INCORPORATION OF CONSTRAINTS TO IMPROVE MACHINE LEARNING APPROACHES ON COREFERENCE RESOLUTION

CEN CEN

(MSc. NUS)

**A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF SCIENCE
SCHOOL OF COMPUTING
NATIONAL UNVIERSITY OF SINGAPORE
2004**

Acknowledgements

I would like to say “Thank You” to everyone who has helped me during the course of the research. Without their support, the research would not be possible.

My first thanks go to thank my supervisor, Associate Professor Lee Wee Sun, for his invaluable guidance and assistance. I am always inspired by his ideas and visions. I cannot thank him enough.

I also want to say thank you to many others - Yun Yun, Miao Xiaoping, Huang Xiaoning, Wang Yunyan and Yin Jun. Their suggestions and concern for me put me always in a happy mood during this period.

Last but not least, I wish to thank my friend in China, Xu Sheng, for his moral support. His encouragement is priceless.

Content

List of Figures

List of Tables

Summary

In this thesis, we utilize linguistic knowledge to improve coreference resolution systems built through a machine learning approach. The improvement is the result of two main ideas: incorporation of multi-level ranked constraints based on linguistic knowledge and conflict resolution for handling conflicting constraints within a set of corefering elements. The method resolves problems with using machine learning for building coreference resolution systems, primarily the problem of having limited amounts of training data. The method provides a bridge between coreference resolution methods built using linguistic knowledge and machine learning methods. It outperforms earlier machine learning approaches on MUC-7 data increasing the F-measure of a baseline system built using a machine learning method from 60.9% to 64.2%.

1. Introduction

1.1. Coreference Resolution

1.1.1. Problem Statement

Coreference resolution is the process of collecting together all expressions which refer to the same real-world entity mentioned in a document. The problem can be recast as a classification problem: given two expressions, do they refer to the same entity or different entities. It is a very critical component of Information Extraction systems. Because of its importance in Information Extraction (IE) tasks, the DARPA Message Understanding Conferences have taken coreference resolution as an independent task and evaluated it separately since MUC-6 [MUC-6, 1995]. Up to now, there have been two MUCs, MUC-6 [MUC-6, 1995] and MUC-7 [MUC-7, 1997] which involve the evaluation of coreference task.

In this thesis, we focus on the coreference task of MUC-7 [MUC-7, 1997]. MUC-7 [MUC-7, 1997] has a standard set of 30 dry-run documents annotated with coreference information which is used for training and a set of 20 test documents which is used in the evaluation. They are both retrieved from the corpus of New York Times News Service and have different domains.

1.1.2. Applications of Coreference Resolution

Information Extraction

An Information Extraction (IE) system is used to identify information of interest from a collection of documents. Hence an Information Extraction (IE) system must frequently extract information from documents containing pronouns. Furthermore, in a document, the entity including interesting information is often mentioned in different places and in different ways. The coreference resolution can capture such information for the Information Extraction (IE) system. In the context of MUC, the coreference task also provides the input to the template element task and the scenario template task. However its most important criterion is the support for the MUC Information Extraction tasks.

Text Summarization

Many text summarization systems include the component for selecting the important sentences from a source document and using them to form a summary. These systems could encounter some sentences which contain pronouns. In this case, coreference resolution is required to determine the referents of pronouns in the source document and replace these pronouns.

Human-computer interaction

Human-computer interaction needs computer system to provide the ability to understand the user’s utterances. Human dialogue generally contains many pronouns and similar types of expressions. Thus, the system must figure out what the pronouns denote in order to "understand" the user's utterances.

1.2. Terminology

In this section, the concepts and definitions used in this thesis are introduced.

In a document, the expressions that can be part of coreference relations are called *markables*. Markable includes three categories: noun, noun phrase and pronoun. A markable used to perform reference is called the *referring expression*, and the entity that is referred to is called the *referent*. Sometimes a referring expression is referred as a referent. If two referring expressions refer to each other, they *corefer* in the document and are called *coreference pair*. The first markable in a coreference pair is called *antecedent* and the second markable is called *anaphor*. When the coreference relation between two markables is not confirmed, the two markables constitute a *possible coreference pair*, and the first one is called *possible antecedent* and the second is *possible anaphor*. Only those markables which are *anaphoric* can be anaphors. All referring expressions referring to the same entity in a document constitute a *coreference chain*. In order to determine a coreference pair, a *feature vector* is calculated for each possible coreference pair. The feature vector is the basis of the classifier model.

For the sake of evaluation, we constructed the system's output according to the requirement of MUC-7 [MUC-7, 1997]. The output is called *responses* and the key file is offered by MUC-7 [MUC-7, 1997] *keys*. A coreference system is evaluated according to three criteria: recall, precision and F-measure [Amit and Baldwin, 1998].

1.3. Introduction

1.3.1. Related Work

In coreference resolution, so far, there are two different but complementary approaches: one is theory-oriented rule-based approach and the other is empirical corpus-based approach.

Theory-oriented Rule-based Model

Theory-oriented rule-based approaches [Mitkov, 1997; Baldwin, 1995; Charniak, 1972] employ manually encoded heuristics to determine coreference relationship. These manual approaches require the information encoded by knowledge engineers: features of each markable, rules to form coreference pairs, and the order of these rules. Because coreference resolution is a linguistics problem, most rule-based approaches more or less employ theoretical linguistic work, such as Focusing Theory [Grosz et al., 1977; Sidner, 1979], Centering Theory [Grosz et al., 1995] and the systemic theory [Halliday and Hasan, 1976]. The manually encoded rules incorporate background knowledge into coreference resolution. Within a specific knowledge domain, the approaches achieve a high precision (around 70%) and a good recall (around 60%). However, language is hard to be captured by a set of rules. Almost no linguistic rule can be guaranteed to be 100% accurate. Hence, rule-based approaches are subject to three disadvantages as follows:

1) Features, rules and the order of the rules need to be determined by knowledge engineers.

2) The existence of an optimal set of features, rules and an optimal arrangement of the rules set has not been conclusively established.

3) A set of features, rules and the arrangement of rules depend much on knowledge domain. Even though a set of features, rules and the arrangement can work well in one knowledge domain, they may not work as well in other knowledge domains. Therefore if the knowledge domain is changed, the set of features, rules and the arrangement of the rules set need to be tuned manually again.

Hence considering these disadvantages, further manual refinement of theory-oriented rule-based models will be very costly and it is still far from being satisfactory for many practical applications.

Corpus-based Empirical Model

Corpus-based empirical approaches aree reasonably successful and achieve a performance comparable to the best-performing rule-based systems for the coreference task's test sets of MUC-6 [MUC-6, 1995] and MUC-7 [MUC-7, 1997]. Compared to rule-based approaches, corpus-based approaches have following advantages:

1) They are not as sensitive to knowledge domain as rule-based approaches.

2) They use machine learning algorithms to extract rules and arrange the rules set in order to eliminate the requirement for the knowledge engineer to

determine the rules set and arrangement of the set. Therefore, they are more cost-effective.

3) They provide a flexible mechanism for coordinating context-independent and context-dependent coreference constraints.

Corpus-based empirical approaches are divided into two groups: one is supervised machine learning approach [Aone and Bennett, 1995; McCarthy, 1996; Soon et al., 2001; Ng and Cardie, 2002a; Ng and Cardie, 2002; Yang et al., 2003], which recasts coreference problem as a binary classification problem; the other is unsupervised approach, such as [Cardie and Wagstaff, 1999], which recasts coreference problem as a clustering task. In recent years, supervised machine learning approach has been widely used in coreference resolution. In most supervised machine learning systems [e.g. Soon et al., 2001; Ng and Cardie, 2002a], a set of features is devised to determine coreference relationship between two markables. Rules are learned from these features extracted from training set. For each possible anaphor which is considered in test document, its possible antecedent is searched for in the preceding part of the document. Each time, a pair of markables is found, it will be tested using those rules. This is called the single-candidate model [Yang et al., 2003]. Although these approaches have achieved significant success, the following disadvantages exist:

Limitation of training data

The limitation of training data is mostly due to training data insufficiency and "hard" training examples.

Because of insufficiency of training data, corpus-based model cannot learn sufficiently accurate rules to determine coreference relationship in test set. In [Soon et al., 2001; Ng and Cardie, 2002a], they used 30 dryrun documents to train their coreference decision tree. But coreference is a rare relation [See Ng and Cardie, 2002]. In [Soon et al., 2001]'s system, only about 2150 positive training pairs were extracted from MUC-7 [MUC-7, 1997], but the negative pairs were up to 46722. Accordingly the class distributions of the training data are highly skewed. Learning in the presence of such skewed class distributions results in models, which tend to determine that a possible coreference pair is not coreferential. This makes the system's recall drop significantly. Furthermore, insufficient training data may result in some rules being missed. For example, if within a possible coreference pair, one is another's appositive, the pair should be a coreference pair. However, appositives are rare in training documents, and it cannot be determined easily. As a result, the model may not include the appositive rule. This obviously influences the accuracy of coreference system.

During the sampling of positive training pair, if the types of noun phrases are ignored, it would result in "hard" training example [Ng and Cardie, 2002]. For example, the interpretation of a pronoun may be dependent only on its closest antecedent and not on the rest of the members of the same coreference chain. For proper name resolution, the string matching or more sophisticated aliasing techniques would be better for training example generation. Consequently, generation of positive training pairs without consideration of noun phrase types may induce some "hard" training instances. "Hard" training pair is coreference pair in its coreference chain, but many pairs with the same feature vectors with the pair may not be coreference pairs. “Hard” training instances would lead to some rules which are hazardous for performance. How to deal with such limitation of training data remains an open area of research in the machine learning community. In order to avoid the influence of training data, [Ng and Cardie, 2002] proposed a technique of negative training example selection similar to that proposed in [Soon et al., 2001] and a corpus-based method for implicit selection of positive training examples. Therefore the system got a better performance.

Considering coreference relationship in isolation

In most supervised machine learning systems [Soon et al., 2001; Ng and Cardie, 2002a], when the model determines whether a possible coreference pair is a coreference pair or not, each time it only considers the relationship between two markables. Even if the model’s feature sets include context-dependent information, the context-dependent information is only about one markable, not both two markables. For example, so far, no coreference system cares about that how many pronouns appear between two markables in a document. Therefore only local information of two markables is used and global information in a document is neglected. [Yang et al., 2003] suggested that whether a candidate is coreferential to an anaphor is determined by the competition among all the candidates. Therefore, they proposed a twin-candidate model compared to the single-candidate model. Such approach empirically outperformed those based on a single-candidate model. The paper implied that it is potentially better to incorporate more context-dependent information into coreference resolution. Furthermore, because of incomplete rules set, the model may determine that (A, B) is a coreference pair and (B, C) is a coreference pair. But actually, (A, C) is not a coreference pair. This is a conflict in a coreference chain. So far, most systems do not consider conflicts within one coreference chain. [Ng and Cardie, 2002] noticed the conflicts. They claimed that these were due to classification error. To avoid such conflicts, they incorporated error-driven pruning of classification rule set to avoid. However Ng and Cardie, 2002 did not take the whole coreference chain's information into account either.

Lack of an appropriate reference to theoretical linguistic work on coreference

Basically, coreference resolution is a linguistic problem and machine learning is an approach to learn those linguistic rules in training data. As we have mentioned above, training data has its disadvantages and it may lead to missing some rules which can be simply formulated manually. Moreover, current machine learning approaches usually embed some background knowledge into the feature set, hoping the machine could learn such rules from these features. However, "hard" training examples influence the rules-learning. As a result, such simple rules are missed by the machine.

Furthermore, it is still a difficult task to extract the optimal features set. [Ng and Cardie, 2002a] incorporated a feature set including 53 features, larger than [Soon et al., 2001]'s 12 features set. It is interesting that such large feature set did not improve system performance and even degraded the performance significantly. Instead, [Wagstaff, 2002] incorporated some linguistic rules into coreference resolution directly and the performance increased noticeably. Therefore, there is no 100% accurate machine learning approach. However, simple rules can make up for the weakness. Another successful example is [Iida et al., 2003] who incorporated more linguistic features capturing contextual information and obtained a noticeable improvement over their baseline systems.

1.3.2. Motivation

Motivated by the analysis of current coreference system, in this thesis, we propose a method to improve current supervised machine learning coreference resolution by incorporating a set of ranked linguistic constraints and a conflict resolution method.

Ranked Constraints

Directly incorporating linguistic constraints makes a bridge between theoretical linguistic findings and corpus-based empirical methods. As we have mentioned above, machine learning can lead to missing rules. In order to avoid missing rules and to encode domain knowledge that is heuristic or approximate, we devised a set of constraints, some of which can be violated and some of which cannot. The constraints are seen as ranked constraints and those which cannot be violated are provided with the infinite rank. In this way, the inflexibility of those rule-based systems is avoided. Furthermore, our constraints include two-level of information: one is pair level and the other is markable level. Pair-level constraints include must-link and cannot-link. They are simple rules based two markables. Markable-level constraints consist of cannot-link-to-anything and must-link-to-something. They are based on single markable. And they guide the system to treate anaphors differently. All of them can be simply tested. And the most important is that the constraints avoid overlooking local information by using global information from the whole documents, while current machine learning methods do not pay enough attention to the global information. By incorporating constraints, each anaphor can have more than one antecedent. Hence the system replaces the single-link clustering with multi-link clustering (described in Chapter 4). For example, one of the constraints indicates that proper names with the same surface string in a document should belong to the same equivalence class.

Conflict Resolution:

As we mentioned above, in testing, conflicts may appear in a coreference chain. This should be reliable signal of error. In this thesis, we also proposed an approach to make use of the signals to improve the system performance. When conflict arises, the conflict is measured and a corresponding process is called to deal with the conflicts.

Because of the use of conflict resolution, the ranked constraint's reliability is reduced. Hence the constraints become more heuristic and approximate. As a result, the system's recall is improved significantly (from 59.6 to 63.8) and precision is improved at the same time (from 61.7 to 64.1).

We observed that by incorporating some simple linguistic knowledge, constraints and conflict resolution can reduce the influence of training data limitation to a certain extent. By devising multi-level constraints and using the coreference chain's information, coreference relationship becomes more global, not isolated. In the following chapter, we show how the new approach achieves the F-measure of 64.2 outperforming earlier machine learning approaches, such as [Soon et al., 2001]’s 60.4 and [Ng and Cardie, 2002a]’s 63.4.

In this thesis, we duplicated Soon work as the baseline for our work. Before we incorporated constraints and conflict resolution, we added two more steps, head noun phrase extraction and proper name identification, into Natural Language Processing (NLP) pipeline. By doing so, the baseline system’s performance increases from 59.3 to 60.9 and consequently achieves an acceptable performance. In Chapter 2, the two additions are described in detail.

1.4. Structure of the thesis

The rest of the thesis is organized as follows:

Chapter 2 and Chapter 3 will introduce the baseline system’s implementation. Chapter 2 will introduce the natural language processing pipeline used in our system and describe the two additional steps, noun phrase extraction and proper name identification, and the corresponding experimental result. Chapter 3 will introduce the baseline system based on [Soon et al., 2001] in brief.

Chapter 4 and Chapter 5 will introduce our approach in detail. Ranked constraints will be introduced in Chapter 4. In this Chapter, we will give the types and definitions of constraints we incorporate in our system. Chapter 5 will describe the conflict resolution algorithm in detail.

In Chapter 6, we will evaluate our system, by comparing it with some existing systems, such as [Soon et al., 2001]. And we also show the contributions of constraints and conflict resolution respectively. At the end of this chapter, we will analyze the remaining errors in our system.

Chapter 7 will conclude the thesis, highlight its contributions to coreference resolution and describe the future work.

2. Natural Language Processing Pipeline

2.1. Markables Definition

Candidate which can be part of coreference chains are called markable in MUC-7 [MUC-7, 1997]. According to the definition of MUC-7 [MUC-7, 1997] Coreference Task, markables include three categories whether it is the object of an assertion, a negation, or a question: noun, noun phrase and pronoun. Dates, currency expression and percentage are also considered as markables. However interrogative "wh-" noun phrases are not markables.

Markable extraction is a critical component of coreference resolution, although it does not take part in coreference relationship determination directly. In the training part, two referring expressions cannot form a training positive pair if either of them is not recognized as markable by the markable extraction component even if they belong to the same coreference chain. In the testing part, only markables can be considered as a possible anaphor or a possible antecedent. Those expressions which are not markables will be skipped. In this case markable extraction component performance is an important factor in coreference system’s recall. It also means markable extraction component performance determines the maximum value of recall.

2.2. Markables Determination

In this thesis, a pipeline of natural language processing (NLP) is used as shown in Figure 2.1. It has two primary functions. One is to extract markables from free text as actually as possible and at the same time determine the boundary of those markables. The other is to extract linguistic information which will be used in later coreference relationship determination. Our pipeline of natural language processing (NLP) imitates the architecture of the one used in [Soon et al., 2001]. Both pipelines consist of tokenization, sentence segmentation, morphological processing, part-of-speech tagging, noun phrase identification, named entity recognition, nested noun phrase extraction

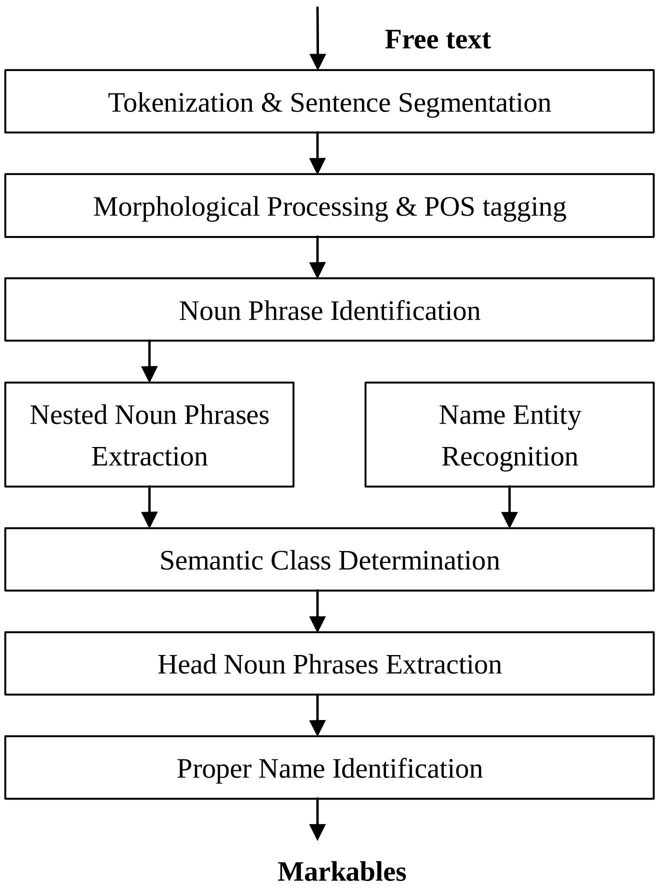

Free text
Tokenization & Sentence Segmentation
Morphological Processing & POS tagging
Noun Phrase Identification
Nested Noun Phrases Extraction
Name Entity Recognition
Semantic Class Determination
Head Noun Phrases Extraction
Proper Name Identification
Markables

Figure 2.1
The architecture of natural language processing pipeline.

and semantic class determination. Besides these modules, our NLP pipeline adds head noun phrase extraction and proper name identification to enhance the performance of NLP pipeline and to compensate the use of a weak named entity recognition that we used. This will be discussed in detail later.

2.2.1. Toolkits used in NLP Pipeline

In our NLP pipeline, three toolkits are used to complete the task of tokenization, sentence segmentation, morphological processing, part-of-speech tagging, noun phrase identification and named entity recognition.

LT TTT [Grover et al., 2000], a text tokenization system and toolset which enables users to produce a swift and individually-tailored tokenization of text, is used to do tokenization and sentence segmentation. It uses a set of hand-craft rules to token input SIML files and uses a statistical sentence boundary disambiguator which determines whether a full-stop is part of an abbreviation or a marker of a sentence boundary.

LT CHUNK [LT CHUNK, 1997], a surface parser which identifies noun groups and verb groups, is used to do morphological processing, part-of-speech tagging and noun phrase identification. It as well as LT TTT [Grover et al., 2000] is offered by the Language Technology Group [LTG]. LT CHUNK [LT CHUNK, 1997] is a partial parser, which uses the part-of-speech information provided by a nested tagger and employs mildly context-sensitive grammars to detect boundaries of syntactic groups. It can identify simple noun phrases. Nested noun phrases, conjunctive noun phrases as well as noun phrases with post-modifiers cannot be recognized correctly. Consider the following example:

Sentence 2.1 (1): ((The secretary of $(\text{Energy})_{a1})_{a2}$ and (local $\text{farmers})_{a3})_{a4}$ have expressed $(\text{concern})_{a5}$ that (a $(\text{plane})_{a6}$ crash) $_{a7}$ into (a ((plutonium) $_{a8}$ storage) $_{a9}$ $\text{bunker})_{a10}$ at (Pantex) $_{a11}$ could spread (radioactive smoke) $_{a12}$ for $(\text{miles})_{a13}$.

Sentence 2.1 (2): (The $\text{secretary})_{b1}$ of $(\text{Energy})_{b2}$ and (local $\text{farmers})_{b3}$ have expressed $(\text{concern})_{b4}$ that (a plane $\text{crash})_{b5}$ into (a plutonium storage $\text{bunker})_{b6}$ at $(\text{Pantex})_{b7}$ could spread (radioactive $\text{smoke})_{b8}$ for $(\text{miles})_{b9}$.

