparison of two rule generalization methods

6.7 Summary

Based on our previous pattern rule induction algorithm, GRID, this Chapter described a bootstrapping pattern induction algorithm, GRID_CoTrain, which combines co-training with two contextual and content views and several active learning strategies. GRID_CoTrain required about 20% human annotation instances through the use of active learning strategy during the bootstrapping process to achieve comparable performance of the fully supervised learner of GRID. We also found that the use of existing domain ontology knowledge can improve the rule induction performance of an information extraction system. In the next Chapter, we will present an alternative bootstrapping scheme (GRID+SP) that combines GRID and a soft pattern learning module (SP) to realize a weakly supervised learner for the IE tasks. GRID+SP outperforms GRID_CoTrain while requiring less human annotation labor to achieve comparable performance of the supervised learning results.

Chapter 7

Cascading Use of GRID and Soft Pattern Learning

In the previous Chapter, we introduced GRID_CoTrain, a bootstrapping scheme that combines co_training and active learning using GRID as the base learner. From the experiments, we could see that active learning strategies that include human to annotate about 20% of training instances could achieve comparable performance of the fully supervised version of GRID system. We consider this manual labor is still too high. In this Chapter, we present a new bootstrapping paradigm with cascading learners using a probabilistic soft pattern learner (SP) with the hard pattern learner, *i.e.* GRID. The combination of SP and GRID can reduce much of the human labor of annotation. With about 5%-10% of manually tagged seed instances, the cascaded learner, GRID+SP, can obtain performance close to the fully supervised learner.

7.1 Introduction

The rules generated by GRID, which are introduced in Chapter 4 are used to match the test instances by performing exact matching for each slot, which we call "*hard matching*". Utilizing hard matching pattern rules can obtain precise results from test instances.

However, it may be problematic in dealing with natural language text, such as news articles, which often exhibits great variations in both lexical and syntactic constructions. For instance, in the terrorism domain, given a common pattern rule of "<victim> be kidnapped by …", it cannot pick up the instance "<victim> , kidnapped by …" due to the mismatch in only one token. Such hard matching techniques often result in low recall, especially in the case when there are insufficient tagged training instances. To achieve the flexibility in pattern matching for natural language text, soft matching pattern rules have been proposed for question answering [Cui, *et. al.*, 2004]. Soft pattern rules match test instances using a probabilistic model, which can better accommodate variations in expressions and favor high recall. However, differing from the question answering problem, both recall and precision are equally important in IE tasks. Moreover, the IE task needs to precisely locate the boundaries of the extracted slots. As such, the performance of soft pattern rules alone may not be sufficient for IE tasks.

In this Chapter, we aim to take advantage of both the soft and hard matching pattern rules to combat the existing problems caused by hard matching methods. Meanwhile, we want to minimize the number of hand-tagged training instances needed to start the learning process by adopting a bootstrapping strategy such as that proposed in [Riloff and Jones, 1999]. In contrast to current work, we propose a weakly supervised IE framework which makes use of both soft and hard matching pattern rules in both training and test phases. Starting with only a small set of tagged training instances, we first generate a set of soft pattern rules and utilize them to tag more training instances. The idiosyncrasy of soft pattern rules ensures sufficient coverage of automatically tagged instances. Next, we conduct the hard matching pattern rule induction algorithm introduced in Chapter 4, GRID, over both manually and automatically tagged instances to generate more accurate rules. These hard pattern rules are utilized to tag training instances for soft pattern rule generation in the next iteration. The process runs iteratively until the termination criteria are met. At the end of the training process, we obtain two sets of pattern rules, namely the hard and soft pattern rules. During test phase, both sets of pattern rules are used in a cascaded way — with hard pattern rules followed by soft pattern rules — to extract target slots on new documents. We conduct two experiments on both semi-structured and free texts to demonstrate the effectiveness of our method. The experimental results show that the bootstrapping scheme with two cascaded pattern rule learners could achieve a performance close to that achievable by fully supervised learning while using only 5~10% of the hand-tagged data, which are less than the human labor needed in the bootstrapping scheme with co-training and active learning in the last Chapter.

The main contribution of our new bootstrapping paradigm is in incorporating the soft matching pattern rules in the weakly supervised framework for IE.

7.2 System Design

Figure 7.1 shows the overall system architecture of our weakly supervised IE system. The training phase of the system is carried out as follows:

(a) Given a small set of hand-tagged instances (*seed instances*) provided by the user.

(b) We generate soft pattern rules using the seed instances, and denote the soft pattern rules as SP_i.