The sentence is extracted from MUC-7 [MUC-7, 1997] dryrun documents and it is shown twice with different noun phrase boundaries. The first sentence is hand-crafted and the second is the output of LT CHUNK. Among 13 markables, LT CHUNK tagged 8 of them (a1, a3, a5, a7, a10, a11, a12, a13) correctly, missed 4 of them (a4, a6, a8, a9) and tagged one (a2,) by error. Among 4 missed markables, “a4” is a conjunctive noun phrase and a6, a8 as well as a9 are nested noun phrases. Among the errors, a2 is a noun phrase with post-modifier, “Energy”, and is tagged as b1. Fortunately, It is possible to extend it to a2 automatically, because besides the article, “The”, b1’s string matches with the string of a2’s head noun phrase, “secretary”. In the following sections, modules which can deal with such problems will be introduced.

As for named entity recognition, in our system dryrun documents, we use the MUC-7 NE keys. For formal documents, we use named entity recognizer offered by Annie [Annie], Annie [Annie] is an open-source, robust Information Extraction (IE) system which relies on finite state algorithms. Unfortunately, Annie’s performance is much lower than the MUC standards. Tested on coreference task’s 30 dryrun document, its F-measure is only 67.5, which is intolerable for the coreference task. To make up for the weakness to a certain extent, we incorporated a module, proper name identification, into NLP pipeline. This module will be introduced in detail later.

2.2.2. Nested Noun phrase Extraction

Nested noun phrase extraction accepts the LT CHUNK’s output and extracts prenominals from the simple noun phrases tagged by LT CHUNK. According to [Soon et al., 2001], there are two kinds of nested noun phrases that need to be extracted:

Nested noun phrases from possessive noun phrases: Possessive pronouns (e.g. “his” in “his book”) and the part before “’s” of a simple noun phrase (e.g. “Peter” in “Peter’s book”).

Prenominals: For instance, in “a plutonium storage bunker”, “plutonium” and “storage” are extracted as nested noun phrases.

After this model, a7 and a8 in above example which were missed by LT CHUNK can be recognized correctly. But according to the task definition of MUC-7 [MUC-7, 1997] coreference resolution, nested noun phrases can be included into coreference chain only if it is coreferential with a named entity or to the syntactic head of a maximal noun phrase. Therefore after getting coreference chains, those chains which consist of only nested noun phrases, but no named entity and syntactic head of a maximal noun phrase, will be deleted.

2.2.3. Semantic Class Determination

This is an important component for later feature vectors computation. Most linguistic information is extracted from here. We use the same semantic classes and ISA hierarchy as [Soon et al., 2001]'s and we also incorporate WordNet 1.7.1's synset [Miller, 1990] to get the semantic class for common nouns. The main difference is in the gender information extraction. Besides WordNet's output, pronouns and designators (e.g. "Mr.", "Mrs."), we incorporate a woman name list and a man name list (See Appendix A). If a person's name is identified by named entity recognition, we will search in name lists to see whether the name is a woman's name, a man's or neither.

2.2.4. Head Noun Phrases Extraction

Head noun phrase is the main noun without left and right modifiers in a noun phrase. The maximal noun phrase includes all text which may be considered a modifier of the noun phrase, such as post-modifiers, appositional phrases, non-restrictive relative clauses, prepositional phrases which may be viewed as modifiers of the noun phrase or of a containing clause. MUC-7 [MUC-7, 1997] required that the string of a markable generated by NLP pipeline must include the head of the markable and may include any additional text up to a maximal noun phrase. Because pre-processing cannot determine accurate boundaries of noun phrases, if the boundary of a markable is beyond its maximal noun phrase, the markable cannot be recognized as an accurate antecedent or anaphor by MUC Scorer program. But after noun phrase extraction (Shown in Figure

```
**Algorithm** Head-Noun-Phrase-Extraction ( $MARKABLE$ : set of all markables)
**for** $i(i\_SEMCLASS) \in MARKABLE$ **do**
    $HeadNP :=$ the most right noun of $i$
    **if** $HeadNP$ is different from $i$ **then**
        $HeadNP\_SEMCLASS := i\_SEMCLASS$
        $MARKABLE := MARKABLE \cup \{HeadNP(HeadNP\_SEMCLASS)\}$
**return** $MARKABLE$
```

Figure 2.2:
The Noun Phrase Extraction Algorithm

2.2), the new markable which is its head noun phrase can be recognized by MUC Scorer. Accordingly, head noun phrase extraction can form a screen for inaccurate boundary determination and improve system's recall. For example:

Sentence 2.2: The risk of that scenario, previously estimated at one chance in 10 million, is expected to increase when current flight data are analyzed (later (this (year)$_1$)$_2$)$_3$, according to a safety board memo dated May 2.

The example is extracted from MUC-7 [MUC-7, 1997] dryrun document. In this example, boundary 3 is determined by NLP pipeline without head noun phrase extraction. Boundary 2 is determined by hand which can be recognized as an accurate referring expression by MUC Scorer and boundary 1 can also be accepted by Scorer. It is obvious that boundary 3 cannot meet Scorer's requirement and it leads to missing a referring expression. But after head noun phrase extraction, "this year" (head noun phrase is "year") is recovered.

Another valuable contribution of noun phrase extraction is that it can improve system's performance noticeably by head noun string matching. Actually, in [Soon et al., 2001], String match is only for the whole markable's string excluding articles and demonstrative pronouns. Consider the following sentence extracted from MUC-7 [MUC-7, 1997] dryrun document:

Sentence 2.3: Mike McNulty, the FAA air traffic manager at Amarillo International, said (the previous (aircraft) [count])$_1$, conducted in late 1994, was a ``(manual [count])$_2$ on a pad," done informally by air traffic controllers.

The two "count"s between square brackets are coreferential. And markable 1 and markable 2 are determined by NLP pipeline without noun phrase extraction. Even though two markables' boundaries can meet the requirement of MUC Scorer, coreference resolution cannot recognize their coreference relationship. It is partially because their string match value is negative (See Figure 3.1). But after noun phrase extraction, two "count"s are extracted as isolate markables respectively. According to the string match, their coreference relationship can be recognized correctly. This is why head noun phrase extraction can recover some coreference relations. Later, we will show that head noun phrase extraction can improve the system's performance significantly –recall improved from 56.1 to 62.7 (Table 2.1).

After adding head noun phrase extraction, there may be two markables with the same head noun appearing in a coreference chain or even two different coreference chains. In our system if two markables with the same head noun appear in coreference chains, the shorter markable will take the place of the longer. This is called head noun preference rule. If they are in different chains, the conflict resolution will be used.

Later we will describe it in detail in Chapter 5.

2.2.5. Proper Name Identification

We introduce the proper name identification into NLP pipeline because of two reasons:

One has been mentioned in 2.2.1: Annie’s poor performance. Its score on the MUC-7 [MUC-7, 1997] named entity task for coreference task’s 30 dryrun documents is only 67.5 in F-measure (Recall is 73.1, precision is 79.6). It is far from the MUC-7 standard. Through reading its output, we find that we can adjust it to meet our requirement in such a way:

Annie always remembers the named entity’s string exactly as it first appears in the document. Accordingly, Annie misses other different expressions of the named entity in the later document. For example, “Bernard Schwartz” is the first appearance of the person in the document and it is recognized as “PERSON” correctly, but the following “Schwartz”s are all missed by Annie. For another example, “Loral” is recognized as “ORGANIZATION” correctly, but the following named entities including “Loral” are missed, for example “Loral Space” is recognized as two named entities: “Loral” and “Space”. To obtain more named entities, we add a post-processing for Annie: for each named entity recognized by Annie, search for its aliases in the document and endow them the same named entity class with the one recognized by Annie.

The other reason incorporating proper name identification is due to nested noun phrase and head noun phrase extraction. As we know, proper name cannot be separated into sub noun phrases. But nested noun phrase and head noun phrase extraction still apply to those proper names which are not recognized as named entities. Consider the example: “Warsaw Convention”. Our named entity recognition does not recognize it as a named entity. Therefore “Warsaw” and “Convention” are extracted as markables by nested noun phrase extraction and head noun phrase extraction, respectively.

```
**Algorithm** Proper-Name-Identification ( $MARKABLE$ : set of all markables)
**for** $i_1(i_1\_SEM),..,i_n(i_n\_SEM) \in MARKABLE$ **&&** they are consecutive proper names connected by “&”,”/” or nothing **do**
    $\Pr operName := \{ i_1(i_1\_SEM),..,i_n(i_n\_SEM) \}$;
    **for** $j(j\_SEM) \in \Pr operName$ **do**
        $j(j\_SEM) := j(j\_SEM)$ ’s root markable with the same head noun;
    $K :=$ the text covered by $\Pr operName$ ’s member and their interval string;
    $K\_SEM := i_n\_SEM$;
    $MARKABLE := MARKABLE \cup K(K\_SEM)$;
    **for** $j(j\_SEM) \in \Pr operName$ **do**
        **if** $j(j\_SEM)$ is not named entity **then**
            $MARKABLE := MARKABLE$ /{ $j(j\_SEM)$ ,its including markables};
**return** $MARKABLE$ ;
```

Figure 2.3
The Proper Name Identification Algorithm

Consequently, all “Warsaw Convention” in the document are extracted. Because of the string match and head noun phrase preference rule (mentioned in last section), all the “Convention”s form a coreference chain but all the “Warsaw Convention”s are missed. It causes system’s performance drop noticeably. Proper name identification is required to resolve such problems. Figure 2.3 shows the module’s algorithm. It recognizes the consecutive tokens tagged with “NNP” or “NNPS” as a markable without nested noun phrases and head noun phrases (“NNP” and “NNPS” are added by POS tagging. The token tagged with one of them should be a part of a proper name.). If there is a token, “&”or“/”, between two proper names, then combine the token and the two proper names to a proper name. In next section we will show through experimental result that proper name identification not only can make up the weakness of named entity recognition but also can improve the system’s performance.

2.2.6. NLP Pipeline Evaluation

In order to evaluate head noun phrase extraction and proper name identification, we tested four different NLP pipelines: NLP without noun phrase extraction and proper name identification, NLP with only noun phrase extraction, NLP with only proper name identification and NLP with both modules. All four NLP pipelines use LT TTT [Grover et al., 2000] to do tokenization and sentence segmentation procession, use LT CHUNK [LT CHUNK, 1997] to do morphological processing and POS tagging, and use Annie to do named entity recognition. They share the common nested noun phrase extraction and semantic class determination module. We take the four NLP pipeline’s

System Variation	dryrun (30)			formal(20)		
	R	P	F	R	P	F
Soon et al.	/	/	/	56.1	65.5	60.4
Ng and Cardie 2002a	/	/	/	57.4	70.8	63.4
Duplicated Soon Baseline						
None	49.2	74.0	59.1	51.0	70.8	59.3
Proper Name only	49.3	74.3	59.2	51.0	71.7	59.6
Head Noun Phrase only	57.1	64.7	60.3	58.9	60.1	59.5
Head NP and Proper Name	57.4	64.7	**60.9**	59.6	62.3	**60.9**
Our Complete System						
None	52.0	73.1	60.8	56.1	70.2	62.4
Proper Name only	52.1	73.4	60.9	56.2	71.2	62.8
Head Noun Phrase only	59.5	66.5	62.8	62.7	62.2	62.5
Head NP and Proper Name	59.8	67.2	**63.3**	63.7	64.7	**64.2**
One Chain						
Soon et al.	/	/	/	87.5	30.5	45.2
None	87.5	30.1	44.8	88.7	30.1	44.9
Proper Name only	87.5	30.4	45.1	88.6	30.6	45.5
Head Noun Phrase only	89.2	22.4	35.8	90.7	22.4	36.0
Head NP and Proper Name	**89.2**	22.7	36.2	**90.6**	23.0	36.6

Table 2.1:
MUC-7 results of complete and baseline systems to study the contribution of head noun phrase extraction and proper name identification. Recall, Precision and F-measure are provided. "One chain" means all markables form one coreference chain.

outputs as coreference resolution system's input. There are three coreference resolution systems used in the experiment: duplicated Soon baseline system, our complete system with ranked constrains and conflict resolution, and the one chain system (all markables form a coreference chain). There are two sets of data used: MUC-7 [MUC-7, 1997] 30 dryrun documents and MUC-7 [MUC-7, 1997] 20 formal documents. Unfortunately, we have no hand annotated corpora to test NLP pipeline. Therefore we cannot evaluate NLP pipeline's performance directly. But the coreference scorer results can imply their performances. The result is shown in Table 2.1.

Table 2.1 shows that both head noun phrase extraction and proper name identification can enhance the performance of NLP pipeline as well as coreference system's performance. Head noun phrase extraction can make recall increase about 7.9 percent and proper name identification mostly improves the precision. If two modules are both used, then the result achieved is the best.

Head noun phrase extraction's contribution is reflected well from one chain system's results. One chain system can tell us the maximum recall that coreference system can achieve based one NLP pipeline. And the higher recall means more markables can be extracted correctly by NLP pipeline. It reflects the capability of a NLP pipeline. From Table 2.1, we see that head noun phrase extraction improves recall about 2 % on both data sets. And the recall on formal data exceeds [Soon et al., 2001]'s by 3.2%.   For the other two systems, the recall increase is much higher, approximately 7 percent. Although the precision drops, the F-measures did not drop and sometimes even increases.

As for proper name identification, we see that although recall does not change too much, all the precisions increase, and F-measures also increase a little bit.

After adding the two modules, duplicated Soon baseline's result (60.9) can beyond [Soon et al., 2001]'s (60.4). It shows that two modules not only can make up for the weakness of NLP pipeline (mostly because named entity recognition), but can also improve the performance. This is also true for our complete system. The best result (64.2) is achieved after adding the two modules, which is higher than most coreference systems, such as [Soon et al., 2001; Ng and Cardie, 2002a].

The experiment shows that NLP pipeline is a critical for a coreference system. After adding the two modules, our duplicated Soon baseline system achieves an acceptable result (60.9). In this thesis, we take it as our departure point. In the later chapters, we will describe how to improve the performance of the baseline system through ranked constraints and conflict resolution.

3. The Baseline Coreference System

Our system takes [Soon et al., 2001] as the baseline model. [Soon et al., 2001] is the first system machine learning system with comparable result to that of state-of-the art non-learning systems on data sets of MUC-6 [MUC-6, 1995] and MUC-7 [MUC-7, 1997]. The system used a feature set including 12 features, decision tree trained by C5.0 and a right-to-left search for the first antecedent to determine coreference relationship. After adding head noun phrase extraction module and proper name identification module into our NLP pipeline, the duplicated Soon baseline system has achieved an acceptable result, 60.9, comparing to Soon et al.'s 60.4. In this chapter, we will describe the baseline system's feature set, training approach and testing approach in brief. More details can be found in [Soon et al., 2001].

3.1. Feature Vector

[Soon et al., 2001] proposed a feature set including 12 features, which contains propositional, lexical, grammatical and semantic information. The feature set is simple and effective, and it can lead to comparable result to that of non-learning systems. After [Soon et al., 2001], [Ng and Cardie, 2002a] extended [Soon et al., 2001]'s feature set to include 53 features. However, 53 features made the performance drop significantly. It proves that more features do not mean higher performance. Consequently in this thesis, we do not do any change to [Soon et al., 2001]'s feature

Feature Type		Feature	Description
Positional		**DIST**	The number of sentences between i and j. O is i and j are in the same sentence
Lexical		**STR_MATCH**	1 if i matches the string of j, else 0.Articles and demonstrative pronouns are removed in advance
		ALIAS	1 if i is an alias of j or vice versa, else 0.i and j should be named entities with the same semantic class
Grammatical	NP type	**I_PRONOUN**	1 if i is a pronoun, else 0
		J_PRONOUN	1 if j ,is a pronoun, else 0
		DEF_NP	1 if j is a definite noun phrase, else 0
		DEM_NP	1 if j is a demonstrative noun phrase, else 0
		PROPER_NAME	1 if both i and j are proper names, else 0. Prepositions such as "of" or "and" are not considered
	Linguistic constraints	**NUMBER**	1 if i and j agree in number, else 0
		GENDER	2 if either i or j's gender is unknown, else 1 if i and j agree in gender, else 0
		APPOSITIVE	1 if j is in apposition to i, else 0
Semantic		**SEMCLASS**	1 if i and j are in agreement if one is the parent of the other or they are the same, else 0 if neither semantic class is unknown, else compare their head noun strings, 1 if matched, 2 else.

Table 3.1:
Feature set for the duplicated Soon baseline system. i and j are two extracted markables. And i is the possible antecedent and j is the possible anaphor.

set but put our emphasis on ranked constraints and conflict resolution.

Table 3.1 describes our system's feature set based [Soon et al., 2001]'s. The features can be linguistically divided into four groups: positional, lexical, grammatical and semantic. The positional feature considers the position relation between two markables. The lexical features test the relation based on markables' corresponding surface strings. The grammatical features can be divided into 2 sub groups. One determines the NP type, such as definite, indefinite, demonstrative NP, proper name. The other determines some linguistic constraints, such as number agreement, gender agreement. The semantic feature gives markable corresponding semantic class: person, male, female, organization, location, money, percent, date and time. The definition of each feature is listed in Table 3.1. More details can be found in [Soon et al., 2001].

3.2. Classifier

3.2.1. Training Part

In training part, most machine learning coreference systems used C4.5 [Quinlan, 1993], C5.0, an updated version of C 4.5 [Quinlan, 1993], or RIPPER [Cohen, 1995], an information-gain-based rule learning system. [Soon et al., 2001] used C5.0 to train its decision tree. In our system, C4.5 [Quinlan, 1993] is used to build the classifier and default setting for all C4.5 parameters is used, except the pruning confidence level. The pruning confidence level is equal to that of [Soon et al., 2001], 60.

The main difference among machine learning coreference systems is the training example generation, especially positive training pair generation.

Positive training pair generation can be divided into three approaches roughly. The simplest approach is to create all possible pairing in a coreference chain. We call the approach RESOLVE (because it is the way RESOLVE [McCarthy, 1996] used). This approach may lead to too many "hard" training examples as we have mentioned above. Another approach, better than RESOLVE, is [Soon et al., 2001]'s approach. [Soon et al., 2001] only extracted the pairs consisting of two referring expressions immediately adjacent in a coreference chain. Even though there will be less positive pairs, more accurate classifier can be obtained. The third approach is more sophisticated than former two. It introduces some rules into the selection of positive training pairs. For example, in [Ng and Cardie, 2002a], they used different generating ways for non-pronominal anaphor and pronoun anaphor. [Ng and Cardie, 2002] even used a more complex approach to generate positive training pair. It incorporates a rule learner into the positive training pair generation. By doing so, they discarded those pairs that do not satisfy rules learned from the training data.

Ng and Cardie showed that the third approach can obtain the most accurate classifier. For simplicity, our system uses [Soon et al., 2001]'s approach to generate positive training pair. As to negative training pair generation, for each positive training pair, we extract the markables between the pair, excluding those markables which has the common part with the two referring expression of the positive training pair. Each of the extracted markables is paired with the positive training pair's anaphor and to form a negative training pair. Using our NLP pipeline with head noun phrase extraction module and proper name identification module, we can extract 1532 positive training pairs which occupy 3.5% among total training pairs we get.

Figure 3.1 shows the decision tree our system used. The tree learned from MUC-7

data sets uses 12 features. In general, we see that STR_MATCH and GENDER are two most important features for coreference relationship determination.

3.2.2. Testing Part

In testing part, [Soon et al., 2001] proposed a right-to-left search which is a good fit to the procession of how humans process documents.

Documents are written with the assumption that a human will be reading them. Like humans, [Soon et al., 2001]'s system processes a document from the beginning to end. Whenever the system encounters a markable in the document, except the first markable, the system searches the markable's antecedent from right to left till it finds one recognized by decision tree. If there is no antecedent found, the markable is considered non-anaphoric and the system moves on to the next markable.

It should be noticed that the test processing should match with the generation of training pairs. In [Soon et al., 2001], positive pair is the adjustment referring expressions in a coreference chain, Therefore in testing processing, [Soon et al., 2001] uses the first antecedent recognized by decision tree as the anaphor's antecedent. But in [Ng and Cardie, 2002a], positive pair is generated differently for non-pronominal anaphor and pronoun anaphor, Therefore in testing, [Ng and Cardie, 2002a] uses the best antecedent recognized by decision tree as the anaphor's antecedent ("best" means the highest probability above 0.5).