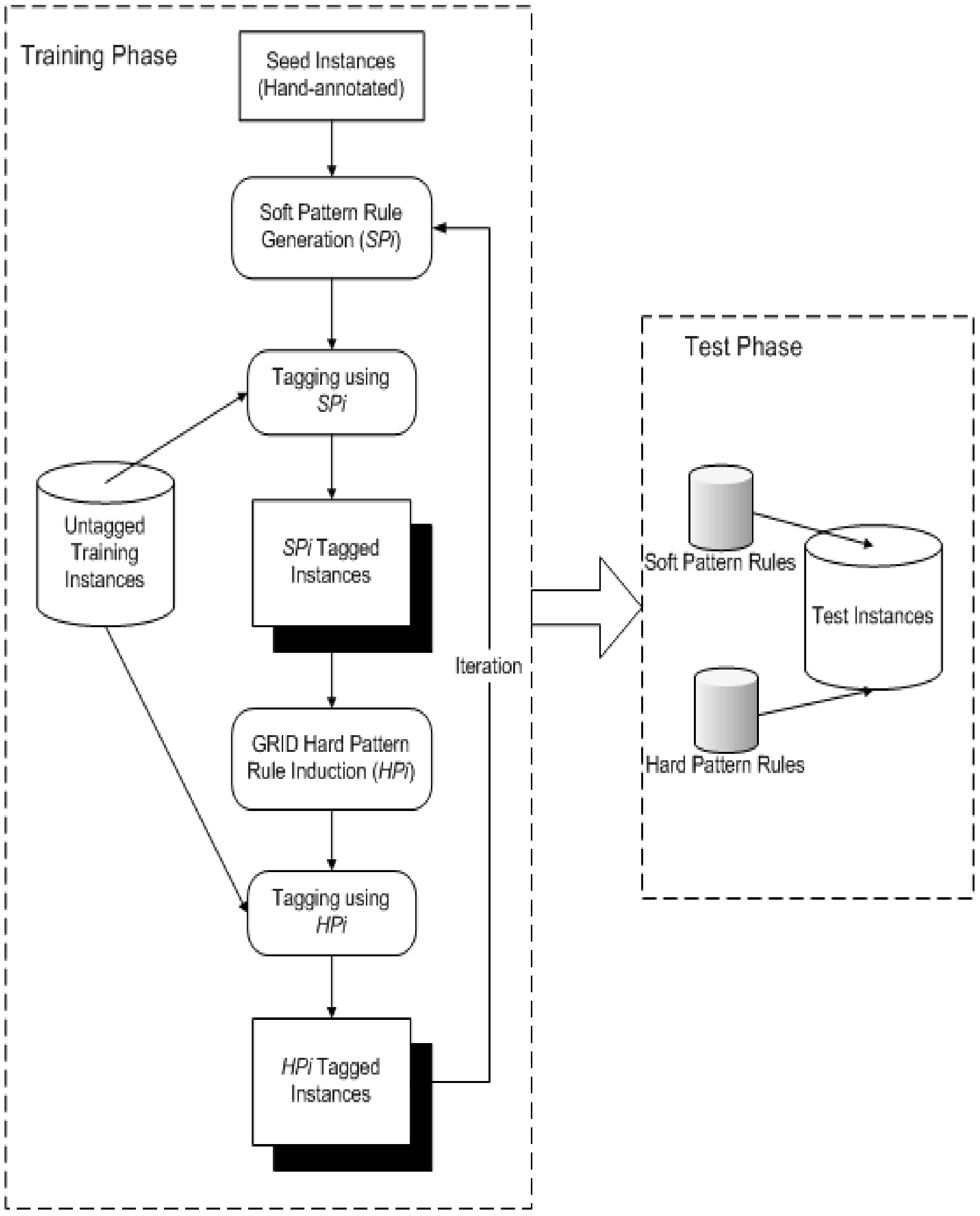


Figure 7.1 Architecture of the weakly supervised IE system by soft and hard pattern learners

(c) We apply the learned soft pattern rules (SP_i) to automatically tag un-annotated data. We employ a simple cut-off strategy that keeps only the highly ranked tagged instances by the soft pattern rules.

(d) We generate hard pattern rules using GRID over the automatically tagged instances and seed instances. The resulting hard pattern rules are denoted as HP_i.

(e) If the termination condition is satisfied, the process ends with a set of learned soft and hard pattern rules. Otherwise, the hard pattern rules HP_i are used to tag the training data again. We start a new round of training from Step (b) using the newly tagged training instances and seed instances.

In the test phase, we apply both the hard and soft pattern rules to match against test instances. Specifically, soft matching pattern rules would assign a probabilistic score to an instance that is not matched by any of the hard matching pattern rules. Only those fields that are matched by hard pattern rules or have high scores in soft pattern matching would be extracted.

7.3 Data Preparation

Similar to the data preparation part in Chapter 4, before the pattern rule learning commences, we pre-process the training and test sentences by using a natural language chunker[6] to perform Part-of-Speech (PoS) tagging and chunking. We also use a rule-based named entity tagger [Chua and Liu, 2002] to capture semantic entities. Given a tagged instance, we consider the left and right k chunks around the tagged slot as context:

$$<c_{-k}>\ldots<c_{-2}><c_{-1}>\ <c_0>\ (tagged\_slot)\ <c_{+1}><c_{+2}>\ldots<c_{+k}> \qquad (7.1)$$

Here $<c_i>$ $\{i=-k \text{ to } +k;\ i\neq 0\}$ represents the contextual chunks (or slots) of the tagged slot, where k is the number of contextual slots considered. $<c_0>$ *(tagged_slot)* represents the central tagged slot itself. $<c_i>$ can be of various feature types, namely words, punctuation, chunking tags like verb and noun phrases, or semantic classes. We use the left and right k chunks beside every noun phrase to prepare for the un-tagged instances since we assume that information extraction is to classify the noun phrases in text documents. We perform

[6] We used NLProcessor from http://www.infogistics.com/

selective substitution to generalize the specific terms in each slot so as to make the learned pattern rules general enough to be applied to other instances. Table 7.1 shows the substitution heuristics employed in our system with examples.

Tokens	Substitution	Examples
9 types of named entities	NP_Person, NP_Location, NP_Organization, NP_Date, NP_Day, NP_Time, NP_Percentage, NP_Money, NP_Number.	"Friday"→NP_Day "Feb.27"→NP_Date
Noun Phrase	NP_HeadNoun	"the seminar" →NP_seminar
Verb Phrase (passive or active)	VPpas_RootVerb, VPact_RootVerb	"will speak" →VPact_speak, "will be held" →VPpas_hold
Preposition Phrase	PP	"in cilivlian clothes" → PP
Adjectival and adverbial modifiers	*To be deleted*	
All other words and punctuations	*No substitution*	"Time", "at", "by", *etc.* are unchanged.

Table 7.1 Substitution heuristics for information extraction

Figure 7.2 gives 5 examples of original training instances for "*starting time*" (*stime*) in the seminar announcement domain. We substitute the more general syntactic or semantic classes for the lexical tokens according to the heuristics in Table 7.1.

7.4 Soft Pattern Learning

Soft pattern rules have been successfully applied to text mining [Nahm and Mooney, 2001] and question answering [Cui, *et. al.*, 2004]. We employ a variation of the soft pattern rules generation and matching method presented in [Cui, *et. al.*, 2004]. We expect soft pattern rules to offer high coverage in matching against a variety of instances in both training and test phases.

```
(1) Original instances for slot <stime>:
    Time : <stime> 2:30 PM </stime>
     … will be at <stime> 3 pm </stime> …
    …Friday, February 17 <stime> 12:00pm </stime> - 1:00pm
    … will be at <stime> 4pm </stime> , Monday, Feb. 27 …
   Time: <stime> 12:00 PM </stime> - 1:30 PM
(2) Substituted instances:
   Time : <stime> NP_Time </stime>
   VPact_be at <stime> NP_Time </stime>
   NP_Day , NP_Date <stime> NP_Time </stime> - NP_Time
   VPact_be at <stime> NP_Time </stime> , NP_Day , NP_Date
   Time : <stime> NP_Time </stime> - NP_Time
```

Figure 7.2 Illustration of generalizing instances

For each type of tagged slot ($Slot_0$) such as the "*stime*" in Figure 7.2, we accumulate all the tagged instances and align them according to the positions of $Slot_0$. As a result, we obtain a virtual vector *Pa* representing the contextual soft pattern rule as:

$$<Slot_{-k}, \ldots, Slot_{-2}, Slot_{-1}, Slot_0, Slot_1, Slot_2, \ldots, Slot_k : Pa> \qquad (7.2)$$

where $Slot_i$ is a vector of tokens occurring in that slot with their probabilities of occurrences:

$$<(token_{i1}, weight_{i1}), (token_{i2}, weight_{i2}) \ldots.(token_{im}, weight_{im}): Slot_i> \qquad (7.3)$$

where $token_{ij}$ denotes any word, punctuation, syntactic or semantic tags contained in $Slot_i$; and $weight_{ij}$ gives the proportion of occurrences of the j^{th} token to the i^{th} slot. $weight_{ij}$ can be expressed as the conditional probability of the token occurring in that slot. Thus it can be approximated by:

$$\Pr(token_{ij} \mid Slot_i) = \frac{f(token_{ij})}{\sum_{s=1}^{m} f(token_{is})} \qquad (7.4)$$

where *f(token$_{is}$)* stands for the number of occurrences of *token$_{is}$* within *Slot$_i$*. Figure 7.3 shows the generated soft pattern rules for the examples given in Figure 7.2.