In our system, we use the right-to-left search. But in order to add constraints and conflict resolution, we make some modifications in testing processing, which will be described in detail in the following chapters.

```
STR_MATCH = 0:
|   |   GENDER = 0: - (31.0/0.5)
|   |   GENDER = 1:
|   |   |   |   J_PRONOUN = 1: + (60.0/6.9)
|   |   |   |   J_PRONOUN = 0:
|   |   |   |   |   |   I_PRONOUN = 0: - (12.0/2.7)
|   |   |   |   |   |   I_PRONOUN = 1:
|   |   |   |   |   |   |   |   DIST <= 2 : + (24.0/8.9)
|   |   |   |   |   |   |   |   DIST > 2   : - (5.0/1.7)
|   |   GENDER = 2:
|   |   |   |   ALIAS = 1: + (41.0/8.9)
|   |   |   |   ALIAS = 0:
|   |   |   |   |   |   J_PRONOUN = 0:
|   |   |   |   |   |   |   |   APPOSITIVE = 0: - (27124.0/460.0)
|   |   |   |   |   |   |   |   APPOSITIVE = 1:
|   |   |   |   |   |   |   |   |   |   PROPER_NAME = 1: - (5.0/0.5)
|   |   |   |   |   |   |   |   |   |   PROPER_NAME = 0:
|   |   |   |   |   |   |   |   |   |   |   |   SEMCLASS = 0: + (1.0/0.4)
|   |   |   |   |   |   |   |   |   |   |   |   SEMCLASS = 1: + (13.0/3.8)
|   |   |   |   |   |   |   |   |   |   |   |   SEMCLASS = 2: - (2.0/0.5)
|   |   |   |   |   |   J_PRONOUN = 1:
|   |   |   |   |   |   |   |   SEMCLASS = 0: - (249.0/12.1)
|   |   |   |   |   |   |   |   SEMCLASS = 2: - (1261.0/136.3)
|   |   |   |   |   |   |   |   SEMCLASS = 1:
|   |   |   |   |   |   |   |   |   |   NUMBER = 0: - (161.0/31.3)
|   |   |   |   |   |   |   |   |   |   NUMBER = 1:
|   |   |   |   |   |   |   |   |   |   |   |   I_PRONOUN = 1: + (9.0/1.7)
|   |   |   |   |   |   |   |   |   |   |   |   I_PRONOUN = 0:
|   |   |   |   |   |   |   |   |   |   |   |   |   |   DIST <= 0 : + (52.0/17.1)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   DIST > 0 : - (43.0/21.0)
STR_MATCH = 1:
|   |   SEMCLASS = 0: + (3.0/1.6)
|   |   SEMCLASS = 2: - (29.0/1.7)
|   |   SEMCLASS = 1:
|   |   |   |   DEM_NP = 1: - (5.0/1.7)
|   |   |   |   DEM_NP = 0:
|   |   |   |   |   |   DEF_NP = 0: + (466.0/56.7)
|   |   |   |   |   |   DEF_NP = 1:
|   |   |   |   |   |   |   |   NUMBER = 0: - (8.0/1.7)
|   |   |   |   |   |   |   |   NUMBER = 1: + (146.0/36.4)
```

Figure 3.1
The decision tree classifier learned from MUC-7 dryrun 30 documents

4. Ranked Constraints

The high-level goal of this thesis is to improve the machine learning coreference system effectively by incorporating linguistic background knowledge in the form of constraints. Some earlier systems have made such attempts. In [Ng and Cardie, 2002b], they used an anaphoricity classifier to filter those non-anaphoric markables before using coreference engine. In order to avoid the anaphoricity classifier's misclassifications, they incorporated STR_MATCH constraint and ALIAS constraint on anaphoricity classifier. By doing so, they improved the result from 58.4 to 64.0 in F-measure. Another successful system incorporating constraints is [Wagstaff, 2002]. Before it, [Wagstaff and Cardie, 2000] had proved that incorporation of instance-level constraints into clustering algorithm can offer substantial benefits. Based on the former work [Cardie and Wagstaff, 1999] of viewing coreference resolution as a clustering task, [Wagstaff, 2002] incorporated instance-level hard constraints into coreference task and made a significant improvement. Both systems indicate that incorporation of linguistic constraints into coreference resolution can be a promising direction to improve the accuracy of the task.

In this chapter, we will give the details of our ranked constraints. The four characteristics of the constraints set, linguistic-based, multi-level, ranked and compatible with supervised machine learning approach, will be introduced in Section 4.1. Then we will present the definition of each constraint (Section 4.2). Finally, we will discuss how to make the constraints cooperate with coreference system (Section 4.3). And the evaluation results will be shown in Chapter 6.

4.1. Ranked Constraints in coreference resolution

In this thesis, we incorporate a set of constraints into a supervised machine learning coreference resolution [Soon et al., 2001]. The constraints have the following characteristics: linguistic-based, multi-level, ranked and compatible with supervised machine learning approach.

4.1.1. Linguistic Knowledge and Machine Learning Rules

Misclassification is inevitable in machine learning coreference resolution. There are three reasons

Insufficient training data

30 dryrun documents of MUC-7 [MUC-7, 1997] are used to train the coreference classifier in our system. Among the training data, there are only 1532 positive pairs which occupy about 3.4% in total training pairs. Obviously 1532 positive pairs are not sufficient enough to capture all rules, especially rare coreference rules, such as appositive rule. For example:

Sentence 4.1: That's certainly how (Eileen Cook)$_{a1}$ and ((her)$_{a2}$ 22-month-old daughter)$_{b1}$, (Jessie)$_{b2}$, see it.

In this sentence, we see that a1 is not a pronoun but a2 is. Since their value of

STR_MATCH and GENDER are 0, 1, respectively, the decision tree (shown in Figure 3.1) recognizes (a1-a2) as coreference pair. And next system thinks b1 and b2 are not coreferential because their STR_MATCH and GENDER are 0, 1, respectively and neither of them is pronoun. Instead, the system assigns a2 as b2's antecedent. The determination is made by error because the decision tree ignores the fact that b1 and b2 are appositive. The main reason may be that there are not sufficient positive training pairs to represent such appositive rule when two referring expressions in appositive relation agree in gender. But the rule is applied in test document. Therefore decision tree cannot recognize b1 and b2 correctly.

Up to now, a decision tree with 100% accuracy is still unavailable. The highest precision achieved is approximately 70%. In the case of lack of sufficient training data, incorporating some easily-formulated constraints based on linguistic knowledge may be a promising idea to overcome misclassification. For instance, by adding the appositive must-link and nested NP cannot-link (they will be described in the next section), b1 and b2 are correctly recognized and a2 and b2's error link is also removed successfully.

"Hard" training example

In general, different noun phrase types have different coreference rules. For pronoun, its antecedent should be the nearest antecedent in its preceding document. For proper name, its antecedent should be the nearest antecedent meeting the requirement of STR_MATCH or ALIAS. Somewhat disappointingly, more sophisticated situation exists generally in coreference. For example:

Sentence 4.2:``It means that (Bernard Schwartz)$_{a1}$ can focus most of ((his)$_{a2}$ time) on ((his)$_{a3}$ foster son)$_{b1}$, (Peter)$_{b2}$. (Bernard Schwartz)$_{a4}$ is fatherly ," (he)$_{c1}$ said.

There are three referents: Bernard Schwartz, Peter and the speaker, "he". In the sentence, a1, a2, a3, a4 refer to "Bernard Schwartz", b1 and b2 refer to "Peter" and c1 refers to the speaker. With regard to the decision tree shown in Figure 3.1, a1, a2, a3, a4, b2, and c1 form a coreference chain. In the coreference chain, (b1-b2) is missed and (b2-a3) as well as (a4-c1) are spurious. If we filter "hard" training examples according to the principle of proper name, it is possible to produce a classifier with higher accuracy for proper name. As a result, such spurious link as (b2-a3) would not appear in new coreference chains. But (a4-c1) is an exception. Although a4 is c1's nearest antecedent and their semantic class, gender class are same, they are never coreferential. This case is too sophisticated for a machine learning approach to resolve without more linguistic knowledge. However it is easy, even obvious for a human. Because we know that a speaker is used to using the first person pronoun to refer to himself in his speech. Even in comparison to the most complex approach of training example generation (Such as [Ng and Cardie, 2002], they incorporated a rule learner to avoid "hard" training example as possible as they can), the rules offered by human are provided with more reliability than those learned by machine. Moreover, it is simpler and more effective to use constraints to resolve such problem in the testing part.

Unreliable feature value and lack of linguistic information

In our system, the features are extracted automatically without any hand-craft information. Inevitably, features include some error linguistic information. The error features influence both training and testing. Suppose Sentence 4.2 appears in training documents. The classifier would learn that two markables are coreferential if they have appositive relation and agree in gender. Based on such classifier, link (b1-b2) in Sentence 4.1 would be recognized correctly. But if “Peter”’s gender is “unknown” in Sentence 4.2 (it is possible if “Peter” is not included in man name list), the classifier will miss the coreference rule again.

Among 12 features, GENDER, SEMCLASS and NUMBER have the highest error rate (POS tagging and named entity recognizer should be responsible for it). Unfortunately, all of them still play important roles in coreference determination. Furthermore, these errors are almost stochastic. It is difficult for machine to capture their common characteristics between train data and test data. If a constraint only employs reliable features, it can be used to check the answers offered by decision tree. Incorporating such constraints not only can avoid overlooking some features but also can filter some errors made by unreliable features. Consider Sentence 4.1, appositive must-link gives feature APPOSITIVE preference on other features while avoiding error in gender. For example:

Sentence 4.3: (Louis Gallois)$_1$, (chief executive)$_2$ of Aerospatiale, is unequivocal about how Europe compares to the U.S. in consolidating the aerospace and defense industries.

Markable 1 and markable 2 are coreferential because of appositive relation. But our named entity recognizer think “Louis Gallois” should be an organization, and semantic class determination module thinks “chief executive” is a person. As a result, their link is missed by decision tree because of the error semantic class of “Louis Gallois”. In our system, we give the appositive must-link a higher score to avoid such errors.

Besides unreliable feature values, lack of linguistic information is a factor of misclassification. In Sentence 4.2, the 12 features set cannot distinguish the difference between (a4-c1) and (a1-a2) using the feature vector. This is because information about speaker and his speech is not included in features set. The reason why we make use of constraints instead of adding more features into feature set is that more features would bring more feature errors into the system. And the relation among features would be more complex. Consequently, such feature set would confuse the machine learning processing.

In conclusion, the misclassification of coreference classifier is due to insufficient training data, “hard” training example, unreliable feature value and lack of linguistic information. It can be resolved by applying linguistic background knowledge in the form of constraints to a certain extent. Moreover constraints apply linguistic knowledge in a more effectively and simpler way. It results in a more robust and error-tolerant coreference system.

4.1.2. Pair-level Constraints and Markable-level Constraints

In [Wagstaff, 2002], they proposed a set of 10 pair-level hard constraints, including 9 cannot-links and one must-link. In this thesis, we expand constraints set to include markable-level constraints. Markable-level constraint is a kind of constraint applied to one markable in isolate, but not to a pair of markables. The constraint captures the common characteristics of some markables, such as anaphoricity, such as cannot-link-to-anything. By using it, we keep away from redundantly presenting cannot-link constraints on each pair formed by a markable which never takes part in coreference relationship. Another advantage is that some constraints cannot be represented by pair-level constraints. Must-link-to-something is such a markable-level constraint used in our system. It is difficult to be transferred to must-link or cannot-link. For example, "he" is the third person pronoun. It is supposed to have an antecedent. But it is hard to say "he" must link to a specific markable.

4.1.3. Un-ranked Constraints vs. Ranked Constraints

Theory-oriented rule's inflexibility has been noted for a long time. It is because that language is infamous for its exceptions to rules. If a rule is violated by an actual text, then the rule will force the system to make an incorrect decision. However, machine-learning approach is better than theory-oriented rule due to its flexibility. How to incorporate constraints to a coreference system built through machine learning without any harm to its flexibility? In this thesis, we devise a set of constraints which is general enough to be used in a large range of knowledge domains. And we give each constraint a score to avoid forcing system to make incorrect decision when it is violated. Furthermore, when a constraint is violated, the conflict resolution technique (described in Chapter 5) can help coreference system to make a correct decision according to corresponding scores.

By doing so, there is no need to ensure the 100% accuracy of each constraint. Constraints can be more heuristic and approximate. Even in a set of constraints, one constraint can violate other constraints in some special case. For example:

Sentence 4.4: “(McDonald's Chief Financial Officer)$_1$, (Jack Greenberg)$_2$”.

Markable 1 and markable 2 are both proper names. Besides appositive must-link, this pair meets the requirement of a cannot-link, which defines that two proper names with totally different strings cannot be coreferential. According to the rank of each constraint, we can resolve such a conflict as explained in the next chapter. Suppose that the constraints have no score at all, we should consider removing one of them and ignore their great contribution in coreference resolution.

4.1.4. Unsupervised and Supervised approach

In this thesis, instead of popular single-link clustering, we view coreference as a multi-link clustering based on both classification and linguistic rules. Therefore we allow unsupervised learning approach and supervised learning approach to work harmoniously in coreference resolution.

Single-link clustering

In [Cardie and Wagstaff, 1999], they viewed coreference as clustering. Each cluster is an equivalence class including the referring expressions which refer to a common entity. Although in recent years, the most popular approach is supervised machine learning approach and not the clustering approach, the testing part of supervised machine learning approach seems like a special clustering algorithm, a classification-based single-link clustering algorithm. Single-link means that each anaphor only has one antecedent in a document. Consider the following example:

Sentence 4.5:

<S>While the state-owned French companies' rivals across the Atlantic have been ``extremely impressive and fast'' about coming together in mergers, European companies, hobbled by political squabbling and red tape, have lagged behind, (Gallois)$_1$ said.</S>

<S>…</S>

<S>``I think in the second step, we will have to consolidate at the level of the big groups,'' (he)$_2$ said.</S>

<S>The competition is even tougher for Aerospatiale in that the U.S. dollar has weakened 10 percent against the French franc last year, giving U.S. companies what (Gallois)$_3$ called a ``superficial'' advantage.</S>

Markable 1, 2 and 3 form a coreference chain. The part between “<S>” and “</S>” is a sentence determined by sentence segmentation. The example includes four sentences. According to the decision tree shown in Figure 3.1, link (2-3) can be recognized correctly because they agree in gender and their distance is no more than one sentence. But link (1-2) is missed because their distance is beyond the limitation in decision tree.

Here the single-link clustering model should be responsible for the missing pair. The single-link clustering model assumes that the current anaphor's antecedents, excluding the nearest one, have been in the coreference chain. It means that there are enough cues to introduce these antecedents into the coreference chain before testing current anaphor. According to the assumption, in Sentence 4.5, markable 1 should be found by markable 2, not by markable 3. However the assumption does not take noun phrase types into account. Besides distance, two markable's types also influence the intensity of their link. In Sentence 4.5, markable 1 and 3 are both proper names and markable 2 is pronoun. Therefore it is easier to find link (1-3) than link (1-2). In this case, single-link clustering results in some missing pairs.

Multi-link clustering based classification and constraints

Actually, one anaphor can have more than one antecedent. Therefore it is reasonable to take a current anaphor as a seed of a new cluster and add all markables which have direct links with it into the new cluster. Consider Sentence 4.5 again. Suppose that markable 3 is the current anaphor, its new cluster should include not only markable 2 but also markable 1. Markable 2 can be added into the cluster by decision tree's determination because it is the nearest antecedent to markable 3. But for markable 1, the rules of coreference decision tree are not reliable enough. Considering generation of training examples, a positive pair is formed by two adjacent referring expressions in a coreference chain. Therefore rules learned from training data are only suited to find the nearest antecedent. For those farther antecedents, they may not be good.

Coreference relation is distance-sensitive. Increasing distance can cause coreference link intensity to drop quickly. Accordingly, the rules, which are used to find farther antecedents, should be provided with higher reliability than those of decision tree. In this thesis, we make use of must-links to find farther antecedents. In Sentence 4.5, markable 1 is found by RC_ML1 (It is a must-link belonging to our must-links set. We will give its definition later). Besides high reliability, constraints are easy to be combined into a right-to-left-search also. Each time no more than two markables are tested based on a rule whether it belongs to constraints set or decision tree. By using the mixed rules, we view coreference task as a multi-link clustering task based on machine learning classification as well as linguistic rules.

Clustering is an unsupervised machine learning approach while classification is a supervised machine learning approach. By incorporating constraints, we make clustering and classification work harmoniously within a coreference system. Our experimental results show that incorporating constraints improves both recall and precision significantly. We will describe it later.

4.2. Ranked Constraints Definition

In this section, we give the detail of the ranked constraints used in our system. In this thesis we incorporate 4 groups of constraints to coreference system built through machine learning approach. They are: must-link (RC_ML), cannot-link (RC_CL), must-link-to- something (RC_MLS), and cannot-link-to-anything (RC_CLA).

4.2.1. Must-link

A must-link constraint specifies that two markables should belong to the same coreference chain. There are four must-links in RC_ML:

Proper Names and String Match (RC_ML1)

The must-link indicates that in a pair (i, j), if both markables are proper names and their strings match or one is the other's abbreviation, they can form a coreference pair and belong to the same coreference chain. We have included the proper name information and the result of string match in the feature vector. Therefore the must-link can be represented as the following: in a possible coreference pair's feature vector, if both PROPER_NAME and STR_MATCH are "1", or PROPER_NAME is "1" and one is the other's abbreviation, they form a coreference pair and belong to the same coreference chain.

Appositive Noun Phrases (RC_ML2)

The must-link constraint indicates that in a pair (i, j), if j is in apposition to i, then they form a coreference pair. It is difficult to detect appositive noun phrases correctly in a document. In our system, we use a set of rules to detect appositive noun phrases. We assume that in an appositive pair: one should be proper name, and the other should not be proper name; between i and j, there should be a comma and there is not any verb or conjunction; both markables should in the same sentence. In addition, we make use of two patterns to enhance the capability of detecting appositive noun phrases. One is "i (person), j , said(say)", the other is "i , j .". Appositive is a very important rule because it is the only rule representing coreference relationship between proper name and common noun phrase in our system. Actually, common noun phrases' coreference resolution is more difficult than that of proper names and pronouns. In error analysis, we will discuss the problem again.

Alias and String Match (RC_ML3)

The must-link constraint indicates that in a pair (i, j) , if both are proper names and i is an alias of j , but not abbreviation, or vice versa, they form a coreference pair. Like RC_ML1, we make use of the feature vector to obtain the parameters of RC_ML3. By doing so, the must-link (RC_ML3) is represented as the following: in a possible coreference pair's feature vector, if PROPER_NAME and ALIAS are both "1"s, and the pair cannot meet the requirement of RC_ML1, they form a coreference pair and belong to the same coreference chain.

Speaker and Speech (RC_ML4)

In general, those pronouns in speech between double quotation marks have to be transferred before referring to the antecedent which is not in the speech, because the sentences in the speech belong to a different domain (different speakers) from those sentences out of the speech. Consider singular first person pronoun appearing in speech between double quotation marks, they should refer to the speaker, even though the speaker's surface string is "he" or "she" (In general, "he", "she" and "I" should refer to different persons). More interestingly, singular third person pronoun appearing in the speech between quotations refers to different person from the speaker “he” or “she”. In this case, machine cannot easily resolve such problem without the help of constraints. In our system, we extract speaker and his speech from documents according to some reliable verbs at first, such as “said”, “reported” [Siddharthan, 2003] and then devise a set of constraints to resolve such problem, including must-links and cannot-links. In this section we introduce the must-links constraints. Cannot-link constraints about Speaker and Speech will be introduced in next section. RC_ML4 includes the following rules:

1) The speaker refers to first person pronouns appearing in his speech between quotations if there is no number disagreement.
2) In a speech between quotations, each pair of first person pronouns or second person pronouns without number disagreement is coreferential.
3) If two speeches appear in sequence in a document and the later speaker is a pronoun, the later speaker refers to the former speaker.

4.2.2. Cannot-link

A cannot-link constraint specifies that two markables can never form a coreference pair. Furthermore, they cannot belong to the same coreference chain. There are three cannot-links in RC_CL:

Proper Names with Totally Different Surface Strings (RC_CL1)

The cannot-link constraint indicates that in a pair (i, j), if both are proper names and their surface strings are totally different, they satisfy the cannot-link's conditions and cannot be in the same coreference chain. The "totally different" means there is no any common token shared by both two markables.

Common Root Markable (RC_CL2)

The cannot-link constraint specifies that in a pair (i, j), if the two markables have a common root markable, they cannot form a coreference pair and they cannot belong to the same coreference chain. According to the cannot-link, a markable cannot link to its nested noun phrases including its head noun phrase. And each pair of these nested noun phrases also satisfies the conditions of RC_CL2. Although in testing part, each pair of referring expression determined by decision tree or RC_ML cannot have a common root markable because we skip those markables with a common root markable with current anaphor when looking for the antecedent of it, it is still possible that two markables with a common root markable belong to the same coreference chain. For example, if A and B have a common root markable and (A-C) and (B-C) are coreference pairs, in this case, A and B belong to the same coreference chain by error. The purpose of RC_CL2 is to identify exactly such problem in a coreference chain.

Speaker and Speech (RC_CL3)

As we have explained in RC_ML3, the cannot-link constraint is to extract information from speaker and his speech. It can be satisfied if a pair reaches the following conditions:

1) A first person pronoun appearing in speech between quotations cannot refer

to the speaker if they disagree in number.

2) Those pronouns which are not first person pronouns appearing in speech between quotations cannot refer to the speaker if the speaker is singular.

Gender Disagreement (RC_CL4)

The cannot-link constraint identifies two markables cannot link together if they disagree in gender.

Semantic Class Disagreement (RC_CL5)

The cannot-link constraint identifies a pair cannot belong to the same coreference chain if both markables disagree in semantic class. Because of the confusion between organization and person name, we loosen the constraints on semantic classes (Our system think organization and person agree in semantic class). Considering the unreliability of semantic class information offered by our NLP pipeline, we give RC_CL5 the lowest score, -0.25. It is even lower than some probabilities obtained from the decision tree.