(1) Training instances:
Time : <stime> NP_Time </stime>
VPact_be at <stime> NP_Time </stime>
NP_Day , NP_Date <stime> NP_Time </stime> - NP_Time
VPact_be at <stime> NP_Time </stime> , NP_Day , NP_Date
Time : <stime> NP_Time </stime> - NP_Time

(2) Soft pattern rules based on the instances:

……	<Slot$_{-2}$>	<Slot$_{-1}$>	<Slot$_0$>	<Slot$_1$>	…...
	Time 0.4	: 0.4	NP_Time 1	- 0.67	
	VPact_be 0.4	at 0.4		, 0.33	
	, 0.2	NP_Date 0.2			

Figure 7.3 An excerpt of soft pattern rules

What results from the generalization process is a virtual vector *Pa* representing the soft pattern rule. The soft pattern vector *Pa* is then used to compute the degree of match for the unseen instances. The unseen instances are also first pre-processed with the identical procedures as outlined in Section 7.3. Using the same window size *k*, the token fragment *S* surrounding the potential slot (noun phrase) is derived:

$$<token_{-k}, ..., token_{-2}, token_{-1}, Potential\_Slot, token_1, token_2, ..., token_k: S> \qquad (7.5)$$

The degree of match for the unseen instance against the soft pattern rule is measured by the similarity between the vector *S* and the virtual soft pattern vector *Pa*. In particular, the match degree is the combination of the individual slot content similarities and the fidelity degree of sequences measured by a bigram model. The final pattern match weight is computed as following:

$$Pattern\_weight = \frac{Pa\_weight_{Slots} \times Pa\_weight_{Seq}}{fragment\_length} \qquad (7.6)$$

where the *fragment_length* is the normalization factor which is usually set to the length of the context length. The first part of *Pattern_weight* is the match degree of individual slots Pa_Weight_{Slots} and it is computed as:

$$Pa\_weight_{Slots} = \Pr(S \mid Pa) = \prod_{i=-w}^{w} \Pr(token_i \mid Slot_i) \tag{7.7}$$

The calculation of Pa_Weight_{Slots} assumes that all slots are independent to each other. $Pr(token_i|Slot_i)$ is calculated by Equation (7.4). We can see Equation (7.7) is very flexible in matching the soft patterns because it considers only individual slots. Even if some slots are missing, it still can give a similarity measure to the test instance.

The second part of *Pattern_weight* considers the sequence of tokens, to filter out unlikely token sequences to increase precision. We adopt a bigram model to formulate this sequence measure. Specifically, given a token sequence *T*, we calculate the conditional probability of *Pr(T|Pa)* which models how likely the sequence occurs according to the underlying soft pattern rules. We calculate the sequence probability for the left and the right sequences starting from the central tagged slot. The probability of the right sequence is calculated as follows:

$$Pr(right\_seq|Pa) = Pr(token_0, token_1, token_2 \ldots token_w|Pa)$$

$$= P(token_0)P(token_1|token_0) \ldots P(token_w|token_{w-1}) \tag{7.8}$$

where $P(token_i|token_{i-1})$ is estimated by counting the occurrences of the bigram $<token_{i-1}token_i>$ and the unigram $token_{i-1}$ as:

$$P(token_i \mid token_{i-1}) = \frac{f(< token_{i-1}token_i >)}{f(token_{i-1})} \tag{7.9}$$

The process for calculating the probability of the left sequence is formally identical as the following:

$$Pr(left\_seq|Pa) = Pr(token_{-1}, token_{-2}, token_{-3}...token_{-w}|Pa)$$

$$= P(token_{-1})P(token_{-2}|token_{-1})...P(token_{-w}|tokenw_{-w+1}) \quad (7.10)$$

In addition, $P(token_0)$ and $P(token_{-1})$ can be estimated based on the proportion of occurrences in the whole cluster of instances. The sequence weight of the token vector denoted by Pa_Weight_{seq} consists of the weights of its left sequence and right sequence which are calculated by the following equation:

$$Pa\_weight_{Seq} = (1-\alpha)\cdot \Pr(left\_seq \mid Pa) + \alpha \cdot \Pr(right\_seq \mid Pa) \quad (7.11)$$

Based on our observations, the right hand context of the central slot is more important in indicating an information extraction pattern rule, thus we set α to 0.7.

When applying soft pattern rules to automatically tag the training instances during the bootstrapping process, we assign a target tag to each potential slot whose soft pattern rule gives the highest score above a pre-defined threshold according to Equation (7.6).

7.5 Hard Pattern Rule Induction by GRID

We employ the hard pattern rule induction algorithm, GRID, to generalize the hard pattern rule over all instances hand-tagged by users and automatically annotated by the soft pattern rules. GRID generates a pattern rule $r_k(\underline{f})$ by adding slot features into the feature set $\underline{f}$. Differing from what we discussed in Chapter 4, we use a modified *Laplacian* expected error (*Laplacian'*) to define the quality of the hard pattern rule as following:

$$Laplacian'(r_k(\underline{f})) = \frac{n_k + 1}{n_k + p_{k1} + 0.7 \times p_{k2} + 1} \quad (7.12)$$

where p_{k1} denotes the number of instances covered by rule $r_k(\underline{f})$ in the manually annotated set, and p_{k2} denotes the number of instances covered by the rule $r_k(\underline{f})$ in the automatically annotated set. n_k is the number of negative examples or errors covered by the rule. We consider all the manually annotated instances as correctly tagged and thus we put more weight to them than those automatically tagged set. We use GRID to generate pattern rules that cover all seed instances and discard some pattern rules generated from the automatically tagged instances whose *Laplacian'* value is greater than a preset threshold.

7.6 Cascading Matching of Hard and Soft Pattern Rules During Testing

After the bootstrapping rule induction process, we obtain the sets of hard and soft pattern rules. We apply both sets of pattern rules in a cascaded way to assign appropriate tags to potential slots in new instances. The tag assigned to the given test instance t is selected by:

1) tag_g matched by GRID $rule_g$;
2) If not matched by any GRID rule,

$$tag_i \quad \arg\max_{Pa_i \in Pa} \Pr(t \mid Pa_i) > \theta$$

We apply the high-precision hard pattern rules generated by GRID first. In this case, we assign tag_g to the instance if it matches $rule_g$. In order to increase the coverage by the hard pattern rules, we allow up to one shift in context vectors of new test instances when matching against the hard pattern rules.