Number Disagreement (RC_CL6)

Like RC_CL4, the cannot-link constraint identifies a pair cannot belong to the same coreference chain if two markables of it disagree in number. Number information is not as reliable as gender information. Consequently, we give RC_CL6 a lower score than RC_CL4.

Article (RC_CL7)

The cannot-link constraint encodes rules that examine the articles used in i and j. In our system, we use the article constraints defined by [Wagstaff, 2002]. There are three rules about article in this cannot-link:

An indefinite markable cannot link backwards to a markable which is not a proper name or a pronoun.

A definite markable cannot link backwards to a markable without articles, unless it is a proper name or a pronoun or their head nouns match.

A markable without any articles cannot link backwards to a markable with articles, unless it is a proper name or a pronoun.

4.2.3. Markable-level constraints

Markable-level constraints have two types: must-link-to-something (RC_MLS) and cannot-link-to-anything (RC_CLA):

Must-link-to-something (RC_MLS)

As we know, pronoun should refer to something in a document, except some special pronouns, such as “it”. For example:

Sentence 4.6: Although different models of the F-14 have been involved in these mishaps, (**it**) is prudent to temporarily suspend routine flight operations for all F-14s in order to assess the available information and determine if procedural or other modifications to F-14 operations are warranted.

In the sentence, the “it” does not refer to anything. Occasionally our system cannot distinguish such case. In our system, must-link-to- something constraint applies to three kinds of pronouns: singular third person pronoun (“he”, “she”, and their corresponding possessive, accusative, reflexive pronouns), plural ambiguous pronoun (“they” and its corresponding possessive, accusative, reflexive pronouns) and “it” with its corresponding possessive, accusative, reflexive pronouns. If such pronouns cannot find any antecedent in its preceding document, we will collect a set of antecedent candidates according to specific rules and test these candidates from the nearest one to the farthest one. Once a candidate is accepted as antecedent of the pronoun, the remaining candidates are skipped.

The specific rules used in RC_MLS are more approximate and heuristic than pair-level constraints. For singular person pronoun, in its preceding document, all markables standing for a person are its antecedent candidates if there is no disagreement in gender and in number. For plural ambiguous pronoun, in its preceding document, all plural markables and markables standing for an organization are its candidates. For “it” and its corresponding pronouns, all singular nonhuman markables are its candidates.

Cannot-link-to-anything (RC_CLA)

According to MUC-7 [MUC-7, 1997] Coreference Task definition, a coreference relation only involves expressions which refer to a given entity. And up to now, coreference task only deal with identical coreference relationship. Set/subset and part/whole coreference relations have not been considered now. Accordingly, we can filter some markables in advance which have no possibility to take part in a coreference relation at all. Cannot-link-to-anything constraint specifies such markables. In our system the following markables satisfy cannot-link-anything constraint's conditions: a markable only including figures which is not currency, percentage, date or time, and common noun phrases beginning with "no", figures or some quantitative indefinite adjectives (Such as "few", "little", "some", "any", "many", "much", "several"). And those markables which have the same head nouns with above markables also satisfy the constraint's conditions.

4.3. Multi-link Clustering Algorithm

The conflict resolution (it will be described in the next chapter) requires constraints to be ranked reasonably. In our system, we give each pair-level constraint a suitable score based on the reliability of the constraint (See Table 4.1). The score not only allow ranking of all pair-level constraints, but can also be a critical criterion to complete the conflict resolution. From Table 4.1, we see that must-link constraints have positive scores and cannot-link have negative scores. Must-link-to-something constraint has a relatively low score, only 0.5. This means must-link-to-something constraint is not as reliable as the rules of decision tree. Cannot-link-to-anything constraint does not have a specific score because it is a filter rule with the highest rank. It cannot be violated. Among the links with specific scores, the link provided with the highest score, 999, is similar to a hard constraint which cannot be violated. These scores as well as probabilities offered by decision tree are the inputs to conflict resolution.

Given the constraints definitions and their scores, we can describe how such ranked constraints are embedded into a coreference system built with machine learning approach. The rough algorithm is shown in Figure 4.1. In the algorithm, we filter those markables satisfying cannot-link-to-anything's conditions before main coreference resolution. Then we build two tables, a must-link table and a cannot-link table. In the main coreference resolution part, for each anaphor, we first form a cluster. Besides the antecedent determined by the decision tree, we add into the cluster all markables which must link to the anaphor through checking the must-link table. Then one by one, we insert each member of the cluster into the existing coreference chains. Due to the

Type	Name	Score	Description
Must Link	RC_ML1	999	Proper name and string match
	RC_ML2	899	Appositive
	RC_ML3	850	Proper name and alias
	RC_ML4	999	Speaker and his speech
Cannot Link	RC_CL1	-799	proper name with totally different strings
	RC_CL2	-989	Common root markable
	RC_CL3	-999	Gender disagreement
	RC_CL4	-899	Speaker and his speech
	RC_CL5	-0.5	Number disagreement
	RC_CL6	-0.25	Semantic class disagreement
	RC_CL7	-1	Articles
Must Link to Something	RC_MLS	0.5	"he","she","they","it" and their corresponding pronouns must link to something before them
Cannot Link to Anything	RC_CLA	/	Figures, common noun phrase beginning with figures, indefinite adjective and "no" can not link to anything

Table 4.1
Ranked Constraints set used in our system

```
Algorithm Find-Antecedent ($MARK$ : set of all markables)
$i := 0$;  $Coref := \Phi$;
for $M_i \in MARK$ do
    if $CLA(M_i) =$ "true" then
        $MARK := MARK \setminus \{M_i\}$
    else
        $ML_i := \{ M_j(Sco_{ij}) :\ i > j\ \&\& ML(M_j, M_i) =$ "true" and $M_j \in MARK \}$
        $CL_i := \{ M_j(Sco_{ij}) :\ i \neq j$ and $CL(M_j, M_i) =$ "true" and $M_j \in MARK \}$
    $i := i + 1$;
$i := 1$;
for $M_i \in MARK$ do
    $Cluster_i := ML_i \cup \{ M_j(Sco_{ij})$ :the antecedent decided by coreference decision tree$\}$
    for $M_j \in Cluster_i$ do
        $Coref := Add\ (Coref,\ M_j, M_i, CL_i,\ CL_j, Sco_{ij})$
    if $M_i \notin Coref$ and $M_i$ is corresponding pronoun to "he", "she", "they" or "it" then
    $Cluster_i = \{ M_j(Sco_{ij}) : MLS(M_j, M_i) =$ "true" and $i > j$ and $M_j \in MARK \}$
    for $M_j \in Cluster_i$ do
        $Coref := Add\ (Coref,\ M_j, M_i, CL_i,\ CL_j, Sco_{ij})$
    $i := i + 1$;
return $Coref$
```

Figure 4.1
The Algorithm of Coreference Chains Generation with Ranked Constraints. Coref is the set of coreference chains existing. The four functions, ML, CL, MLS and CLA, check that whether two markables satisfy must-link, cannot-link, must-link-to-something or cannot-link-to-anything or not, respectively. Sco is the score of the constraint. Add function includes the conflict resolution (described in next chapter).

existence of conflicts, it is not certain that the anaphor can be added into any of coreference chains successfully. If an anaphor fails to be added into coreference chain and it satisfies must-link-to-something constraint's conditions, the coreference system will use must-link-to-something constraints to build a new cluster for the anaphor and then add each member of the cluster into coreference chains by the same way. Note that each member of the new cluster is also checked by conflict resolution when trying to add them into coreference chains. Inserting stops after we first find that a member of the cluster is accepted by the coreference chain As a result, it is still not certain that the anaphor which must link to something can be added into one chain successfully. The processing of adding coreference pairs into coreference chains is very critical for coreference chains generation. It not only filters out some error pairs, but also rearranges current coreference chains in order to remove some error links existing in current coreference chains and obtain back some missing links. By doing so, we can achieve a reasonably high precision. In the next chapter, we will explain how it is done in detail.

5. Conflict Resolution

As we have mentioned above, coreference system built through machine learning approach may encounter some contradictory pairwise classifications while generating coreference chains. For example, classifier determines two links, (A-B) and (B-C), whereas A and C are not coreferential actually. Most systems do not take such problem into account except [Ng and Cardie, 2002]. They proposed an error-driven rule pruning algorithm that optimizes the coreference classifier rule-set with respect to the clustering-level coreference scoring function. But language is infamous for its exception of rules. 100% accuracy rule-set does not exist. Therefore such contradictory pairwise classification may still possible to appear in coreference chains. In this chapter, we propose a new approach to resolve such contradictory pairwise classifications. The approach with ranked constraints can achieve a reasonable result that is better than most coreference systems.

In Section 5.1, we will define the concept of “conflict” used in this thesis. And we will explain how the approach can improve the performance of the coreference system. Next, we will give the details about the approach.

5.1. Conflict

A conflict appearing in a coreference chain is a contradictory pairwise classification as we have mentioned above. (A-B) and (B-C) are determined as coreference pairs by coreference system, whereas A and C are not coreferential actually. Consider the following example extracted from the output of our baseline system on MUC-7 [MUC-7, 1997] formal documents:

Sentence 5.1:

``This deal means that (Bernard Schwartz)$_1$ can focus most of (his)$_2$ time on Globalstar and that is a key plus for Globalstar because (Bernard Schwartz)$_3$ is brilliant," said (Robert Kaimowitz)$_4$, (a satellite communications analyst)$_5$ at Unterberg Harris in New York.

In the example, 5 markables tagged belong to a common coreference chain in our baseline system. (2-3) and (2-4) are recognized as coreference links. But we see that markable 3 and 4 obviously refer to different entities. This is a conflict. The conflict is caused by the error link between markable 2 and 4.

Human can distinguish a conflict easily, but it is not easy for a machine. How to decide that two markables are not coreferential is the key of to detect a conflict. Using the decision tree is one choice. But it is not reliable and it can even degrade the performance of coreference system. Because decision tree is used to find the nearest antecedent, other antecedents are difficult to be determined by decision tree. For example, if a “Robert Kaimowitz” appears in the next sentence of Sentence 5.1. The decision tree will determine that the “Robert Kaimowitz” is coreferential with markable 4 because of string match. But the “Robert Kaimowitz” and markable 5 are recognized as negative. If we use decision tree to detect conflict, markable 4, 5 and the

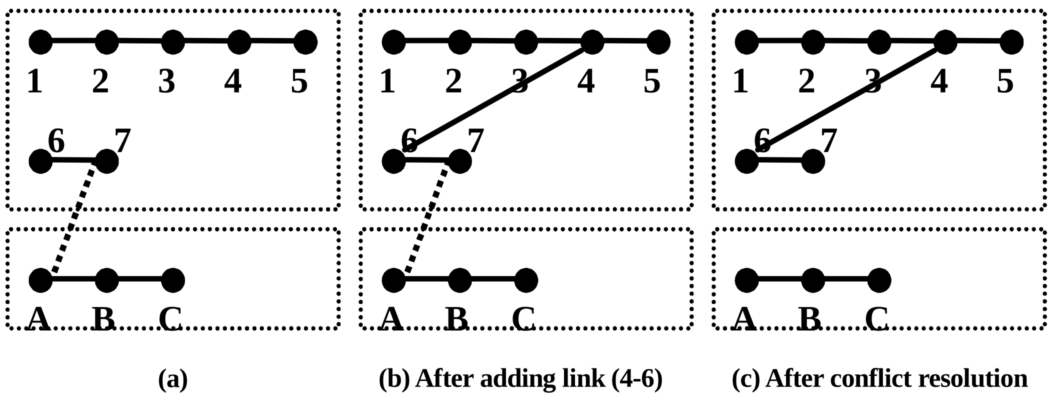


Figure 5.1:
An example of conflict resolution. Actually, there are two coreference chains. One is (1, 2, 3, 4, 5, 6, 7), the other is (A, B, C). (a) shows the coreference chains before inserting the link between 6 and 7. The link draw by broken line is an error link determined by coreference system. (b) is the case after addling link (4-6) and before conflict resolution. (c) is the result after conflict resolution.

"Robert Kaimowitz" form a conflict. But there is no conflict existing among them at all. As we can see, using decision tree to detect conflict is not desirable. In this thesis, we use a set of ranked cannot-link constraints to detect conflicts in coreference chains. If two markables in a coreference chain satisfy the conditions of any cannot-link constraint, there is a conflict existing in the coreference chain.

Before we introduce the detailed algorithm of conflict resolution, we discuss that how the conflict resolution can improve the performance of coreference system. See Figure 5.1:

There are actually two coreference chains in the figure. One is (1, 2, 3, 4, 5, 6, 7). The other is (A, B, C). However, (a) shows the result of coreference system. We see that there is an error link between 7 and A. According to the definition of recall and precision [Baldwin, 1995] used in MUC-7 [MUC-7, 1997]:

$$\mathrm{Re}\,call = \frac{\sum(|S_i| - |p(S_i)|)}{\sum(|S_i| - 1)}$$

S_i is the i-th coreference chain generated by the key offered by MUC-7 [MUC-7, 1997]. $p(S_i)$ is a partition of S_i relative to the response. And precision is computed by switching the roles of the key and response in the above formulation.

According to the two formulations, (a)'s recall and precision are both 87.5%. After adding the link (4-6), the recall increases into 100 % and the precision is about 88.9%. As we can see, although there is no referring expression missed, the precision is still below 100%. It is mainly because there are still some spurious links existing in the chains. The conflict resolution is to rearrange current coreference chains. By remove spurious links, the approach enhances the performance of coreference system. Figure 5.1 (c) shows the result of conflict resolution. We see that after adding the new link, the system detects a conflict existing in coreference chain (1, 2, 3, 4, 5, 6, 7, A, B, C) and call conflict resolution module to decide how to deal the conflict. In this example, link (7-A) is cut. By doing so, the conflict disappears and the precision increases into 100% without any loss of recall.

As we can see, conflict resolution improves the performance of coreference system through referring expressions rearrangement in a coreference chain with conflicts. The approach contributes a lot to precision.

5.2. Main Algorithm

Each time a new coreference pair is inserted into coreference chains, conflict resolution module will be called to detect conflicts and resolve conflicts. The module checks every updated chain and the new chain just formed by the pair. If a conflict is detected in a coreference chain, for each two referring expressions which satisfy one cannot-link in the coreference chain, conflict resolution will find a path in the chain to link the two conflicting referring expressions. Each path will cover some links in the chains. As a result, those links covered by all paths consist of a common conflict path. In the common conflict path, the link, which has the lowest score minus conflict score (the sum of cannot-link pairs' scores appearing in the chain), will be removed. Consequently, all cannot-link constraints existing in the chain are separated in the link. The conflicts are resolved and the chains are rearranged. In order to resolve a conflict by removing only one link, we make some changes to coreference chain's data structure.

5.2.1. Coreference tree

Chain vs. Tree

In this thesis, we propose a concept of "coreference tree", which is different from "coreference chain". Actually, coreference chain used in most systems is just an equivalence class. The relationship between each two referring expressions is not included in an equivalence class. It means that once a coreference pair is added into a coreference chain successfully, the link of the pair is no longer used. A coreference chain is maintained as a set of isolated referring expressions. All referring expressions in a coreference chain do not link together until the document has come to the end and all coreference chains are generated completely.

As we have explained above, conflict resolution involves a process of searching for a path between two members in a cluster. Such a “coreference chain” cannot meet our requirement. Therefore we use “coreference tree” instead of “coreference chain” (In the remaining of the thesis, we will use “coreference tree” in place of “coreference chain”).

The coreference tree includes the information of coreference links and these links’ scores. And for each link, if it is the only link in the coreference tree, the referring expression in the preceding the other expression in the document is called the parent and correspondingly the other expression is called child. If the link is not the only link, the expression which is inserted into the tree earlier than the other is called parent. By doing so, we give each link in a coreference tree a direction: the child expression links to parent expression. Furthermore, we have mentioned in NLP part that before adding a coreference pair into one coreference tree, the system will check each expression of the pair to see whether among all existing coreference trees there is already a markable which has a common head noun with the expression. If the system makes sure that the expression has not existed in any coreference tree, the expression can be added into one tree as a new member. We call the processing “existence check”. Existence check guarantees that each expression only appears once in coreference trees. It means that it is impossible that there is an expression simultaneously appearing in two difference coreference trees. And there is no expression appearing twice in one coreference tree. With the “One Appearance” guarantee, we can make sure that in any coreference tree, each expression has only one parent or no parent at all. But for each parent, it has no less than one child or has no child at all. After these definitions, a coreference chain can be changed into coreference tree.

Coreference tree has the same characteristics with general trees. For each two members in a tree, a path can be found. And removing any link can make the tree to be separated into two parts. The two characteristics are the important foundations of our conflict

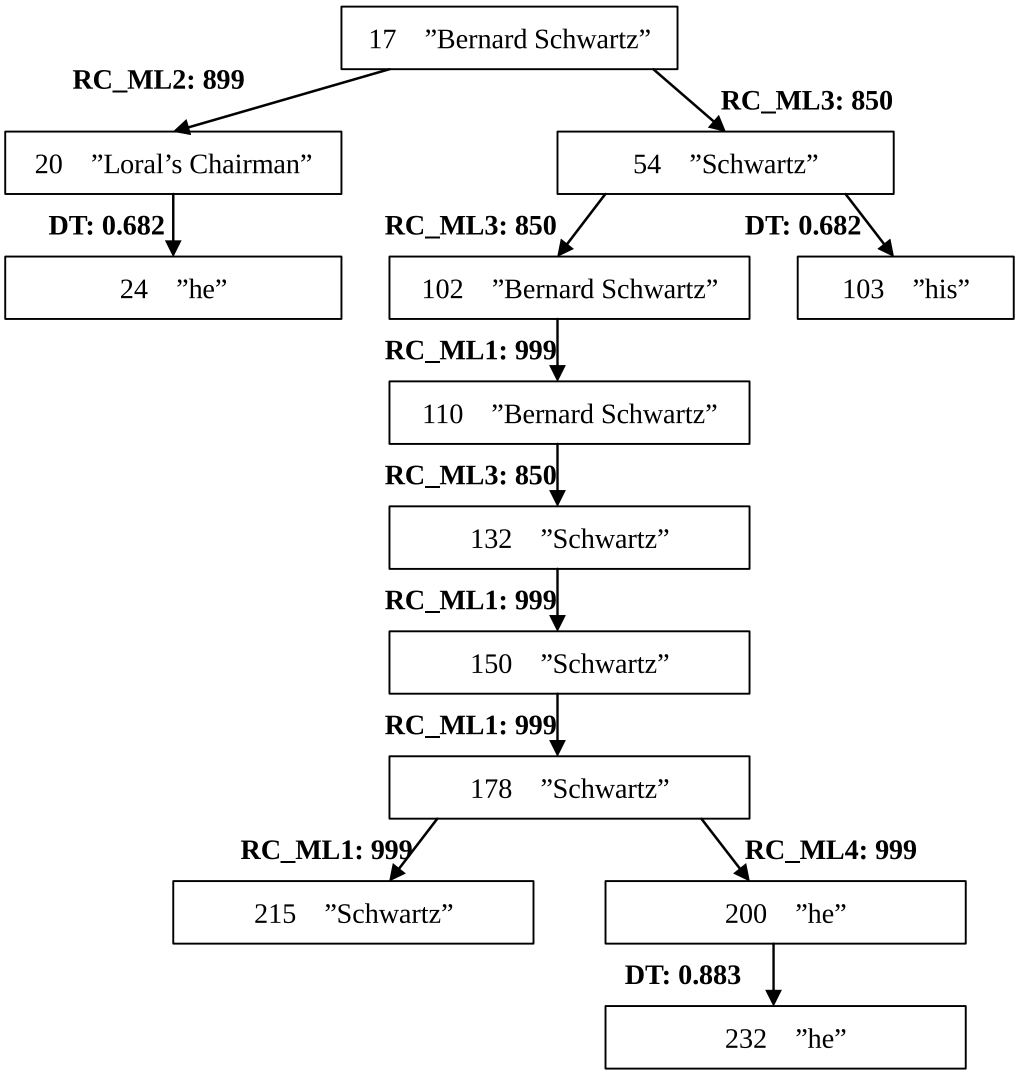


Figure 5.2:
An example of coreference tree in MUC-7. Each rectangle stands for a referring expression in the coreference tree. In each rectangle, markable ID and surface string are given. The bold string beside arrow is the link type and corresponding score. “DT” means decision tree result. The other types have been described in Table 4.1

resolution, which will be used in two subroutines, extending trees and merging trees.

An Example of Coreference Tree

If we view an equivalence class as a tree, the expressions in the class are nodes of the tree and the similarity between two expressions is an edge.

An example of coreference tree on MUC-7 [MUC-7, 1997] is shown in Figure 5.2. The example is extracted from the output of our complete system. It is desirable that the tree is consistent with human knowledge. In Figure 5.2, we see that there is no link beginning with a pronoun. And proper names are linked together according string match or alias rule.