For the remaining test instances that are not matched by any of the hard pattern rules, we score them using the soft pattern rules. A test instance is assigned tag_i if it has the highest conditional probability of having t given the soft pattern rule i (represented by vector Pa_i) which is greater than a pre-defined threshold θ among all the soft pattern rules.

7.7 Experimental Evaluation

To verify the generality and effectiveness of our bootstrapping framework, we conduct two experiments on both free and semi-structured texts. In our supervised IE system using GRID in Chapter 4, we perform some trial experiments for examining the effect of choosing the different context length *k*. We find the IE performance became stable when the context length reaches 4. As such, we set the context length *k* to 4 for all experiments afterwards.

7.7.1 Results on Free Text Corpus

The first evaluation is conducted on the terrorism domain using the MUC-4 free text corpus [MUC-4, 1992]. We employ the same evaluation measures as that in [Riloff, 1996; Xiao, *et al.,* 2003]. The extracted target slots are “perpetrator” (Perp.), “victim” (Vic.) and “target” (Tar.). We vary the number of the human-annotated instances by randomly selected to be used in IE learning from the 772 relevant document set (the standard training documents for MUC-4 plus TST1 and TST2 tasks) used in supervised IE learning. The manual annotation is guided by the associated answer keys given in the MUC-4 corpus. During testing, we use the 100 texts composing 25 relevant and 25 irrelevant texts from TST3 test set, plus 25 relevant and 25 irrelevant texts from the TST4 test set.

As discussed in Section 7.2, we repeat the automated annotation process several times ($i \geq 1$ in Figure 7.1). To examine the variation of performance along with the changing of the number of iterations, we plot the average F_1 measures of the three target slots against the iteration number (see Figure 7.4). We also varied the number of manually tagged instances that are utilized as seed instances for starting the bootstrapping process. As can be seen in Figure 7.4, the results improve as the number of iterations is increased. The system achieves a steady performance when the number of iterations reaches 4. Accordingly in the following experiments, we present the system's performance based on 4 bootstrapping iterations.

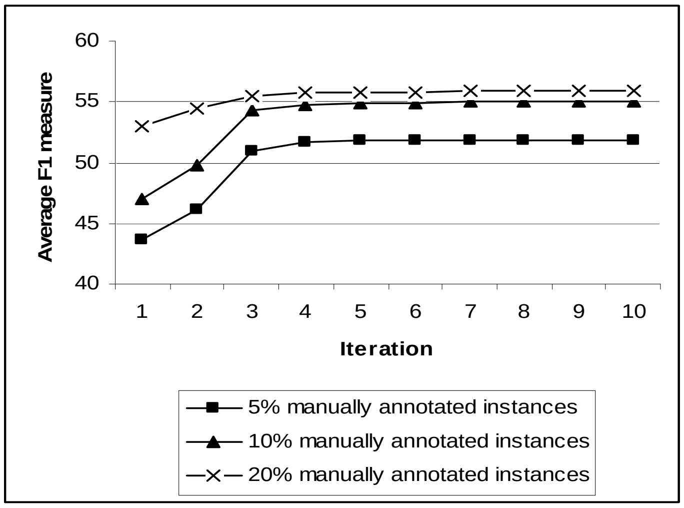


Figure 7.4 Effect of the number of iterations

	Perp.	Vic.	Tar.
5%(SP)	36 (42/32)	45 (49/42)	42 (47/38)
5%(GRID)	34 (35/33)	44 (40/49)	39 (36/43)
5%(SP+GRID)	47 (49/45)	58 (59/57)	50 (50/50)
10%(SP)	38 (45/33)	46 (51/42)	45 (49/42)
10%(GRID)	37 (39/35)	46 (41/52)	44 (41/47)
10%(SP+GRID)	50 (53/47)	61 (63/59)	53 (52/54)
20%(SP)	40 (46/35)	48 (54/43)	47 (50/44)
20%(GRID)	40 (41/39)	47 (44/50)	47 (45/49)
20%(SP+GRID)	51 (52/50)	62 (63/61)	54 (55/53)
AutoSlog-TS	38 (53/30)	48 (62/39)	47 (58/39)
supervised(GRID)	52 (48/57)	62 (58/67)	56 (51/62)

Results presented in terms of F_1(recall/precision).

Table 7.2 Results on free text domain

Table 7.2 shows the performance of the system on the test data in terms of F_1-measure (with recall/precision value in the brackets) using various amounts of manually tagged data after 4 iterations. To demonstrate the effectiveness of the combination of hard and soft pattern rules, we also run 4 iterations using only soft pattern rules (SP) or only GRID rules.

From Table 7.2, we can draw the following conclusions:

(a) The cascaded learner by combining SP and GRID outperforms the learner SP or GRID alone. The soft pattern learner (SP) alone cannot achieve good precision while the hard pattern learner (GRID) alone cannot achieve high recall with a small set of hand-annotated instances.

(b) Compared with another weakly supervised IE system on the same domain, AutoSlog-TS, our cascaded learner outperforms it with the use of only 5% of the manually tagged instances.

(c) As the percentage of hand-annotated instances increases from 5% to 20%, the performance of the cascaded learner (GRID+SP) increases steadily, indicating that the bootstrapping process is stable and consistent.

(d) With 20% of hand-tagged training instances, the performance of the cascaded learner approaches that of the fully supervised IE tagger. When more manually tagged instances (>20%) are used, we found that the performance of the cascaded learner becomes steady.

(e) By observing the automatically tagged instances by the soft pattern rules, we found that about 75% instances are correctly annotated in the first and second iteration. The percentage of correctly tagged instances by soft pattern rules increased to 90% after we run the bootstrapping process 4 times. The increase of the percentage of correctly tagged instances verifies that our automated annotation could provide relatively accurate training instances for later rule induction.

(f) The cascaded learner GRID+SP outperforms GRID_CoTrain which is described in Chapter 6 with less human labor (from 20% down to 5%-10%).

Our system missed some cases that require deeper NLP analysis. For example, given a test sentence "THEY ARE THE TOP MILITARY AND POLITICAL FIGURES IN ALFREDO CRISTIANI'S ADMINISTRATION." The system can not identify "ALFREDO CRISTIAN'S ADMINISTRATION" as the "perpetrator". If we can associate the previously found "perpetrator" (maybe located far away) to "they", then we may infer that the "ALFREDO CRISTIAN'S ADMINISTRATION" is the "perpetrator" too.

7.7.2 Results on Semi-structured Corpus

The second experiment is conducted on the semi-structured text documents. We use the CMU seminar announcements [www$_3$] for the evaluation. The IE task for this domain is to extract the entities of "speaker" (SP), "location" (LOC), "starting time" (ST), and "ending time" (ET) from a seminar announcement. There are 485 seminar announcements. In the supervised IE experiments, we perform 5 runs and in each run we use one half for training and the other half for testing. Similarly, to evaluate our weakly supervised learning framework, we carry out 5 trials as well. In each run, we vary the percentage of manually annotated instances for training in the supervised experiments. Table 3 shows the performance (the average F_1 measure and recall/precision for 5 runs) of the system with different percentage of manually tagged instances used to start the training. We also compare the performances between the single learners and the cascaded learner. All results are based on 4 bootstrapping iterations.