5.2.2. Conflict Detection and Separating Link

For simplicity, here we view the coreference tree as a graph without cycle. In the graph, all members are separated into two groups: S_a, S_b. The algorithm to detect the conflicts existing between the two groups and find the separating link is explained in Figure 5.3. As we know, two members of a tree must have a path between them and only have one. For each expression of S_a which forms a cannot-link with any member of S_b, we find the path between the two members. Next the corresponding cannot-link's score is recorded also. The system sums up all scores to obtain the *ConflictScore* between the two groups. And all paths are combined to obtain the *CommonPath* (including the links covered by all paths). Among all links in the *CommonPath*, the link, which is with the lowest score after adding the *ConflictScore* (*ConflictScore* is negative

```
Algorithm Conflict-Detection ( $S_a$ , $S_b$ : markable groups; $NoncuttingSco$ : a threshold defined in advance)
$ConflictScore$ :=0; $CommonPath := \Phi$

for $M_i \in S_b$ do
    for $M_j \in S_a$ do
        if $CL(M_i, M_j) =$ "true" then
            $Path := FindPath(M_i, M_j)$
            $CommonPath := CommonPath \cap Path$
            $ConflictScore := ConflictScore + Score(CL(M_i, M_j))$
$SeparatingLink := \Phi$ ; $SeparatingLinkScore$ := 9999
for $Link_i(Sco_i) \in CommonPath$ do
    if $SeparatingLinkScore > Sco_i$ then
        $SeparatingLinkScore = Sco_i$ ;
        $SeparatingLink := Link_i$ ;
    else
        if $SeparatingLinkScore == Sco_i$ && $SeparatingLink \neq \Phi$ then
            if Distance( $SeparatingLink$ )< Distance ( $Link_i$ ) then
                $SeparatingLinkScore = Sco_i$ ;
                $SeparatingLink := Link_i$ ;
if $SeparatingLinkScore + ConflictScore < NoncuttingSco$ then
    return $SeparatingLink$ ;
else
    return $\Phi$ ;
```

Figure 5.3
The Algorithm to detect conflict and find separating link.

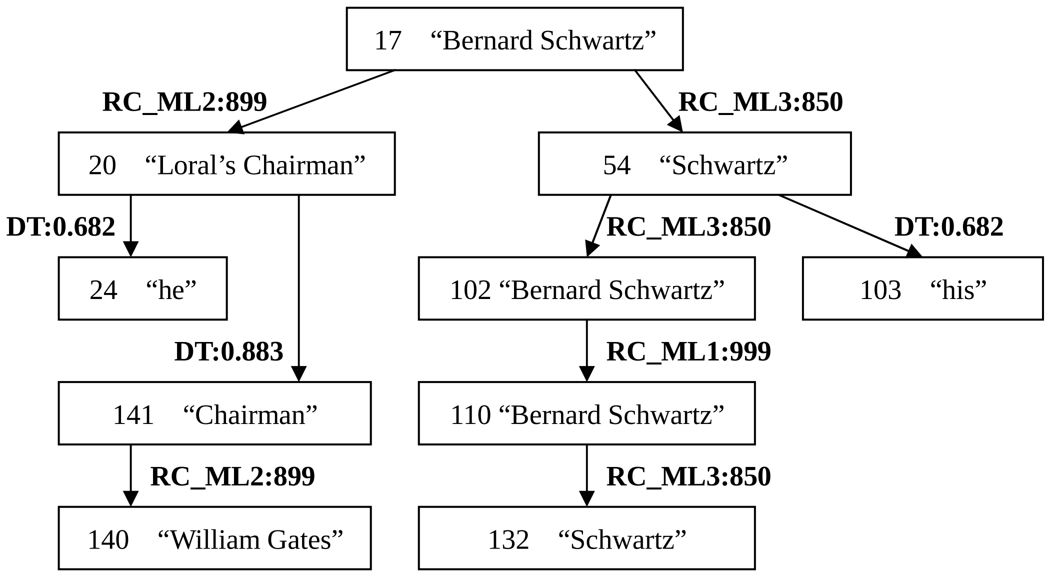


Figure 5.4
An example of extending coreference tree. 140 is the new expression for the tree. (54, 102, 110, 17, 132) is 140's objective set and 140's objective score is -3995. Objective common path is (20-141-140). The link to be removed is (20-141).

because cannot-link's score is negative. Consequently, we add not subtract), will be considered for removal. If there is more than 1 link with the lowest score, the distance between two members of the link in the document is taken into consideration. The link with greater distance will be chosen as *SeparatingLink*. In order to make a choice between to cut and not to cut, we give the system a threshold, *NoncuttingSco*, in advance. If the *SeparatingLink* is still stronger than the threshold, then the system decides not to cut this tree. As we have mentioned above, for a tree, cutting any link can separate the tree into two parts. Partitioning a tree is equivalent to find a separating link. After separating, all objecting expressions to the new expression are separated from it. As a result, it costs only one link to resolve a conflict.

We use an example to explain the Conflict Detection (Figure 5.4). In the example, markable 140 is the only expression of S_a. After checking the cannot-link table, there are 5 expressions (54, 102, 110, 17, 132) in S_b objecting to markable140 because of RC_CL1. Therefore the objecting set is (54, 102, 110, 17, 132) and the *ConflictScore* is -799*5=-3995. For 54, the path between it and markable140 is (54-17-20-141-140). Like 54, we can find other paths between remaining objecting expressions and 140. In the 5 paths, they share 3 links, (17-20), (20-141) and (141-142). Therefore the *CommonPath* is (17-20-141-140). Among the three links, link (20-141) has the lowest score (3994.117). Hence link (20-141) is *SeparatingLink*. After removing link (20-141), the conflict disappears. Here the conflict resolution makes a right decision.

5.2.3. Manipulation of Coreference Tree

The generation of coreference trees includes 4 manipulations: creating, extending, separating and merging. They are used in “Add” function in the algorithm of Coreference Chains Generation with ranked constraints (Figure 4.1) Figure 5.5.

Creating Coreference Tree

If the existence check tells the system that both members in a pair do not appear in any current coreference tree, the system begins to create a new coreference tree which only includes the pair. The expression with smaller markable ID will be the parent of the other. Then Conflict-Detection (see Figure 5.3) is called to check the new coreference tree. Here, S_a and S_b include the two members of the pair, respectively. If Conflict-Detection does not return null, the new tree is removed from coreference trees.

$Add\ (Coref,\ M_j, M_i, CL_i,\ CL_j, Sco_{ij})$

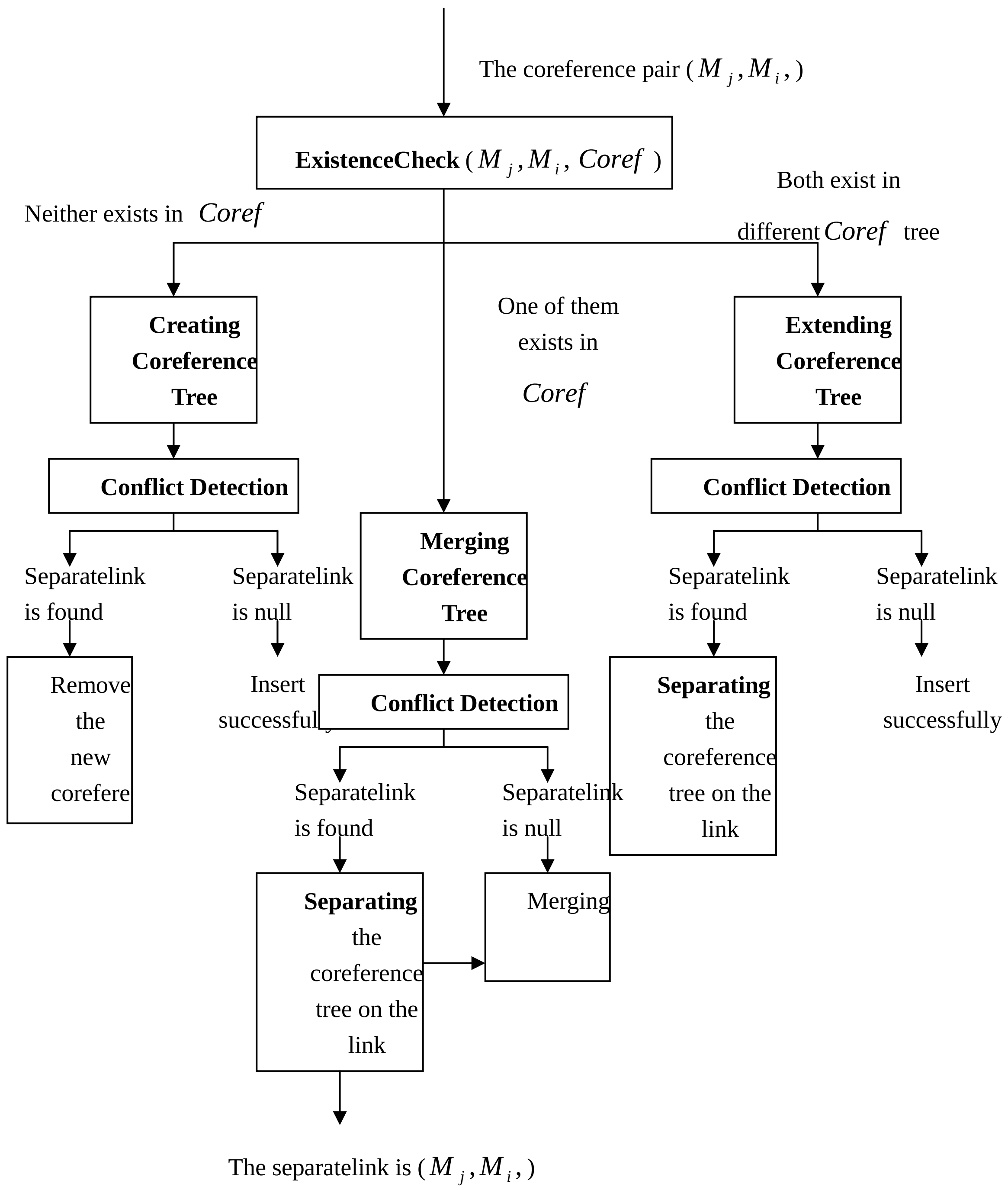

The coreference pair ($M_j, M_i,$)
ExistenceCheck ($M_j, M_i, Coref$)
Neither exists in $Coref$
Both exist in different $Coref$ tree
One of them exists in $Coref$
Creating Coreference Tree
Extending Coreference Tree
Conflict Detection
Conflict Detection
Merging Coreference Tree
Separatelink is found
Separatelink is null
Separatelink is found
Separatelink is null
Remove the new corefere
Insert successfully
Conflict Detection
Separating the coreference tree on the link
Insert successfully
Separatelink is found
Separatelink is null
Separating the coreference tree on the link
Merging
The separatelink is ($M_j, M_i,$)

Figure 5.5:
The Add function of the algorithm of Coreference Chain Generation.

Extending Coreference Tree

If existence check tells the system that one member of a pair belongs to one coreference tree but the other does not appear in any coreference tree, the system calls extending subroutine to add the new member to the coreference tree including the other member already. The new member will be added into the tree as one child of the other member which has already existed in the tree. Next, the conflict resolution will be called to check the updated tree. Because our system will call conflict resolution to check a tree each time a new expression is inserted into the tree, there is no conflict among expressions excluding the new member which is just inserted. Therefore the conflict resolution only checks the conflicts between the new member and other expressions. If *SeparatingLink* is found, our system calls separating subroutine to separate the tree.

Merging Coreference Trees

Merging coreference trees is similar to extending coreference tree. If two members of a pair exist in two different coreference trees, the merging subroutine will be called to deal such problem. Given a pair (A, B) which leads to a merging process, let TA and TB be the trees of A and B, respectively. At first, we link A and B temporarily. Then call Conflict-Detection to detect the conflicts existing between the TA and TB. After that, remove the temporary link (A-B) between two trees. If *SeparatingLink* is exactly (A-B), nothing will be done in the merging processing. If *SeparatingLink* belongs to TA, the system separate TA on *SeparatingLink* at first and then add the part including A into TB. The same process is done when *SeparatingLink* belongs to TB.

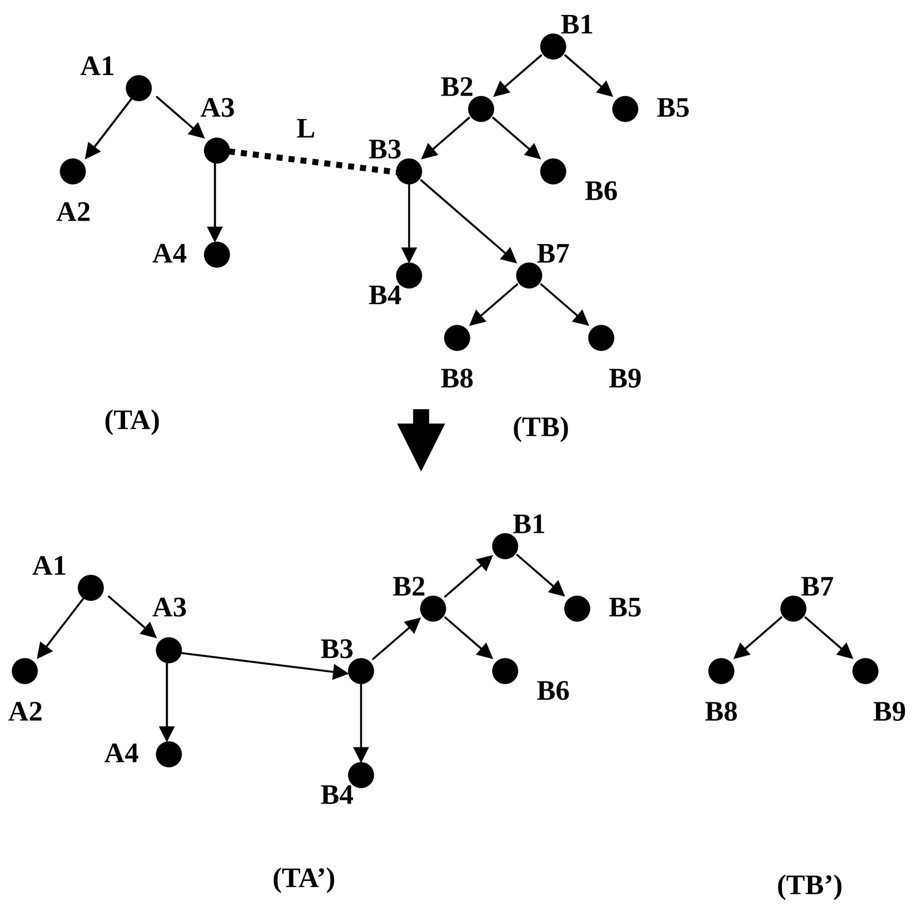


Figure 5.6:
An example of merging coreference trees. TA and TB are two trees. Link L leads to the merge of the two trees. After merging, two new trees are generated, TA' and TB'.

It should be noticed that we should change some of the tree's links before adding them into another tree. In order to guarantee tree structure, we should change some links' directions. Given two trees TA and TB, we need to add TA into TB on link (A-B). In link (A-B), B belonging to TB should be parent of the link. If A belonging to TA has a parent in TA already, there would appears two parents of A in the new TB after adding TA into TB. Therefore before adding TA into TB, we search the path from A to TA's root and reverse directions of all links on the path. It means to exchange parent and child roles in each link. By doing so, the new tree is still a tree. Consider the following example:

Two trees (TA and TB) need to merge on the link L. Among expressions in TA, A1 is objected by B8 and B7. Its path is (A1-A3-B3-B7) and score is S1. A2 is objected by B9. Its path is (A2-A1-A3-B3-B7-B9) and score is S2. Then the common path for TA is (A1-A3-B3-B7) and the objecting score to TA is S1+S2. After checking each links score minus (S1+S2) in the common path, we find the link (B3-B7) is the weakest. Hence we separate TB on (B3-B7) at first and get Ttemp and TB’. Ttemp includes B3. Before we add Ttemp into TA, we reverse the links, (B3-B2) and (B2-B1), covered by the path from B3 to root B1 (B3-B2-B1). After changing their directions, we add

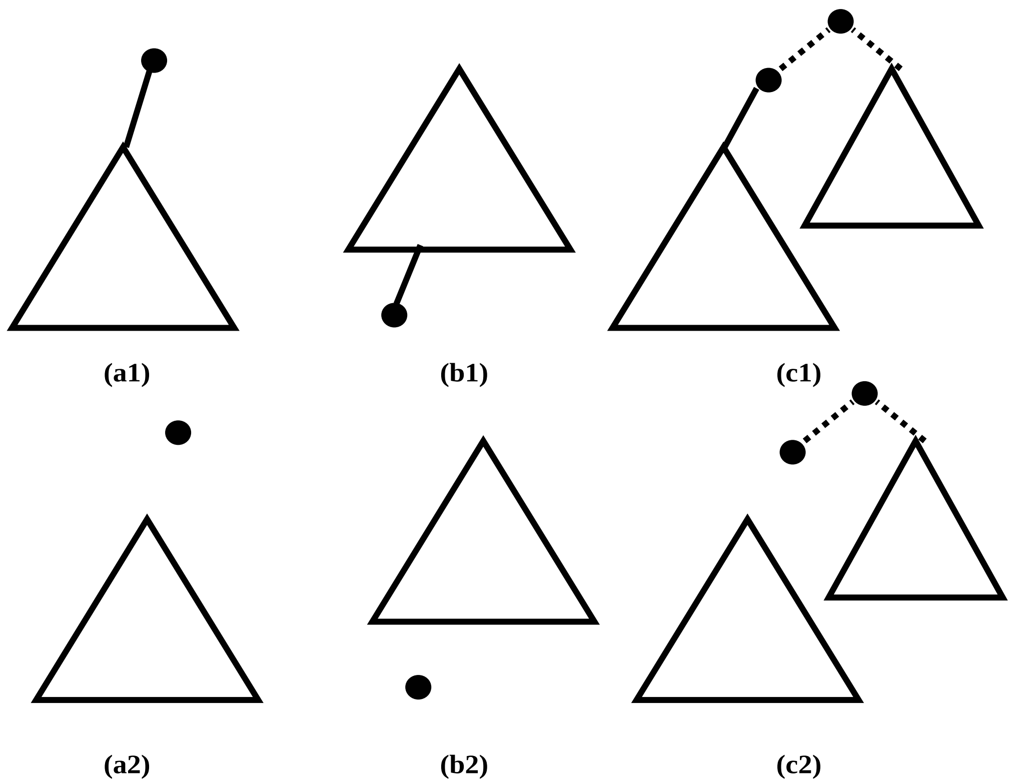
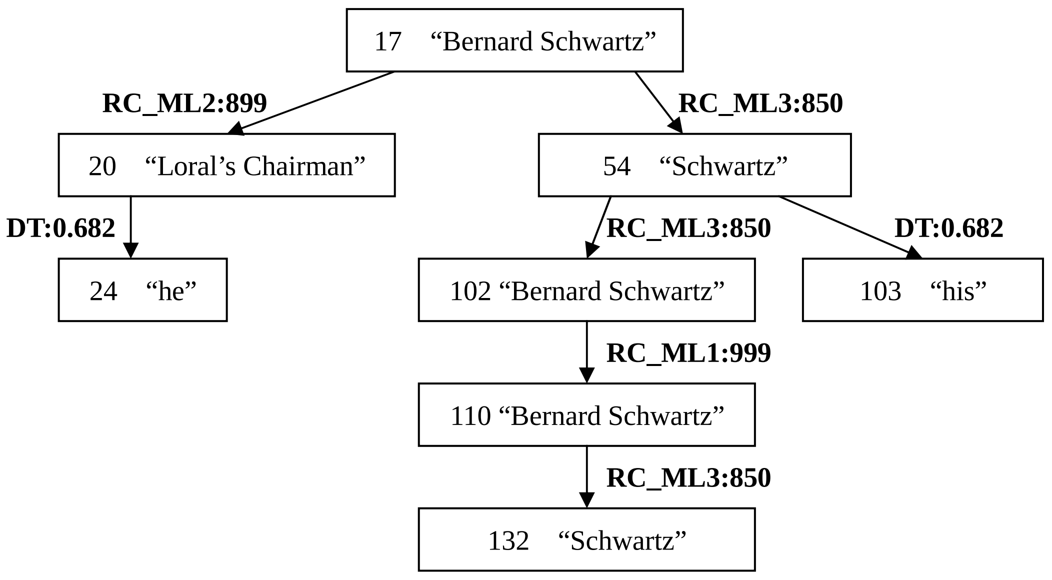


Figure 5.7:
Examples of separating coreference tree. The bold line is considered to be removed. (a1), (b1) and (c1) show the trees before separating. (a2), (b2) and (c2) show the trees after separating.

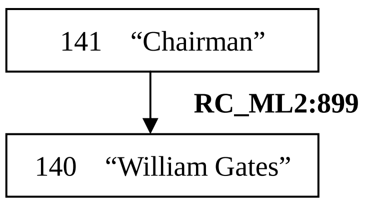
Ttemp into TA on L. And make A3 to be B3's parent. The result of this merging is TA' and TB', which is shown in Figure 5.6.

Separating Coreference Tree

Given a coreference tree and a link of the tree, we can cut the tree into two parts on the link. There are three cases when separating a tree on a specific link. See Figure 5.7. The first case is shown in Figure 5.7 (a1) and (a2). The bold line is separating link, which includes the root expression of the tree. And the root has only one sub-tree



(a)



(b)

Figure 5.8:
The result of separating the tree with conflict shown in Figure 5.4. The link between 20 and 141 has been removed. And two trees are generated as shown in (a) and (b).

linked by the bold line. After separating, the root becomes isolate point and consequently it is removed from coreference trees. And remaining part takes place of the old one. The second case is shown in Figure 5.7 (b1) and (b2). One member of the bold line is a leaf. Consequently after removing the separating link, the leaf is also removed from coreference trees. The third case (Figure 5.7 (c1), (c2)) is that after removing the bold line, each part is still a tree.

For example shown in Figure 5.4, after removing link (20-140), two new trees generate. The result after separating processing is shown in Figure 5.8.

We observed that after rearranging the expressions in current coreference tree, we obtain a more accurate result.

6. Evaluation

Our coreference resolution approach is evaluated on the standard MUC-7 [MUC-7, 1997] data set. For MUC-7 [MUC-7, 1997], 30 dryrun documents annotated with coreference information are used as training data. There are also 20 formal documents from MUC-7 [MUC-7, 1997]. For testing, we use the formal data as our input. The performance is reported in terms of recall, precision, and F-measure using the model-theoretic MUC scoring program. Our ranked constraints and conflict resolution produce scores which are higher than those of the best MUC-7 coreference resolution system and earlier machine learning systems, such as [Soon et al., 2001] and [Ng and Cardie, 2002a]. And F-measure increases with regard to our duplicated Soon baseline system from 60.9 to 64.2 for MUC-7/C4.5.

In this chapter, we will describe our experimental results as well as those of some earlier machine learning systems. Next, we will discuss the contributions to coreference system of ranked constraints and conflict resolution, respectively. In the last section, the errors remaining in our coreference system will be analyzed.

6.1. Score

As we have mentioned in Chapter 3, we use C4.5 to learn a classifier based on MUC-7 [MUC-7, 1997] 30 dryrun documents. The annotated corpora produce 44133 training pairs, of which about 3.5% are positive pairs. By using 60% pruning confidence, we

System	MUC-7 formal		
	R	P	F
Soon et al.(2001)	56.1	65.5	60.4
Ng and Cardie (2002a)	57.4	70.8	63.4
Ng and Cardie (2002b)	59.7	69.3	64.2
Ng and Cardie (2002)	54.2	76.3	63.4
Yang et al. (2003)	50.1	75.4	60.2
Ng and Cardie (2003)	53.3	70.3	60.5
Duplicated Soon Baseline	59.6	62.3	60.9
Ranked Constraints (RC)	63.5	64.5	64.0
RC and Conflict Resolution	63.7	64.7	**64.2**

Table 6.1:
Results for MUC-7 formal data in terms of recall, precision and F-measure. Results in boldface indicate the best results obtained for a particular data set and decision tree by using a particular constraints group and conflict resolution.

get a decision tree shown in Figure 3.1.

Based on MUC-7 20 formal documents, results of our system are shown in Table 6.1.

For comparison, Table 6.1 shows some other coreference systems' best performances given out in corresponding papers. [Soon et al., 2001] achieved 60.4 in F-measure based on a set of 12 features and a classifier learned by C5.0. [Ng and Cardie, 2002a] improved upon [Soon et al., 2001]'s model by expanding the feature set from 12 features to 53 features, and introducing a new training instance selection approach and a new search algorithm that searches for antecedent with highest coreference likehood value. They increased their F-measure from 61.6 to 63.4 for MUC-7/C4.5 by using a hand-selected features set instead of all 53 features. [Ng and Cardie, 2002b] incorporated an anaphoricity classifier into [Ng and Cardie, 2002a]'s model. And in order to overcome the loss in recall caused by the anaphoricity classifier, they also incorporated two constraints, STR_MATCH and ALIAS, to increase F-measure from

58.4 to 64.2. [Ng and Cardie, 2002] is another attempt to improve coreference model. By using a new positive sample selection approach and error-driven pruning, they achieved 63.4 in F-measure. [Yang et al., 2003] proposed a promising twin-candidate model instead of single-candidate model although their score drops behind those of former systems. And [Ng and Cardie, 2003] focused the resolution of weakly supervised learning for coreference task through self-training or an EM with feature selection. The six coreference systems are only machine learning-based systems we could find, which reported their scores based on MUC-7 formal data with F-measure above 60%.

From Table 6.1, we see that our complete coreference system with ranked constraints and conflict resolution has achieved a recall of 63.7% and a precision of 64.7%, yielding a balanced F-measure of 64.2%. The F-measure is the highest score among those of the systems listed in Table 6.1. And with regard to our duplicated Soon baseline system, the recall increases 4.1% from 59.6% to 63.7% and the precision increases 2.4% from 62.3% to 64.7%, resulting in a significant increase of 3.3% in F-measure. It is interesting to note that the complete system achieves the highest recall among all the systems in Table 6.1, but the lowest precision compared to others. One reason for the highest recall is that our NLP pipeline includes two additional modules: head noun phrase extraction and proper name identification (the corresponding experimental results are shown in Table 2.1). It makes the recall of duplicated Soon baseline system is higher by 3.5% than [Soon et al., 2001]'s. The other reason is that our must-links and must-link-to-something introduce some spurious links into the system. The corresponding experimental results will be shown in next section. For the

System	MUC-7 dryrun			MUC-7 formal		
	R	**P**	**F**	**R**	**P**	**F**
Duplicated Soon Baseline	**57.4**	**64.7**	**60.9**	**59.6**	**62.3**	**60.9**
Only Pronoun	13.3	70.2	22.3	10.6	60.0	18.0
Only Proper Name	25.1	84.9	38.7	29.6	81.4	43.4
Only Common Noun phrases	26.7	52.0	35.2	26.8	49.3	34.7
Ranked Constraints (RC)	**59.5**	**66.7**	**62.9**	**63.5**	**64.5**	**64.0**
Only Pronoun	15.9	62.9	25.4	13.9	55.6	22.3
Only Proper Name	26.9	86.1	41.0	31.8	82.3	45.8
Only Common Noun phrases	26.3	57.0	36.0	26.5	54.1	35.6
RC and Conflict Resolution	**59.8**	**67.2**	**63.3**	**63.7**	**64.7**	**64.2**
Only Pronoun	15.9	63.1	25.4	13.9	55.6	22.3
Only Proper Name	26.9	86.1	41.0	31.8	82.5	46.0
Only Common Noun phrases	26.4	57.2	36.1	26.7	54.4	35.8

Table 6.2
Results for baseline and complete systems to study the effects of ranked constraints and unsupervised conflict resolution. For each of the NP-type-specific runs, the overall coreference performances are measured by restricting anaphor to be of the specified type.

lowest precision, one reason is the decision tree we use. Our baseline system and our complete system use a common decision tree listed in Figure 3.1. We see that the precision of our baseline system is very low already. Therefore the low precision in our complete system has no relation to ranked constraints and conflict resolution. Furthermore, higher recall tends to result in lower precision. However, the F-measure increased showing that the sacrifice of precision is tolerable.

A closer examination of the results is shown in Table 6.2. In the table, three systems are evaluated. Besides our duplicated Soon baseline system and the complete system,

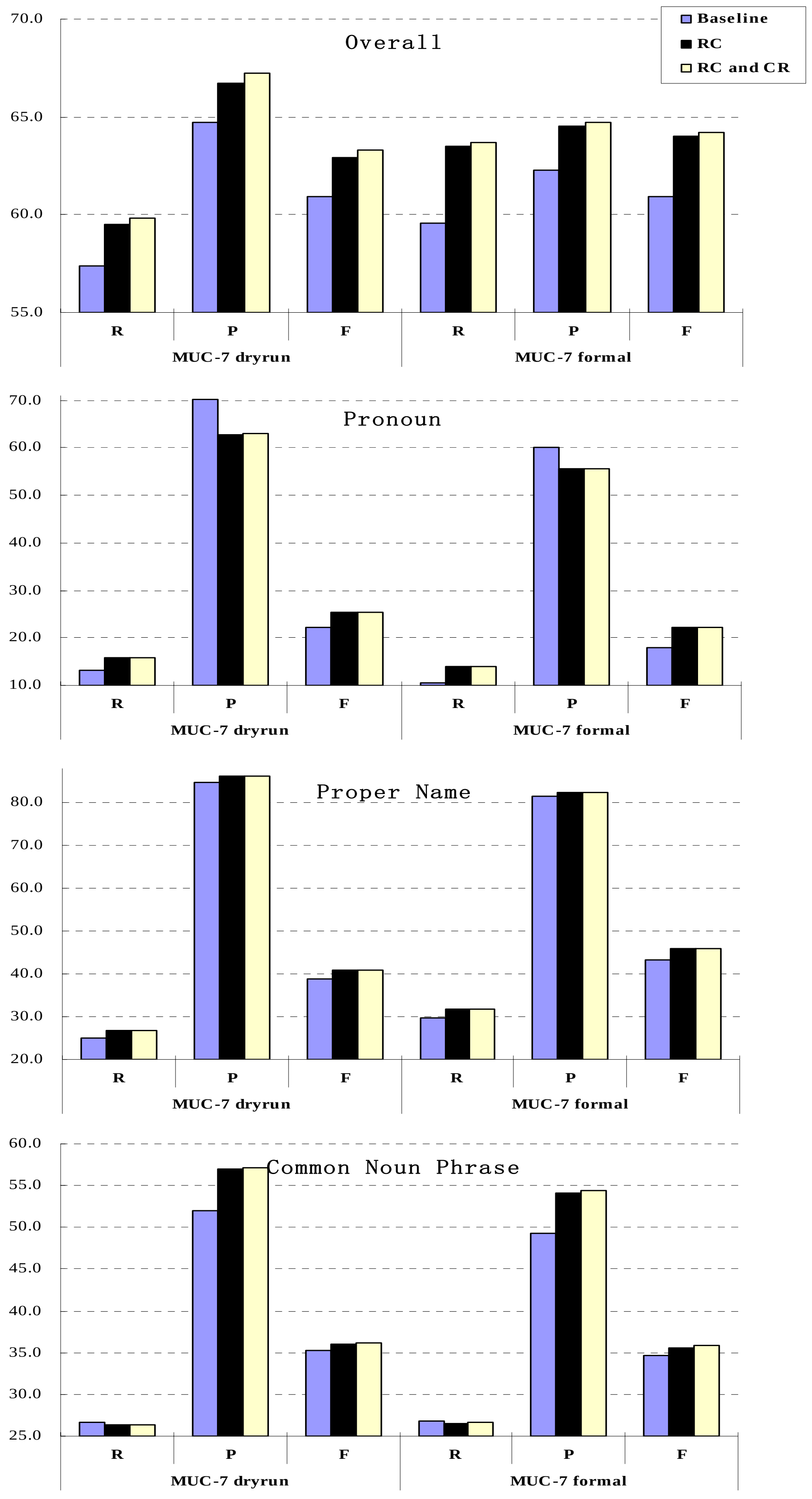


Figure 6.1

Results for the effects of ranked constraints and unsupervised conflict resolution on overall NP types, pronouns, proper names and common noun phrases.

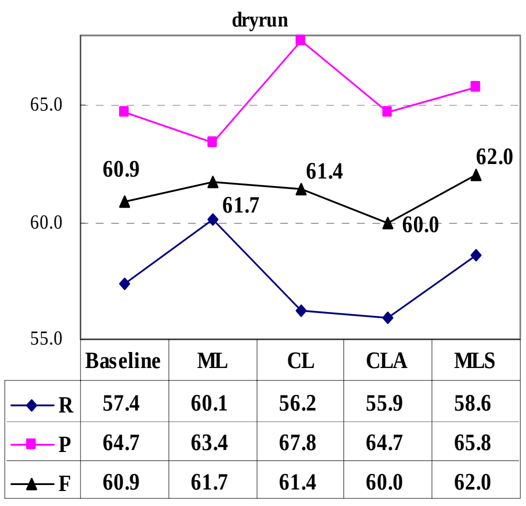
in order to evaluate the effects of ranked constraints and conflict resolution (CR), respectively, we make a coreference system which replaces CR with a simple conflict resolution. In the simple conflict resolution, coreference system gives up inserting a referring expression into a coreference tree if the expression is objected by some members of the tree. The system can evaluate the effect of ranked constraints without the influence of CR. Table 6.2 shows that both ranked constraints and CR have a positive effect on coreference system built through machine learning approach. And they improve recall without any loss in precision.

In the first chart of Figure 6.1, we see that the ranked constraints make a significant contribution to both recall and precision of baseline coreference system: recall increased with regard to baseline from 57.4% to 59.5% for 30 dryrun documents, and from 59.6% to 63.5% for 20 formal documents, respectively; precision increases 3% for dryrun and 2.2% for formal, respectively. As a result, F-measure increases from 60.9% to 62.9% for dryrun, and from 60.9% to 64.0% for formal. In contrast to the system including both ranked constraints and CR, the simple conflict resolution works not so well as our CR: 0.2% loss in both recall and precision for formal, 0.3% loss in recall with 0.5% loss in precision for dryrun, respectively. We see that after adding CR to our coreference system, F-measure increases about 0.4% and 0.2% for dryrun and formal, respectively.

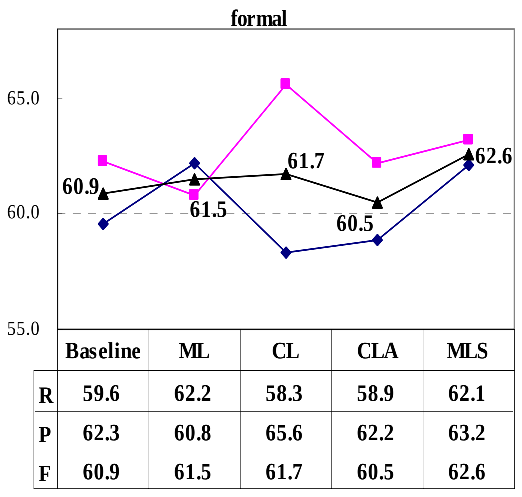
In an attempt to gain additional insight into the effects on different noun phrase types, we show the performances on pronouns, proper names and common nouns (Table 6.2). The last three charts of Figure 6.1 give us more intuitionistic knowledge of the effects of ranked constraints and our CR on different noun phrase types. After adding ranked constraints, except for the precision of pronoun and the recall of common noun phrase, the results of different noun phrase types indicate an improving trend. In particular, all F-measures increase along with the addition of ranked constraints and CR. As to the loss of the precision of pronoun after adding constraints to baseline, it is caused by must-link-to-something. And the loss of common noun phrase's recall after adding constraints to baseline is because of cannot-link-to-anything's effect. We will discuss about it in the error analysis.

6.2. The contribution of constraints

One factor that affects the performance of our system is the incorporation of ranked constraints. As we have explained above, there are four groups of constraints used in our system. It is interesting to find out the contribution of each group on coreference



	Baseline	ML	CL	CLA	MLS
◆ R	57.4	60.1	56.2	55.9	58.6
■ P	64.7	63.4	67.8	64.7	65.8
▲ F	60.9	61.7	61.4	60.0	62.0



	Baseline	ML	CL	CLA	MLS
R	59.6	62.2	58.3	58.9	62.1
P	62.3	60.8	65.6	62.2	63.2
F	60.9	61.5	61.7	60.5	62.6

Figure 6.2
Results of coreference systems to study the contribution of each constraints group

task. In order to evaluate it, we apply one group at each time. The results are shown in Figure 6.2.

6.2.1. Contribution of Each Constraints Group

In the figure, ML stands for must-link constraint group including four must-links as defined in Chapter 4. Cannot-link group, CL, includes all cannot-links defined in Chapter 4. CLA stands for cannot-link-to-anything and MLS means must-link-to-something. In Figure 6.2, we see that the recall lines of dryrun data and formal data have the similar figure. The precision lines of the two data sets are similar to each other also. As we know, the dryrun data and the formal data of MUC-7 [MUC-7, 1997] belong to the different knowledge domains. The dryrun data is a set of documents about aircraft accident. However, the formal data is a set of documents about launch event. Therefore based on documents with different knowledge domains, similar lines indicate some domain-independent characteristics of the four constraint groups. From the figure, we see that ML and MLS increase recall with regard to baseline, but with the loss of precision. In contrast to ML and MLS, CL and CLA have the capability to improve precision, but with the drop of recall. In particular, the CL's contribution to precision is outstanding comparing to other constraint groups. As a result, recall drops precipitously on both data sets. Similar to CL, ML's contribution to recall is significant among all constraints groups, but ML also makes the precision drops quickly. It is interesting to note that recall and precision are pairwise opposite. We are satisfied to see that three of four groups improve the F-measure with regard to the baseline system, especially MLS, which makes F-measure increase 1.1% and 1.7% for dryrun and formal, respectively.

6.2.2. Contribution of Each Combination of Constraints Group

To get more insight into the contribution of constraint groups on coreference task, we measure the overall performance of the coreference system with each combination of the four constraint groups. The results are shown in Table 6.3 and Figure 6.3. From Figure 6.3, we see that the recall lines and precision lines between dryrun data and formal data are also similar to each other. For both dryrun and formal data set, the combination of ML and MLS contributes maximum to recall among all the combination, and the combination of ML, CL and CLA contributes the most to precision. As expected, in comparison to all coreference systems with different combinations of four constraint groups, the combination including all constraint groups achieves the best F-measure of 63.3% and 64.2% for dryrun and formal data sets, respectively. The results prove that strategies employed to combine the available linguistic knowledge play an important role in machine learning approaches to coreference resolution.

Analysis of ML

Among the 16 system with different combinations of constraint groups, we compare the systems with ML to those without ML (See Figure 6.4). It is interesting to note that after adding ML, we see significant gains in recall and F-measure on each system.

No	ML	CL	CLA	MLS	MUC-7 dryrun			MUC-7 formal		
					R	P	F	R	P	F
1					57.4	64.7	60.9	59.6	62.3	60.9
2	√				60.1	63.4	61.7	62.2	60.8	61.5
3		√			56.2	67.8	61.4	58.3	65.6	61.7
4			√		55.9	64.7	60.0	58.9	62.2	60.5
5				√	58.6	65.8	62.0	62.1	63.2	62.6
6	√	√			58.6	65.8	62.0	62.1	63.2	62.6
7	√		√		58.0	68.1	62.6	61.0	66.1	63.4
8	√			√	**61.9**	63.6	62.7	**64.9**	61.2	63.0
9		√	√		54.4	67.9	60.4	57.6	65.6	61.4
10		√		√	58.6	63.4	60.9	61.6	60.7	61.2
11			√	√	58.5	66.0	62.0	61.1	63.8	62.4
12	√	√	√		57.3	**69.1**	62.6	60.8	**66.5**	63.5
13	√	√		√	61.2	64.4	62.8	**64.9**	61.2	63.0
14	√		√	√	60.5	66.3	63.2	63.7	64.2	64.0
15		√	√	√	57.2	66.1	61.3	60.4	63.9	62.1
16	√	√	√	√	59.8	67.2	**63.3**	63.7	64.7	**64.2**

Table 6.3
Results for each combination of four constraint groups, ML, CL, CLA and MLS

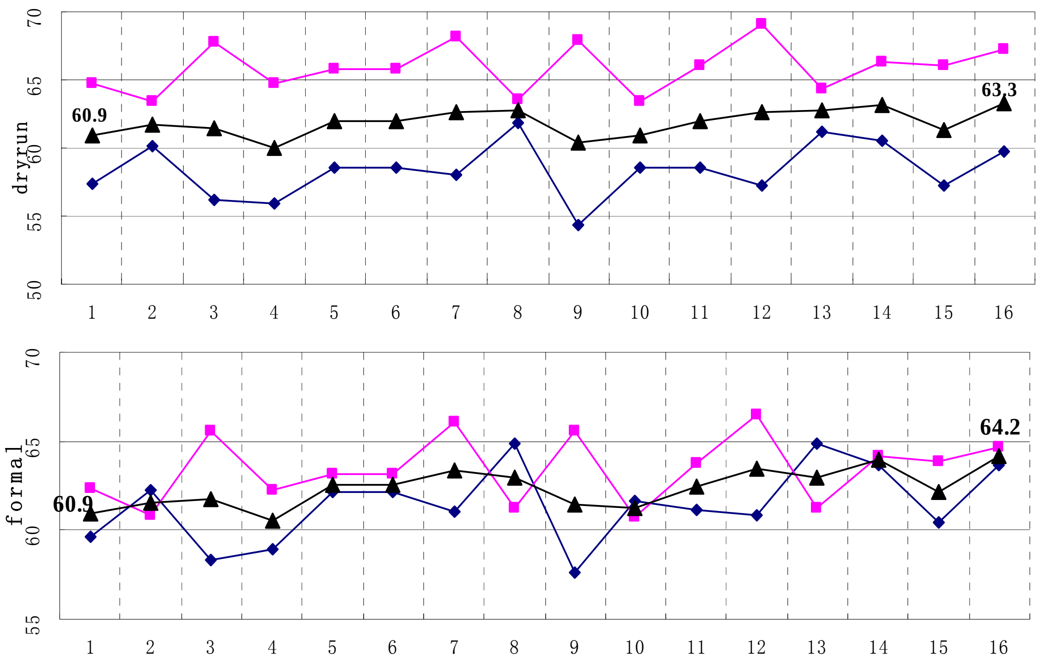


Figure 6.3:
Results for each combination of four constraint groups, ML, CL, CLA and MLS.