	SP	LOC	ST	ET
5%(SP)	70 (74/66)	65 (70/61)	94 (95/93)	90 (93/88)
5%(GRID)	68 (65/72)	61 (59/64)	93 (91/94)	89 (86/92)
5%(SP+GRID)	82 (83/81)	73 (74/72)	98 (98/98)	94 (96/92)
10%(SP)	72 (75/70)	68 (72/64)	96 (96/95)	93 (94/92)
10%(GRID)	72 (67/77)	67 (63/72)	95 (94/96)	93 (91/96)
10%(SP+GRID)	84 (84/83)	75 (75/74)	99 (99/99)	95 (97/94)
20%(SP)	75 (77/74)	71 (75/67)	97 (97/97)	95 (96/95)
20%(GRID)	75 (69/82)	71 (66/77)	97 (95/99)	95 (94/96)
20%(SP+GRID)	85 (85/85)	76 (76/75)	99 (99/99)	96 (97/95)
supervised (GRID)	86 (84/88)	76 (73/80)	99 (99/100)	96 (95/97)

Results presented in terms of F_1(recall/precision).

Table 7.3 Results on semi-structured corpus

From Table 7.3, we have the following observations:

(a) The cascaded learner (GRID+SP) with two pattern learners significantly outperforms the learner SP or GRID alone as the case for the free text corpus. With 10% of hand-tagged instances, GRID+SP can approach the performance of the fully supervised IE tagger. Also the performance of the cascaded learner increases steadily with the number of hand-tagged instances increased from 5% to 20%.

(b) We also found that with more hand-annotated instances (>20%), the performance of the bootstrapping system with cascading use of SP and GRID becomes stable and consistent.

(c) We also randomly checked some automatically tagged instances by the soft pattern rules after each iteration and found that about 90% of the instances are correctly tagged by the soft pattern rules.

The lower performance of our system on the “location” slot is mainly due to the use of a general named entity recognizer which is good at identifying common locations such as cities, mountains *etc.*. In seminar announcements, many locations are room numbers such as “WeH 8220” thus we missed out some seminar venues.

7.8 Summary

In this Chapter, we presented a novel bootstrapping approach for information extraction by cascading use of soft and hard pattern rules. Our framework takes advantage of high-recall by soft pattern rules and high-precision of hard pattern rules. We used soft pattern rules to automatically annotate more training instances so as to provide a more comprehensive basis for hard pattern rule induction. The integration of soft pattern matching in the extraction phase also brings in more target entities from the test instances that are likely to be missed by hard pattern matching [Xiao, *et. al.* 2004]. With much less manual labor (only need 5%-10% annotated data compared to the fully supervised version), the proposed bootstrapping system approaches the performance obtainable by the fully supervised learning on both the semi-structured and free text corpora.

Chapter 8

Conclusions

8.1 Summary of This Thesis

This thesis presents a general pattern rule learning algorithm GRID which is designed for information extraction tasks but can be applied to other tasks such as definitional question answering and story boundary detection in news video. Different from most existing pattern rule learning systems for information extraction, GRID utilizes global statistical information of the training instances to kick off the rule induction process. GRID also uses named entities as semantic constraints and applies lexical chaining for pattern rule generalization.

In order to realize weakly supervised learning based on GRID learner, this research extends GRID with some other machine learning approaches such as co-training (CoTrain), active learning and soft pattern matching (SP). With much less human manually tagging labor, the weakly supervised paradigms, GRID_CoTrain and GRID+SP, can obtain comparable performance to the fully supervised learning version of GRID.

In the following sub-sections, we discuss some issues in information extraction and present directions for future work.

8.2 Some Issues in IE

8.2.1 Slot-based *vs* tag-based IE

We may consider a pattern rule as composed of a left hand side, containing a pattern of conditions on a sequence of adjacent words/tokens or syntactic/semantic units, and a right hand side that is an action to insert an XML tag in the texts. For example, a rule is to insert both the opening tag of "<*speaker*>" and the closing tag of "</*speaker*>" around a person of "Mr. Smith". We call this kind of rule "slot-based". Currently, most pattern rule learning systems such as WHISK, RAPIER, GRID *etc.* for IE are slot-based, *i.e.* to insert both side tags beside the extracted target. In (LP)^2, the author proposed a "tag-based" IE approach in which each rule inserts a single XML tag, *e.g.* inserting <*speaker*> before "Mr. Smith". This separation of recognition of tags (*i.e.* <*speaker*> is recognized independently from </*speaker*>) allows higher recall. This is because the separation of opening and closing tag allows easier generalization in rule writing. For example in a slot-oriented rule learning strategy in order to learn patterns equivalent to the regular expression ("at" | "starting from") *digit* ("pm" | "am") four examples will be needed: "at" + "pm", "at" + "am", "starting from" + "pm", "starting from" + "am". In tag-based learning strategy, it just need two, *e.g.* "at" + "pm" and "starting from" + "am", because the algorithm will write two rules for <*stime*> ("at" and "starting from") and two for </*stime*> ("am" and "pm"). Thus tag-based learning systems require less training examples than the slot-based ones. On the other hand, the tag-based pattern rule learning systems need additional contextual rules to improve the tagging performance. For example rules like "put a </*organization*> tag after a capitalized word and before a lowercase word" will never be selected as a good rule because it will tag as

</organization> every sequence of capitalized word + lowercase word. But this rule is a good rule if its use is constrained by the presence of an *<organization>* tag in the immediate left context. This means that the rule becomes "end an organization name at the first lowercase word", *i.e.* "after an *<organization>* tag put an *</organization>* tag right after the first capitalized word followed by a lowercase word".

Slot-based pattern rule induction usually requires more training examples than tag-based pattern rule induction does. Tag-based pattern rule induction requires additional contextual rules to link the opening tag and closing tag. In this thesis, we only conduct the slot-based rule induction in GRID. Extending GRID to tag-based rule induction could be one of the future works.