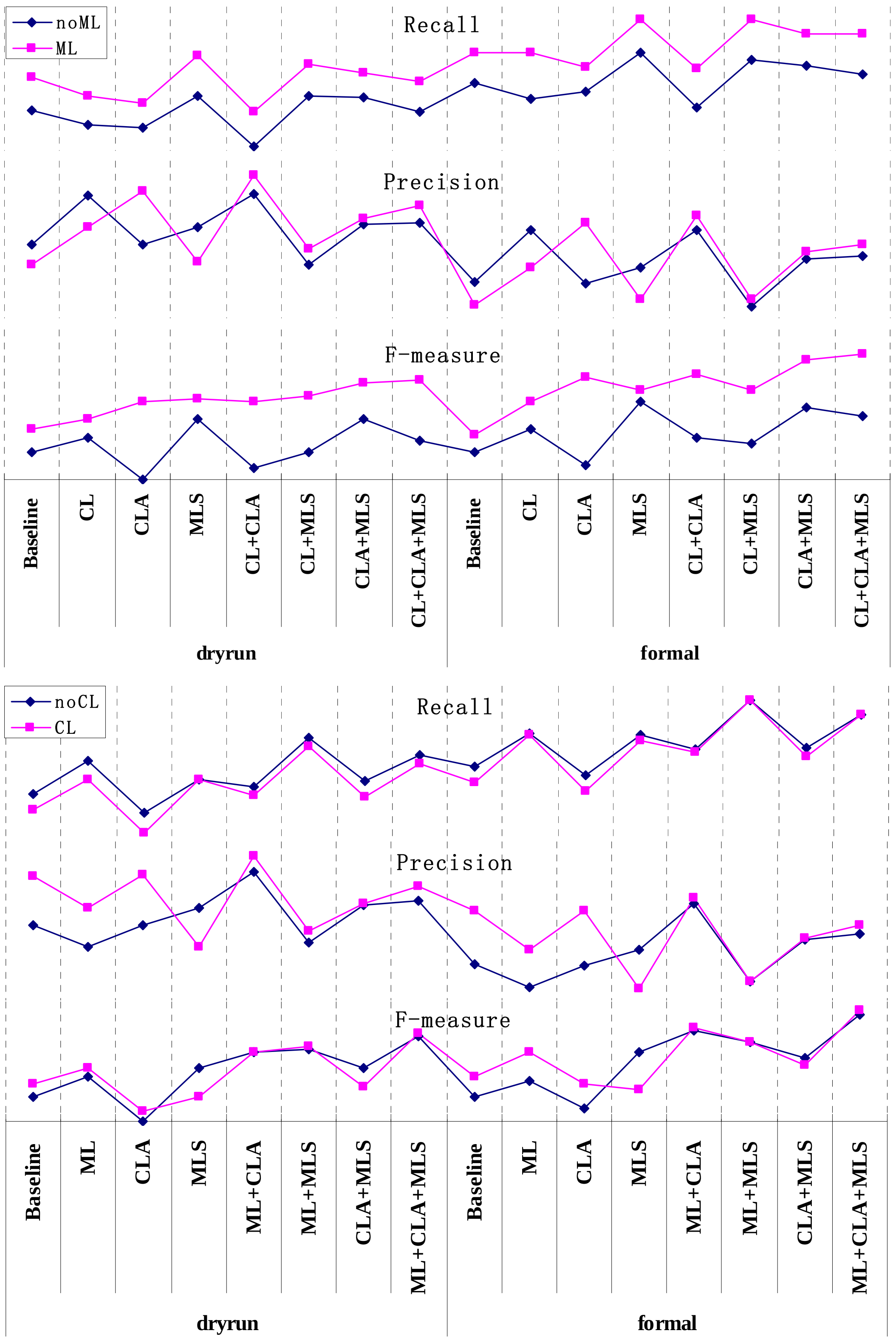
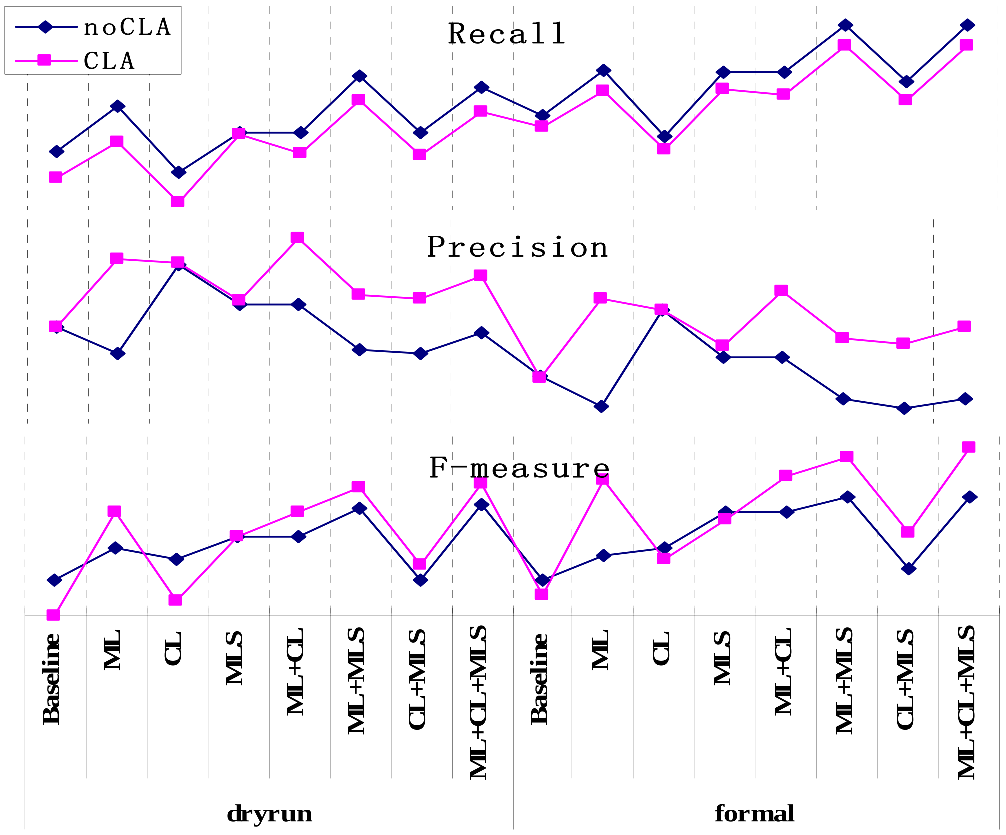


Figure 6.4
Results of coreference system with different combination of constraint groups to study the effect of ML and CL on performance of coreference system.



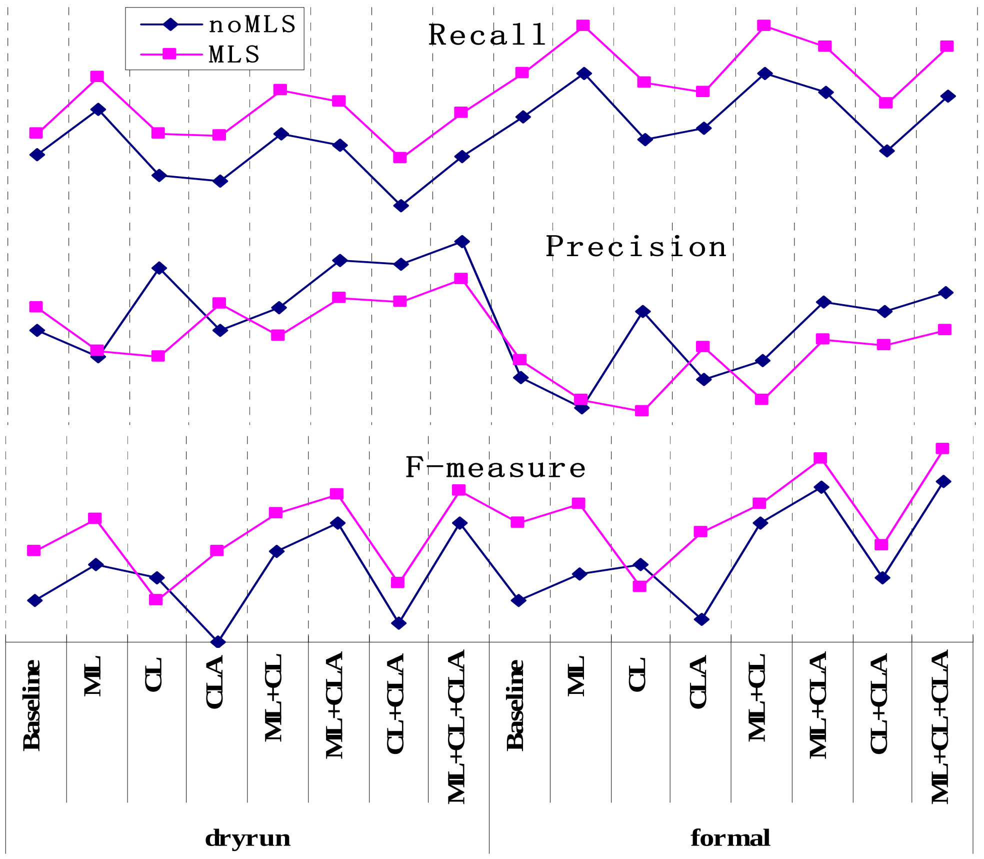


Figure 6.5

Results of coreference system with different combination of constraint groups to study the effect of CLA and MLS on performance of coreference system.

Our results provide direct evidence for the claim that our constraints can resolve the problems due to training data insufficient and "hard" training examples. And the experiment shows that ML is the most useful group to improve the coreference system's performance.

Analysis of MLS

MLS has the similar function to ML. After adding MLS, we observe reasonable increases in recall for both data sets in comparison to those systems without MLS. F-measure also increases except the system with only CL. Somewhat disappointingly, after adding MLS into it, F-measure drops for both data sets. It may be caused by strict cannot-link definition. MLS is the most approximate constraint group in our system. Its contribution is mainly to increase recall through adding pronouns into coreference trees, even if pronouns' antecedents are determined by error. Consequently MLS brings more conflicts into coreference trees. On the other hand, cannot-links detect such conflicts in coreference trees and choose a link to cut. Without must-links, each conflict must lead to a separating process. It influences the accuracy of conflict resolution. As a result, precision drops precipitously, which kills the increase of recall. Therefore F-measure drops too.

Analysis of CL

CL cannot improve those systems without ML but with MLS (See Figure 6.3 and Table 6.3) has. We have analyzed the combination of CL and MLS. For another system with the combination of CL, CLA and MLS, in contrast to the combination of CLA and

MLS, the F-measure drops 0.2%. Except the two systems, CL still contributes something to the performance of coreference system.

Analysis of CLA

If original system does not have any must-constraints group, such as must-links or must-link-to-something, CLA results in the worse performance, which is even worse than that of the baseline in F-measure (Comparing to baseline system, F-measure drops 0.9% and 0.5% on dryrun data and formal data, respectively. Comparing to the system with only CL, after adding CLA, F-measure drops 0.5% on dryrun data). Accordingly, its positive effect on coreference task is based on a reasonable recall. We see that the F-measure of the system without CLA is 62.8% and 63% for dryrun and formal data sets, respectively. With adding CLA, F-measure increases 0.5% and 1.2% for dryrun and formal, respectively.

As we can see, the four groups of constraints can be divided into two types: One is must-constraints and the other is cannot-constraints. Must-constraints improve recall with the loss of precision. And cannot-constraints improve precision with the loss of recall. Combination of them can achieve a balance between recall and precision. As a result, we can yield a satisfiactory F-measure.

6.2.3. Contribution of Each Constraint in ML and CL

In our system, ML includes 4 constraints and CL, 7. We add each must-link into the baseline system to see its contribution in isolate. As to cannot-link, we use the system

System	dryrun			formal		
	R	P	F	R	P	F
Baseline	57.4	64.7	60.9	59.6	62.3	60.9
Only RC_ML1	57.7	64.8	61.0	60.2	62.6	61.4
Only RC_ML2	57.9	64.8	61.2	60.4	62.2	61.3
Only RC_ML3	58.1	64.6	61.2	60.2	62.4	61.3
Only RC_ML4	58.2	64.9	61.3	60.2	62.5	61.3
ML+CLA+MLS+CR	**60.5**	**66.3**	**63.2**	**63.7**	**64.2**	**64.0**
Only RC_CL1	60.1	67.0	63.4	63.6	64.4	64.0
Only RC_CL2	60.3	66.3	63.1	63.7	64.4	64.0
Only RC_CL3	60.3	66.3	63.2	63.7	64.3	64.0
Only RC_CL4	60.2	66.1	63.0	63.8	64.3	64.0
Only RC_CL5	60.3	66.1	63.1	63.7	64.3	64.0
Only RC_CL6	60.3	66.5	63.3	63.7	64.4	64.1
Only RC_CL7	60.5	66.3	63.2	63.7	64.2	64.0
Our complete system	59.8	67.2	63.3	63.7	64.7	64.2

Table 6.4
Results of coreference system to study the effect of each constraint. Must-link constraint group is tested based on our duplicated Soon baseline system. And cannot-link constraint group is based on the system with ML, CLA and ML three constraint groups and conflict resolution

with ML, CLA and MLS constraint groups and conflict resolution as our baseline to test each cannot-link in isolate. The results are shown in Table 6.4.

Must-links

Table 6.4 shows that each must-link can contribute a little bit to the performance of coreference system. Among four must-links, RC_ML1 and RC_ML4 increase F-measure without any loss in either recall or precision. The results provide the evidence for the score determination in ranked constraints. RC_ML1 and RC_ML4 are the most reliable constraints. Therefore they are provided with the highest score. For

RC_ML2 and RC_ML3, RC_ML2 results in drop of 0.1% in precision for formal data and RC_ML3 results in drop of 0.1% in precision for dryrun data. However, they still improve F-measure in both data sets. Therefore the two must-links are provided with a little lower score than RC_ML1 and RC_ML4. Consider RC_ML2's contribution to common noun phrase coreference resolution, RC_ML2's score is set to be higher than that of RC_ML3. Table 6.4 also lists the result of the coreference system with the whole ML set based on duplicated Soon baseline system. The system with the whole set outperforms those systems with only one must-links on both data sets.

Cannot-link

For cannot-link, the contribution to F-measure of single cannot-link is not desirable in comparison to the contribution of complete cannot-links set. For dryrun data, only RC_CL1 and RC_CL6 improve F-measure with regard to the corresponding baseline, and RC_CL3 and RC_CL7 do not cause any loss in F-measure and precision. All the remaining cannot-links make F-measure drop. And RC_CL4 and RC_CL5 even cause drop in both recall and precision. For formal data, only RC_CL6 contribute 0.1% to F-measure. The other cannot-links maintain the baseline's performance. In our complete system, we use the whole CL set and achieve the best results comparing to those systems with only one cannot-links. We also evaluate the performance of the system with each combination of the seven cannot-links. Our results show that besides the whole set, some other sets of the 7 cannot-links also achieve the best result for a specific input. For dryrun, using RC_CL6 can get the best result, the combination of

RC_CL1 and RC_CL2 can also do it. For formal, the combination of RC_CL1, RC_CL2, RC_CL4 and RC_CL6 can get the F-measure of 64.2%. However, we use the whole CL set to ensure the constraints group to be general enough to suit different knowledge domains.

6.3. The contribution of conflict resolution

The contribution of conflict resolution is not as significant as that of ranked constraints. But it is interesting to note that our conflict resolution is an approach which can increase recall and precision simultaneously.

As we have explained above, conflict resolution is an approach based on cannot-links set and it improves performance of coreference system through rearrangement of current coreference trees. In comparison to simple conflict resolution, it usually would not cause the loss in recall. And after adjusting some links in a coreference tree, it improves precision and even recall. As a result, F-measure increases too. Our experimental results are shown in Table 6.2. We see that with regard to the system using simple conflict resolution, incorporating conflict resolution makes recall increase 0.3% and 0.2% for dryrun data and formal data, respectively. And the precision increases 0.5% and 0.2% for dryrun and formal. As a result, F-measure increases 0.4% and 0.2% for the two data sets, respectively. Furthermore, there is not any loss in recall or precision in pronoun, proper name and common noun phrase’s corresponding coreference resolution. It is a desirable result.

In an attempt to gain additional insight into the contribution of conflict resolution in our coreference system, we follow the processing of conflict resolution in dryrun data and formal data. We find that in the 50 documents of dryrun and formal data, separating subroutine of conflict resolution is called for 102 times in 26 documents and merging subroutine is called for 19 times, in 16 documents. 65 of 102 separating processing and 11 of 19 merging happen in dryrun. In comparison to formal, dryrun data encounters more conflicts than formal data. As a result, on dryrun data, the improvement (0.4% in F-measure) made by conflict resolution with regard to the simple conflict resolution is more than that of formal data (0.2% in F-measure). In order to evaluate the accuracy of conflict resolution, we track the 20 documents of formal. Significantly, all 37 separating processes choose the right links to cut. Among 8 merging processes, 7 are done correctly. In particular, 2 of the 7 merging processes employ separating processes. Such merging processes cut one of coreference trees at first and then combine the other coreference tree with one part just produced by cutting. It is more complex than a simple merging procession without cutting. Our results show that the conflict resolution can deal with such problem correctly without any supervised learning. For the only one wrong merging taking place in formal data, it is shown in the following example:

Sentence 6.1: ``Satellites give (us)$_{a1}$ an opportunity to increase the number of (customers)$_{b1}$ (we)$_{a2}$ are able to satisfy with the McDonald's brand," said McDonald's Chief Financial Officer, Jack Greenberg. ``It's a tool in our overall convenience strategy."

The merging between tree “a”(a1-a2) and tree “b”(the tree including b1) happens because there is no conflict between the two trees detected by cannot-links. Therefore, such error can be resolved introducing more elaborate cannot-links into the coreference system.

As we have mentioned above, if a conflict is detected, the conflict resolution is able to decide whether the conflict is true or false. If true, system calls separating subroutine to cut the tree. If false, system can ignore the conflict. In order to evaluate the capability of distinguishing true conflict and false conflict, we also search the conflicts which are skipped in formal data. There are 51 such conflicts found. And 7 of them happen in merging processes and the remaining happen in separating processes. We see that 45 of 51 conflicts which are determined as false correctly by conflict resolution. All error determinations belong to the separating processing. The main reason is information insufficiency. For example:

Sentence 6.2: The (National Association of Broadcasters)$_1$, which represents television and radio stations, has said the new satellite services would threaten local radio stations. (Broadcasters)$_2$ lobbied the FCC to delay issuing the license because of the threat of competition, Margolese said.

In the sentence, markable 2 is recognized as alias of markable 1 by error. Although they disagree in number, the conflict is skipped because must-link on alias has the preference to cannot-link on number disagreement. It is the error of alias determination which causes the failure of conflict resolution. And if the number information had higher accuracy, such conflict would not be skipped by error.

Another reason is that the rank of constraint also influences the accuracy of conflict resolution. For example:

Sentence 6.3: ``Since 1989-1990, there has not been another channel launched with this kind of immediate growth curve," said (Thomas S. Rogers)$_{a1}$, the (president)$_{a2}$ of (NBC Cable)$_{a3}$, a (member)$_{a4}$ of the executive committee in charge of the History Channel.

Sentence 6.4: ``Satellites give us an opportunity to increase the number of customers we are able to satisfy with the McDonald's brand," said (McDonald's Chief Financial Officer)$_{b1}$, (Jack Greenberg)$_{b2}$.

In Sentence 6.3, a3 and a4 satisfy the conditions of RC_ML2. Although a3 and a1 satisfy the conditions of RC_CL1, such conflict is skipped because RC_ML2 has higher score than RC_CL1. Unfortunately, devising a set of optimal score setting for general usage is impossible. Consider Sentence 6.4, b1 and b2 exactly form such example that RC_ML2 exceeds RC_CL1. In our system, we use a set of approximate optimal score setting for constraints. Such scores are determined based on human background knowledge. How to determine scores for constraints by machine is our future work.

As we can see, cannot-links have a significant effect on the accuracy of conflict resolution. And we find that for dryrun and formal, there are more than 50% documents which have not used conflict resolution at all. If we incorporate more cannot-links into the system, the conflict resolution will play a more important role on performance improvement. But it would also bring more difficulties in arranging the

Approach	Errors
NLP	Errors in head noun phrase extraction
	Errors in conjoint noun phrase identification
ML	Errors in Proper Name Identification
	Errors in Alias determination
	Errors in apposition determination
	indefinite proper name
MLS	non-anaphoric pronoun "it"
	Errors in antecedent determination of plural pronoun
CL	Using reliable features
	Language Exception
CLA	Number antecedent missing
CR	Conflict between constraints
	reliable features used in constraints
Baseline	Distant pronouns with same surface strings
	The same common noun phrases but they don't refer to anything

Table 6.5:
Errors in our complete system.

score of each constraint. Additional research on it is required in the future.

6.4. Error analysis

In [Soon et al., 2001], they have analyzed the errors made by their machine learning system. They classed errors into two groups: missing links (false negative) and spurious links (false positive). False negative causes recall errors and false positive causes precision errors. For missing links, they listed six types of errors caused by inadequacy of current surface features, errors in noun phrase identification, errors in semantic class determination, errors in part-of-speech assignment, errors in apposition determination and errors in tokenization. For spurious links, they also give out six types. They are caused by pronominal modifier string match, different entities with same strings, errors in noun phrase identification, errors in apposition determination and errors in alias determination. In this thesis, we focus our error analysis on those errors made by ranked constraints and conflict resolution. As another improvement made by head noun extraction and proper name identification, we also analyze the errors made by them. We randomly extract some formal documents from MUC-7 [MUC-7, 1997] and classes the errors according to different reasons. Breakdowns of such errors made by our new approach are shown in Table 6.5.