8.2.2 Portability of IE systems

One difficulty in information extraction is the cost of developing extraction systems for new domain, *i.e.* the portability problem in IE. Although system designers have been quite successful in separating a domain-independent core from domain-specific knowledge sources, the cost of customization remains considerable. Several knowledge sources need to be adapted: the largest are typically the patterns for identifying the events of interest; specialized rules for filling the template slots and the inference strategies; an additional lexicon or thesaurus will also be required. The cost is particularly high if these knowledge sources need to be manually built by the external experts who are expensive both in terms of time and cost. Therefore, there has been a push towards facilitating greater user customization, through visual interfaces, example-based methods, and corpus-trained systems. In this thesis, all the IE systems for automatically learning pattern extraction rules are corpus-trained which are categorized into learning from annotated samples and learning from un-annotated samples with a small set of tagged instances. Another concern for separating the domain-independent core from the domain-specific knowledge sources in pattern rule learning systems is to divide the pattern rules into domain-dependent and domain-independent portions. The domain-independent part of the domain-phase consists of a number of rules that one might characterize as parameterized macros. The rules cover various syntactic constructs at a relatively coarse granularity, the objective being to construct the appropriate predicate-argument relations for verbs that behave according to that pattern. The domain-dependent rules comprise the cluster of parameters that must be instantiated by the "macros" to produce the actual rules. These domain-dependent rules specify precisely which verbs carry the domain-relevant information, and specify the domain-dependent restrictions on the arguments, as well as the semantics of the rule.

The system described in this thesis, GRID, is a data-driven machine learning IE system. It can be easily ported to other domains such as the bioinformatics. However, in order to achieve high effectiveness, some domain-dependent lexicon or semantic constraints are needed while porting GRID to other domains.

8.2.3 Using Linguistic Information

Information extraction can be regarded as one of the direct applications of natural language processing technologies. Traditional NLP techniques such as syntactic analysis, semantic analysis and discourse-level analysis are widely applied in many information extraction systems. But how much linguistic information do we need for realizing an efficient information extraction system? The answer is dependent on the text genre. [Krupka, 1995] did some experiments using SRV system in this vein and found that providing linguistic information to SRV yielded little benefit. In RAPIER [Califf, 1998], the author also pointed that the use of an external linguistic dictionary, WordNet, did not help in improving the system performance. Both SRV and RAPIER were tested on semi-structured documents, *i.e.* the job listing corpus. We drew the same conclusion in GRID experiments when we tried to perform full-parsing for the online semi-structured documents. But for the MUC text genre (free-text with more grammatical structures), deep NLP understanding is useful to improve the system performance as we can see in the GRID experiments with the comparison between shallow parsing and full parsing in MUC-4 domain. However, it is not easy to obtain robust deep NLP analysis such as co-referencing resolution and discourse analysis. This may affect the performance of the IE systems on free text domains.

Although deep natural language understanding can help to improve information extraction performance in free text genre, experience from semi-structured documents seem to suggest that useful entities can be gleaned from a semi-structured document without deep understanding it. Therefore information extraction technology can be effectively applied to semi-structured and structured text corpora, especially the web-based text documents, without the need to deal with deep linguistic analysis.

8.3 Future Work

8.3.1 IE from multi-event document

In this thesis, most documents are single event-based except for a few documents in the MUC-4 corpus. Single event per document means that each document should produce a single filled template or case frame. For example, in the AUSTIN job postings domain in our experiment, a single document only contains one job posting. We therefore just employed some heuristic rules to extract the slot values for the template from a document and did not do much research on how to determine whether the extracted slot values might belong to different events. There are several issues on the future work of IE from multi-event document.

First, the system needs to recognize the need to create multiple templates. One way to do this is to recognize when slots which should have only one filler have multiple potential filler extractions. This could be very effective in a domain such as the rental ads. It is less so in a domain like the job postings where almost any slot can have multiple fillers, such as the job titles may appear in multiple variations for the same job. Another concern would be to recognize the typical ordering of slots, or whether there are typical orderings. For example, if all fillers for slot A typically come before all fillers for slot B. Then in a document with fillers for slot A followed by fillers for slot B and followed by more fillers for slot A would provide clues that multiple templates should be created. Another option would be to learn text segmentation rules to recognize the transition from one event to another.

Second, there is a need to associate the correct filler with each of the multiple templates. For some domains this may be facilitated by learning rules which extract fillers for multiple slots. For examples, for domains like job listings, it may be possible to simply divide the document into sections describing each separate case and to apply rules only within each section.

8.3.2 IE from Bioinformation

The explosive growth of textual material in the biology area means that no one can keep up with what is being published. There is too much new, complex and non-standardised terminology appearing in publications everyday. One effective approach to manage those key terms efficiently is to extract them using information extraction techniques and put them into databases for indexing or querying purposes. The information extraction techniques discussed in this thesis can provide the basic tools for bioinformation extraction, such as new protein and virus names extraction. Also the template extraction in information extraction can be mapped to medicine domain. For example, scientists working on drug discovery have an ongoing interest in reactions catalyzed by enzymes in metabolic pathways. These reactions may be viewed as a class of relation extraction, like corporate management succession events, in which various classes of entities such as enzymes, compounds with attributes such as names, concentrations are related by participating in the event in particular roles such as substrate, catalyst, product *etc.*. Thus the techniques in relation extraction in information extraction can be extended to the medicine domain smoothly. To cope with the bio-text, we may also need the domain knowledge such as the medical lexicon, specific semantic classes and the concept hierarchy *etc.* for bio-domain.

8.3.3 IE and Text Mining

Text mining is concerned with applying data mining techniques to unstructured text. Data mining assumes that the information to be "mined" is already in the form of a relational database. Unfortunately, for many applications, available electronic information is in the form of unstructured natural language documents rather than structured databases. Consequently, text mining is evolved to discover useful knowledge from unstructured text. Information extraction can play obvious role in text mining. Natural language information extraction methods can transform a corpus of textual documents into a more structured database. On the other hand, the rules mined from a database can be used to predict additional information to extract from future documents, thereby improving the recall of IE [Nahm and Mooney, 2000]. Thus IE and text mining can be integrated in which they are mutually beneficial to each other as indicated in Nahm and Mooney [2000]. We hope the IE techniques described in this thesis can be helpful in text mining applications.

Bibliography

B. Adelberg. 1998. NoDoSE: A Tool for Semi-Automatically Extracting Structured and Semi-Structured Data from Text Documents. *SIGMOD Record*, 27, 2 (1998), 283-294.

E. Agichtein and L. Gravano. 2000. Snowball: Extracting Relations from Large Plain-Text Collections. *Proceedings of the 5th ACM International Conference on Digital Libraries.* 2000.

D. E. Appelt and D. J. Israel. 1999. Introduction to Information Extraction Technology. *The Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99) Tutorial*. Stockholm, Sweden. 1999.

A. Blum and T. Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-training. *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT-98).*

R. R. Bouckaert. 2002. Low Level Information Extraction, a Bayesian Network Based Approach. *The 19th International Conference on Machine Learning Workshop on Text Learning (TextML-2002).*

D. M. Bikel, R. Schwartz and R. M. Weischedel. 1999. An Algorithm that Learns What's in a Name. *Machine Learning*, 34, pages 211-231. 1999.

M. E. Califf. 1998. Relational Learning Techniques for Natural Language Information Extraction. *PhD dissertation,* University of Texas at Austin. 1998.

M. E. Califf and R. J. Mooney. 1999. Relational Learning of Pattern-Match Rules for Information Extraction. *Proceedings of the 16th National Conference on Artificial Intelligence. (AAAI-99),* pages 328-334.

C. Cardie. 1997. Empirical Methods in Information Extraction. *AI Magazine*, 18(4), pages 65–79. 1997.

J. Y. Chai and A. W. Biermann. 1997. Corpus Based Statistical Generalization Tree in Rule Optimization. *Proceedings of the 5th Workshop on Very Large Corpora (WVLC-5)*, pages 81-90. 1997.

L. Chaisorn, T. –S. Chua, C. –H. Lee and Q. Tian. 2004. A Hierarchical Approach to Story Segmentation of Large Broadcast News Video Corpus. *2004 IEEE International Conference on Multimedia and Expo (ICME-04)*.

H. L. Chieu and H. T. Ng. 2002a. A Maximum Entropy Approach to Information Extraction from Semi-Structured and Free Text. *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-02)*, pages 786-791.

H. L. Chieu and H. T. Ng. 2002b. Named Entity Recognition: A Maximum Entropy Approach Using Global Information. *Proceedings of 19th International Conference on Computational Linguistics (COLING-02)*, pages 190-196. 2002.

H. L. Chieu, H. T. Ng and Y. K. Lee. 2003. Closing the Gap: Learning-Based Information Extraction Rivaling Knowledge-Engineering Methods. *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 216-223. 2003.

T.-S. Chua and J. Liu. 2002. Learning Pattern Rules for Chinese Named Entity Extraction. *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-02)*, pages 411-418, Edmonton, Canada, Jul/Aug 2002.

T. –S. Chua, Y. Zhao, L. Chaisorn, C. –K. Koh, H. Yang, H. Xu and Q. Tian. 2003. TREC 2003 Video Retrieval and Story Segmentation Task at NUS PRIS. *In the notebook of the 12th Text REtrieval Conference Video Workshop (TRECVID 2003)*, Maryland, USA. 2003.

F. Ciravegna. 2000. Learning to Tag for Information Extraction. *ECAI Workshop on Machine Learning for Information Extraction*, 2000.

F. Ciravegna. 2001. Adaptive Information Extraction from Text by Rule Induction and Generalisation. *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01).*

D. Cohn, L. Atlas and R. Ladner. 1994. Improving Generalization with Active Learning. *Machine Learning*, 15(2), pages 201-221.

M. Collins and Y. Singer. Unsupervised Models for Named Entity Classification. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora.* 1999.

H. Cui, M. –Y. Kan and T. –S. Chua. 2004. Unsupervised Learning of Soft Patterns for Definitional Question Answering. *Proceedings of the 13th World Wide Web Conference (WWW-04).* 2004.

A. Douthat. 1998. The Message Understanding Conference Scoring Software User's Manual. *Proceedings of the 7th Message Understanding Conference*. 1998.

D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, K. Ng, Y. Quass and R. D. Smith. 1999. Conceptual-Model-Based Data Extraction from Multiple-Record Web Pages. *Data and Knowledge Engineering*, 31, 3 (1999), 227-251.

D. Freitag. 1998. Information Extraction from HTML: Application of a General Learning Approach. *Proceedings of the 15th Conference on Artificial Intelligence (AAAI-98)*, pages 517-523.

D. Freitag and A. McCallum. 1999. Information Extraction with HMMs and Shrinkage. *AAAI-99 Workshop on Machine Learning for Information Extraction.*

D. Freitag and A. McCallum. 2000. Information Extraction with HMM Structures Learned by Stochastic Optimization. *Proceedings of 17th National Conference on Artificial Intelligence (AAAI-00)*, pages 584-589.

Y. Freund, H. S. Seung, E. Shamir and N. Tishby. 1997. Selective Sampling using the Query by Committee Algorithm. *Machine Learning*, 28, pages 133-168. 1997.

R. Grishman. 1997. Information Extraction: Techniques and Challenges. *Information Extraction (International Summer School SCIE-97)*, ed. Maria Teresa Pazienza, Spring-Verlag, 1997.

H. Han, C. Giles, E. Manavoglu, H. Zha, Z. Zhang and E. Fox. 2003. Automatic Document Meta-data Extraction using Support Vector Machines. *Proceedings of Joint Conference on Digital Libraries 2003*.

C. –N. Hsu and M. –T. Dung. 1998. Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web. *Information Systems,* 23(8):521-538, 1998.

S. B. Huffman. 1995. Learning Information Extraction Patterns from Examples. *Proceedings of the 1995 IJCAI Workshop on New Approaches to Learning for Natural Language Processing*, pages 246-260.

R. Jones, R. Ghani, T. Mitchell and E. Riloff. 2003. Active Learning for Information Extraction with Multiple View Feature Sets. *ECML-03 Workshop on Adaptive Text Extraction and Mining.* 2003.

M. Kavakec and V. Svatek. 2002. Information Extraction and Ontology Learning Guided by Web Directory. *ECAI 2002 Workshop on Natural Language Processing and Machine Learning for Ontology Engineering*. 2002.

J. –T. Kim and D. I. Moldovan. 1995. Acquisition of Linguistic Patterns for Knowledge-Based Information Extraction. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 7, No. 5, October, pages 713-724, 1995.

G. Krupka. 1995. SRA: Description of the SRA System as Used for MUC-6. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 221-237.

N. Kushmerick. 1997. Wrapper Induction for Information Extraction. *PhD dissertation*, University of Washington. 1997.

N. Kushmerick, D. Weld and R. Doorenbos. 1997. Wrapper Induction for Information Extraction. *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97),* pages 729-737.

A. H. F. Laender, B. A. Ribeiro-Neto, A. S. Da Silva. 2002a. DEByE-Data Extraction by Example. *Data and Knowledge Engineering,* 40(2): 121-154. 2002.

A. H. F. Laender, B. A. Ribeiro-Neto, A. S. Da Silva and J. S. Teixeira. 2002b. A Brief Survey of Web Data Extraction Tools. *SIGMOD Record*. Vol. 31, No. 2, June, 2002.

S. Lappin and H. J. Leass. 1994. An Algorithm for Pronominal Anaphora Resolution. *Computational Linguistics*, Volume 20:4, pages: 535-561.

D. D. Lewis and J. Catlett. 1994. Heterogeneous Uncertainty Sampling for Supervised Learning. *Proceedings of the 11th International Conference on Machine Learning (ICML-94)*, pages 148-156. 1994.

A. McCallum, D. Freitag and F. Pereira. 2000. Maximum Entropy Markov Models for Information Extraction and Segmentation. *Proceedings of the 7th International Conference on Machine Learning (ICML-00)*, pages 591-598.

R. S. Michalski. 1983. A Theory and Methodology of Inductive Learning. *Artificial Intelligence*, 20, pages 111-161, 1983.

G. A. Miller, *et al.* Introduction to WordNet: An Online Lexical Database.

T. M. Mitchell. 1997. Machine Learning. McGraw Hill. 1997.

A. Moschitti, P. Morărescu and S. Harabagiu. 2003. Open Domain Information Extraction via Automatic Semantic Labeling, *Proceedings of the 2003 Special Track on Recent Advances in Natural Language at the 16th International FLAIRS Conference*, May 11-15, 2003, St. Augustine, Florida.

MUC-3, 1991. *Proceedings of the Third Message Understanding Conference*, Morgan Kaufmann Publishers, 1991.

MUC-4, 1992. *Proceedings of the Fourth Message Understanding Conference*, Morgan Kaufmann Publishers, 1992.

MUC-5, 1993. *Proceedings of the Fifth Message Understanding Conference*, Morgan Kaufmann Publishers, 1993.

MUC-6, 1995. *Proceedings of the Sixth Message Understanding Conference*, Morgan Kaufmann Publishers, 1995.

MUC-7, 1998. *Proceedings of the Seventh Message Understanding Conference*, Fairfax, VA, 1998. http://www.itl.nist.gov/iaui/894.02/related_projects/muc/.

S. Muggleton. 1992. Inductive Logic Programming. Academic Press, New York, NY.

S. Muggleton. 1995. Inverse Entailment and Progol. *New Generation Computing, Special Issue on Inductive Logic Programming*, 13, 1995.

I. Muslea. 1999. Extraction Patterns for Information Extraction Tasks: A Survey. *The AAAI-99 Workshop on Machine Learning for Information Extraction.*

I. Muslea, S. Minton and C. Knoblock. 1999. A Hierarchical Approach to Wrapper Induction. *Proceedings of the Third International Conference on Autonomous Agents (AA-99)*, pages 190-197.

I. Muslea, S. Minton and C. Knoblock. 2003. Active Learning with Strong and Weak Views: a Case Study on Wrapper Induction. *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 415-420. 2003.

U. Y. Nahm and R. J. Mooney. 2000. A Mutually Beneficial Integration of Data Mining and Information Extraction. *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000)*, pages 627-632. 2000.

U. Y. Nahm and R. J. Mooney. 2001. Mining Soft Matching Rules from Textual Data. *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 979-986.

G. Paliouras, V. Karkaletsis, G. Petasis and C. D. Spyropoulos. 2000. Learning Decision Trees for Named-Entity Recognition and Classification. *ECAI workshop on Machine Learning for Information Extraction.*

D. Pierce and C. Cardie. 2001. Limitations of Co-Training for Natural Language Learning from Large Datasets. *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-01).*

J. Quinlan. 1990. Learning Logical Definitions from Relations. *Machine Learning*, 5(3), pages 239-266, 1990.

E. Riloff. 1993. Automatically Constructing a Dictionary for Information Extraction. *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-93)*, pages 811-816. 1993.

E. Riloff. 1996. Automatically Generating Extraction Patterns from Untagged Text. *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, pages 1044-1049. 1996.

E. Riloff and R. Jones. 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, pages 474-479. 1999.

S. J. Russell and P. Norvig. 2003. Artificial Intelligence: A Modern Approach. Prentice Hall, 2003, 2nd edition.

G. Salton and M. J. McGill. 1983. Introduction to Modern Information Retrieval. McGraw Hill, 1983.

D. Sankoff and J. Kruskal. 1999. Time Wraps, String Edits, and Macromolecules the Theory and Practice of Sequence Comparison. CSLI Publications. 1999.

S. Sekine, R. Grishman and H. Shinnou. 1998. A Decision Tree Method for Finding and Classifying Names in Japanese Texts. *Proceedings of the 6th Workshop on Very Large Corpora.*

S. Soderland, D. Fisher, J. Aseltine and W. Lehnert. 1995. CRYSTAL: Inducing a Conceptual Dictionary. *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95),* pages 1314-1319. 1995.

S. Soderland. 1997a. Learning Text Analysis Rules for Domain-Specific Natural Language Processing. *PhD dissertation*, University of Massachusetts, Amherst. 1997.

S. Soderland. 1997b. Learning to Extract Text-based Information from the World Wide Web. *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 252-254, 1997.

S. Soderland. 1999. Learning Information Extraction Rules for Semi-structured and Free Text. *Machine Learning*, 34, pages 233-272. 1999.

W. M. Soon, H. T. Ng and D. C. Y. Lim. 2001. A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Computational Linguistics (Special Issue on Computational Anaphora Resolution)*, Vol 27, No 4, pages 521-544.

C. A. Thompson, M. E. Califf and R. J. Mooney. 1999. Active Learning for Natural Language Parsing and Information Extraction. *Proceedings of the 16th International Conference on Machine Learning (ICML-99)*, pages 406-414. 1999.

E. M. Voorhees. 2003a. Overview of the TREC 2002 Question Answering Track. *Proceedings of the 11th Text Retrieval Conference (TREC-11)*, 2003.

E. M. Voorhees. 2003b. Draft Overview of the TREC 2003 Question Answering Track. *The 12th Text REtrieval Conference (TREC 2003) Notebook*, pages 14-27. 2003.

J. Xiao, J. Liu and T. –S. Chua. 2002. Extracting Pronunciation-translated Names from Chinese Texts Using Bootstrapping Approach. *First SIGHAN Workshop on Chinese Language Processing, in conjunction with COLING 2002*.

J. Xiao, T. –S. Chua and J. Liu. 2003. A Global Rule Induction Approach to Information Extraction. *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-03)*, pages 530-536. 2003.

J. Xiao, T. –S. Chua and J. Liu. 2004. Global Rule Induction for Information Extraction. *International Journal on Artificial Intelligence Tools.* Dec. 2004.

J. Xiao, T. –S. Chua and H. Cui. 2004. Cascading Use of Soft and Hard Matching Patterns Rules for Weakly Supervised Information Extraction. *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*. 2004.

H. Yang, *et. al.*, 2003. QUALIFIER in TREC 12 QA Main Task, *The 12th Text REtrieval Conference (TREC 2003) Notebook*, pages 54-63. 2003.

R. Yangarber, 2003. Counter-Training in Discovery of Semantic Patterns. *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*, pages 343-350. 2003.

D. Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-95)*, pages 189-196. 1995.

J. M. Zelle, R. J. Mooney and J. B. Konvisser. 1994. Combing Top-down and Bottom-up Methods in Inductive Logic Programming. *Proceedings of 11th International Workshop on Machine Learning (ML-94)*. 1994.

[www$_1$] http://ontology.teknowledge.com/

[www$_2$] http://protege.stanford.edu/

[www$_3$] http://www.isi.edu/info-agents/RISE/repository.html

[www$_4$] ftp://ftp.cs.brown.edu/pub/nlparser/

[www$_5$] http://www-nlpir.nist.gov/projects/tv2003/tv2003.html