6.4.1. Errors Made by NLP

Our NLP pipeline simply takes the most right noun in a markable as head noun phrase. It leads to partially missing some compound noun phrases (including more than one token in head noun phrase). For example:

Sentence 6.5(1): When not focused on other nations' military bases, American spy satellites have been studying a dusty habitat of the humble (desert $(\text{tortoise})_{b1})_{a1}$ in an effort to help scientists preserve this threatened species.

Sentence 6.5(2): (Desert $(\text{tortoise})_{b2})_{a2}$ research is one of six environmental projects overseen by the CIA as part of a pilot program to use intelligence technology for ecological pursuits.

Compound noun phrase, “desert tortoise”, is separated into two parts by our nested noun phrase and head noun phrase extraction. Although our system can recognize the coreference pair (a1-a2), the link is replaced by (b1-b2) due to head noun phrase preference. But the link (b1-b2) is a spurious link to coreference system because they are not markables at all. As a result, (a1-a2) is missed.

Our NLP pipeline often misses conjoint noun phrases. It tends to recognize a conjoint noun phrase as two separated noun phrases. Such shortage leads to several errors. For example:

Sentence 6.6(1): $((\text{Ruth Ann Aldred})_{b1}$ and $(\text{Margaret Goodearl})_{c1})_{a1}$, both of who were once supervisors at a Hughes plant in California, accused the company of lying about the testing of components for missiles and fighter planes.

Sentence 6.6(2): Since their evidence resulted in the government recovering money, the False Claims Act law says $((\text{Aldred})_{b2}$ and $(\text{Goodearl})_{c2})_{a2}$ are due part of the fine.

According to MUC-7 [MUC-7, 1997] Coreference Task definition, “a1” and “a2” should be a markable without nested markables, respectively. Our NLP pipeline cannot recognize them. Instead, b1, b2, c1 and c2 are recognized by NP identification. As a result, (a1-a2) becomes a missing link. And two spurious links (b1-b2) and (c1-c2), appear.

6.4.2. Errors Made by ML

Obviously, must-link constraints mainly lead to spurious links. Some common noun phrases beginning with uppercase letter are often recognized as proper names by part-of-speech tagging. If such common noun phrases satisfy our RC_ML1, they will be tagged as coreferential pair with highest score. In a document’s title, such problem often appears. The errors in alias and apposition determination are similar to those explained in [Soon et al., 2001]. For example, in Sentence 6.3, “NBC Cable” and “member” are recognized as apposition, which results in series of problems. Alias determination is also difficult. For example, “American Airlines” and “American Eagle” are different entities. But they have the common part “American”. It results in the spurious link between them. Another errors made by must-links is “indefinite proper name”. In general, proper name should refer to a specific entity. But there are a lot of exceptions. Such as “American”, it not only can refer to one person born in America, but also can refer to a group of people living in U.S. Our must-link cannot distinguish such proper names which have the same surface strings, but have different referents.

6.4.3. Errors Made by MLS

MLS is similar to ML. It brings spurious links into system. Our results show that we can deal well with “he”, “she” and corresponding pronouns. The main errors are about “it” and plural pronouns. See Sentence 6.7:

Sentence 6.7: ``(It)'s been good for both companies,'' said Buddy Burns, Wal-Mart's manager of branded food service.

The “it” in the sentence does not refer to anything. It is non-anaphoric. Our system cannot determine the anaphoricity of “it”. As a result, some non-anaphoric “it” are forced to link some antecedents. Other frequent errors are about plural pronouns. As we have mentioned above, our NLP pipeline is not good at recognition of conjoint noun phrases. It is more difficult for a plural pronoun to search an antecedent. For example:

Sentence 6.8: (Wei Yen and Eric Carlson)$_{a1}$ are leaving to start (their)$_{a2}$ own Silicon Valley companies, sources said.

In the sentence, due to the miss of a1, a2 cannot be correctly linked to a1.

6.4.4. Errors Made by CL

As we have mentioned above, such errors are almost due to inaccurate information of number, semantic class and so on. For example, two “Monday” appear in a document. One of them is tagged as “DATE” but the other is “unknown”. As a result, they disagree in number (we take all “DATE”,”MONEY” and “PERCENTAGE” as “plural”). Fortunately, our conflict resolution skips such error. Other errors are due to the language exception. For example:

Sentence 6.9: And why not, since 75 percent of (McDonald's) diners decide to eat at (its) restaurants less than five minutes in advance? `` (They) want to be the first sign you see when you get hungry," said Dennis Lombardi, an analyst at Chicago-based market researcher Technomics Inc.

In the sentence, “McDonald’s”, “its” and “They” refer to the same entity. It is interesting to note that “it” and “they” can refer to each other although they disagree in number obviously.

6.4.5. Errors Made by CLA

Our CLA removes those figures which are not recognized as DATE, TIME, MONEY or PERCENTAGE. Such rule does not take into account the errors made by named entity recognition. For example, two “1992” appearing in the same document refer to the same year. But one “1992”’s semantic class is unknown. The “1992” is removed. Consequently, a link is missed by the error in CLA.

6.4.6. Errors Made by CR

The errors made by CR have been explained in last section. In conclusion, unsuitable score setting is the main reason which leads to errors in conflict resolution.

6.4.7. Errors Made by Baseline

There are two kinds of errors which have no relation to ranked constraint and conflict resolution. We class them as errors made by baseline system. The first error is about pronoun. It is that pronouns with the same surface string tend to link together. For example:

Sentence 6.10(1): ``Satellites give us an opportunity to increase the number of customers (we) are able to satisfy with the McDonald's brand," said McDonald's Chief Financial Officer, Jack Greenberg.

Sentence 6.10(2): ``When (we) come to Wal-Mart for diapers, we come here," said

Cook, 31, sitting at a table in the McDonald's inside the North Brunswick, New Jersey, store.

We see that two sentences are both speeches, but with different speakers. The two “we” should not refer to each other obviously. But due to “STR_MATCH”’s important role in coreference determination, they are linked together in our system.

Another error is also resulted from “STR_MATCH”. For example:

Sentence 6.11(1): But with no customers expected until 1998, the need for nearly $2 billion in (investment) and numerous competitors lurking in the shadows, Globalstar's prospects would not appear to be valuable to the average Lockheed shareholder.

Sentence 6.11(2): ``Any service that is based on satellites is going to be a fertile area for our (investment)," he said.

Although the two “investment” are over almost the whole document, they are recognized as coreference pair because of string match. It is a common phenomenon in our system. Common noun phrases coreference resolution is more difficult than that of proper name and pronoun. It needs more semantic information to see the inside relation between them. Simple string match cannot resolve the coreference problem of common NP. This problem is a remaining challenge for us.

7. Conclusion

7.1.1. Two Contributions

We investigate two methods to improve the coreference system built through machine learning approach. Based on the two methods, we increase F-measure of our baseline system from 60.9% to 64.2%.

Multi-level Ranked Constraints

First, we propose a set of linguistic-based, multi-level and ranked constraints which is compatible with supervised machine learning approach. We also make some changes in search algorithm. We use a multi-link clustering algorithm to replace the single-link clustering algorithm. With the set of constraints, the coreference system produces significant gains in both recall and precision and corresponding increases in F-measure. The set of constraints includes four kinds of constraints: must-link, must-link-to-something, cannot-link and cannot-link-to- anything. The first two constraints can be called must-constraints and the remaining two can be called cannot-constraints. Must-constraints improve recall, but at the cost of precision loss. Cannot-constraints behave in an opposite way. They improve precision with the loss of recall. The combination of must-constraints and cannot-constraint makes our system achieve the best result of 64.0% in F-measure, which is higher than that of baseline system about 3.1%. Our results show that the set of constraints resolves some problems in using machine learning for building coreference resolution systems, primarily the problem of having limited amounts of training data. The constraints also provide a bridge between coreference resolution methods built using linguistic knowledge and machine learning methods.

Conflict Resolution

We also propose conflict resolution for handling conflicting constraints within a set of corefering elements. In order to detect conflicts and remove conflicts in a coreference chain, first we use the data structure “coreference tree” to replace the “coreference chain”. Coreference tree retains the information of relation among referring expressions. For each referring expression in a coreference tree, we record the parent who introduced the expression into the coreference tree. Second, we use cannot-links to detect conflicts in a coreference tree. Lastly, after a conflict is detected, the resolution is to cut the separating link which has the lowest score. By using the tree structure, cannot-links and the separating link finding algorithm, the conflict resolution provides better performance compared to simple conflict resolution, which gives up inserting a link once a conflict is encountered. In contrast to the simple conflict resolution, our conflict resolution increases F-measure 0.2%. Furthermore, the conflict resolution is able to increase both recall and precision.

7.1.2. Future Work

The work of the thesis suggests some possible directions of future work.

There are still many ways to expand the constraints set. Up to now, our system includes 4 must-links, 7 cannot-links, 1 must-link-to-something and 1 cannot-link-to-anything. Adding more constraints into the four groups and introducing new types of constraints into the set of constraints are both promising directions.

As we have mentioned before, how to provide an optimal score for each constraints is a challenge for future research. In our system, the score is determined based on human knowledge and the score is approximately optimal. Making machine decide the rank of constraints is another task for future work.

In the error analysis, we see that common noun phrase coreference resolution still require improvement in our system. Common noun phrase coreference resolution requires more linguistic knowledge and semantic information. Up to now, our system only offers 12 features. And among them, only one indicates some semantic information. Expanding the feature set will not only help the common noun phrase coreference resolution, but also help us generate more useful constraints. Furthermore, it may be useful to employ more theoretical linguistic work, such as Focusing Theory [Grosz et al., 1977; Sidner, 1979], Centering Theory [Grosz et al., 1995] and the systemic theory [Halliday and Hasan, 1976].

Another aspect that requires improvement is the NLP pipeline. How to improve the accuracy of NLP pipeline requires further research for the state-of-the-art coreference resolution systems.

Appendix A : Name List

A.1 Man Name List

Aaron	Bevil	Elias	Gillam	Jeremy	Machutus
Abacuck	Blaise	Eliass	Godfrey	Jerman	Manasses
Abraham	Botolph	Eliza	Goughe	Jermanus	Mark
Adam	Brian	Elizeus	Gregory	Jerome	Marmaduke
Adlard	Cadwallader	Ellis	Griffin	Jervais	Martin
Adrian	Cesar	Ely	Griffith	Jesper	Mathew
Alan	Charles	Emanuel	Guy	Jesse	Matthew
Albert	Christian	Emery	Halius	John	Maurice
Alexander	Christopheer	Emmanuel	Hamond	Joice	Melchior
Allan	Christopher	Emmett	Hansse	Jonathan	Meredith
Alveredus	Chroferus	Enoch	Harman	Joos	Michael
Ambrose	Chroseus	Erasmus	Harmond	Joosus	Miles
Anchor	Ciriacus	Evan	Harry	Jordan	Mike
Andrew	Clement	Everard	Hector	Joseph	Morgan
Annanias	Conrad	Faustinus	Helegor	Joshua	Nathaniel
Anthony	Cornelius	Felix	Heneage	Josias	Newton
Archibald	Court	Ferdinand	Henry	Jossi	Nicholas
Archilai	Cuthbert	Frances	Hercules	Jucentius	Ninion
Arnold	Cutlake	Francis	Hieronimus	Julius	Noe
Arthur	Daniel	Fulk	Holland	Justin	Oliver
Augustin	David	Gabriel	Howel	Justinian	Osmund
Augustine	Denton	Garnett	Howell	Kenelm	Ottewell
Augustus	Didimus	Garret	Hugh	Kyle	Owen
Barnabas	Digory	Garrett	Humphrey	Lambert	Owin
Barnard	Dionisius	Gawen	Humphry	Lancelot	Paschall
Bartholomew	Drugo	Gawin	Ingram	Laurence	Pasco
Bartram	Dudley	Gentile	Isaac	Lawrence	Pasquere
Basil	Ebulus	Geoffrey	Isaacs	Leonard	Paul
Bellingham	Edi	George	James	Lewis	Peter
Benedict	Edmund	Gerrard	Jankin	Lionel	Philip
Benjamin	Edward	Gervase	Jasper	Lodowick	Phillip
Bennett	Edwin	Gilbert	Jeffery	Lucas	Pierce
Bertram	Eli	Giles	Jenkin	Ludwig	Polidore
Pompey	Rees	Rowland	Simon	Tobias	William
Prospero	Reginald	Ryan	Stephen	Tristram	Williams

Quivier	Richard	Salamon	Steven	Valentine	Wombell
Ralph	Robert	Sampson	Symon	Vincent	Wymond
Randall	Roger	Samuel	Thadeus	Walter	Zacharias
Randel	Roland	Sander	Theodosius	Warham	Zachary
Randolph	Roman	Sean	Thomas	Watkin	
Reece	Rook	Silvester	Timothy	Wilfred	

A.2 Woman Name List

Agnes	Dionise	Gartheride	Laura	Petronella
Alice	Dolora	Georgette	Lauren11	Phillipa
Amanda	Dorothea	Grace	Lettice	Prudence
Amie	Dorothy	Gwenhoivar	Luce	Rachel
Ann	Ebotte	Heather	Lucretia	Rawsone
Anna	Edith	Helen	Lucy	Rebecca
Annabella	Effemia	Helena	Mable	Rosanna
Anne	Eleanor	Hellen	Magdalen	Rose
Ashley	Elena	Isabel	Magdalena	Samantha13
Aveline	Elianora	Isabella	Magdalene	Sarah
Barbara	Elinor	Jane	Margaret	Sibil
Beatrice	Elizabeth	Janikin	Margareta	Sibill
Blanche	Ellen	Jennette	Margarete	Stephanie
Bridget	Ellena	Jennifer	Margarita	Susanna
Brittany	Ellois	Jessica	Margerie	Susannah
Cassandra	Ely	Joan	Margery	Susanne
Catherine	Emily	Joane	Maria	Suzanna
Cecily	Emma	Jocatta	Marian	Sybil
Charity	Etheldreda	Jocosa	Marion	Tabitha
Christiana	Ethelreda	Johanna	Martha	Thomasina
Christina	Ethelrede	Jone	Mary	Thomazine
Cicilia	Faith	Joyce	Matilda	Ursula
Constance	Florence	Judith	Megan	Venetia
Danielle	Frances	Juliana	Mildred	Winefred
Dionis	Francisca	Katherine	Nicole	Winifred

Appendix B: MUC-7 Sample

B.1 Sample MUC-7 Text

```
<DOC>
<DOCID> nyt960905.0652 </DOCID>
<STORYID cat=a pri=u> A6992 </STORYID>
<SLUG fv=taf-z> BC-TWA-CRASH-NYT </SLUG>
<DATE> &LR; </DATE>
<NWORDS> 09-05 </NWORDS>
<PREAMBLE>
BC-TWA-CRASH-NYT
ROUGH SEAS PARALYZE SEARCH FOR PLANE WRECKAGE
(sw)
By ANDREW C. REVKIN
c.1996 N.Y. Times News Service
</PREAMBLE>
<TEXT>
<p>
   SMITHTOWN, N.Y.   &MD;   On the 50th day after the crash of Trans World
Airlines Flight 800, senior investigators said that persistent
rough seas off the coast of Long Island had paralyzed efforts to
collect the remaining wreckage of the shattered jumbo jet.
<p>
   But some of the most coveted pieces of wreckage were still
missing, he said, including many parts of the center fuel tank,
which sat under a group of seats that many investigators say were
the likely center of the explosion.
</TEXT>
<TRAILER>
NYT-09-05-96 2017EDT
</TRAILER>
</DOC>
```

B.2 Sample MUC-7 Key

```
<DOC>
```

```
<DOCID> nyt960905.0652 </DOCID>
<STORYID cat=a pri=u> A6992 </STORYID>
<SLUG fv=taf-z> BC-<COREF ID="3" MIN="CRASH"><COREF
ID="1">TWA</COREF>-CRASH</COREF>-NYT </SLUG>
<DATE> &LR; </DATE>
<NWORDS> <COREF ID="79">09-05</COREF> </NWORDS>
<PREAMBLE>
BC-<COREF ID="2" TYPE="IDENT" REF="3" MIN="CRASH"><COREF ID="0"
TYPE="IDENT" REF="1">TWA</COREF>-CRASH</COREF>-NYT
<COREF ID="9" MIN="SEAS">ROUGH SEAS</COREF> PARALYZE <COREF
ID="11" MIN="SEARCH">SEARCH FOR <COREF ID="13"
MIN="WRECKAGE"><COREF ID="7">PLANE</COREF>
WRECKAGE</COREF></COREF>
(sw)
By ANDREW C. REVKIN
c.1996 N.Y. Times News Service
</PREAMBLE>
<TEXT>
<p>
    SMITHTOWN, N.Y. &MD; On the 50th day after <COREF ID="4"
TYPE="IDENT" REF="2" MIN="crash">the crash of <COREF ID="6"
TYPE="IDENT" REF="7" MIN="Flight 800"><COREF ID="5" TYPE="IDENT"
REF="0">Trans World
Airlines</COREF> Flight 800</COREF></COREF>, senior investigators said that
<COREF ID="8" TYPE="IDENT" REF="9" MIN="seas">persistent
rough seas</COREF> off the coast of Long Island had paralyzed <COREF ID="10"
TYPE="IDENT" REF="11" MIN="efforts">efforts to
collect <COREF ID="12" TYPE="IDENT" REF="13" MIN="wreckage">the
remaining wreckage of <COREF ID="14" TYPE="IDENT" REF="6" MIN="jet">the
shattered jumbo jet</COREF></COREF></COREF>.
<p>
    But some of the most coveted pieces of wreckage were still
missing, <COREF ID="68" TYPE="IDENT" REF="66">he</COREF> said, including
many parts of the center fuel tank,
which sat under a group of seats that many investigators say were
the likely center of <COREF ID="69" TYPE="IDENT" REF="55">the
explosion</COREF>.
</TEXT>
<TRAILER>
NYT-<COREF ID="78" TYPE="IDENT" REF="79">09-05-96</COREF> 2017EDT
</TRAILER>
</DOC>
```

Bibliography

[Amit and Baldwin, 1998] Bagga Amit and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the Seventh Message Understanding Conference(MUC-7).*

[Annie] Annie. http://www.aktors.org/technologies/annie/

[Aone and Bennett, 1995] Chinatsu Aone and Scott W. Bennett. 1995. Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceeding of the 33th Annual Meeting of the Association for Computational Linguistics*, Pages 122-129.

[Baldwin, 1995] Breck Baldwin. 1995. CogNiac: A discourse processing engine. *Ph.D. Thesis, University of Pennsylvania, Department of Computer and Information Sciences*.

[Cardie and Wagstaff, 1999] Clarie Cardie and Kiri Wagstaff. 1999. Noun phrase coreference as clustering. In Proceedings of the 1999 Joint SIGDAT Coreference on Empirical Methods in Natural Language Processing and Very Large Corpora , 82-89, Association for Computational Linguistics, 1999.

[Charniak, 1972] Charniak, Eugene. 1972. Towards a model of children's story comprehension. *AI-TR 266, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1972*.

[Cohen, 1995] W. Cohen. 1995. Fast Effective Rule Induction. In Proceedings of the Twelfth International Conference on Machine Learning.

[Deemter and Kibble, 2000] Kees van Deemter and Rodger Kibble. 2000. On Coreferring: Coreference in MUC and related annotation schemes. *Computational Linguistics, 26(4).*

[Grosz et al., 1977] B. J. Grosz. The representation and use of focus in a system for understanding dialogs. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence, pages 67-76, 1977*.

[Grosz et al., 1995] B. J. Grosz, A. K. Joshi, and S. Weinstein. 1995. Centering: a framework for modeling the local coreference of discourse. *Computational Linguistics, 21(2):203-226*.

[Grover et al., 2000] Claire Grover, Colin Matheson, Andrei Mikheev, and Marc Moens. 2000. LT TTT - A Flexible Tokenization Tool. In Second International Conference on Language Resources and Evaluation, LREC'00, 2000. http://www. ltg.ed.ac.uk/software/ ttt/.

[Halliday and Hasan, 1976] M. Halliday and R. Hasan. 1976. Cohesion in English. *Longman*.

[Iida et al., 2003] Ryu Iida, Kentaro Inui, Hiroya Takamura and Yuji Matsumoto. 2003. Incorporating Contextual Cues in Trainable Models for Coreference Resolution. *EACL Workshop "The Computational Treatment of Anaphora", 2003*.

[LT CHUNK, 1997] LT CHUNK. 1997. *http://www.ltg.ed.ac.uk/software/chunk/* index.html.

[LTG] LTG Software. *http://www.ltg.ed.ac.uk/software*.

[McCarthy, 1996] Joseph F. McCarthy. 1996. A trainable approach to coreference resolution for Information Extraction. Ph.D. thesis. University of Massachusetts.

[Miller, 1990] George A. Miller. 1990. WordNet: An on-line lexical database. International Journal of Lexicography, 3(4):235-312.

[Mitkov, 1997] Ruslan Mitkov. 1997. Factors in anaphora resolution: they are not the only things that matter. A case study based on two different approaches. In *Proceedings of the ACL'97/EACL'97 Workshop on Operational Factors in Practical, Robust Anaphora Resolution.*

[MUC-6, 1995] MUC-6. 1995. Coreference task definition (v2.3, 8 Sep 95). In

Proceedings of the Sixth Message Understanding Conference (MUC-6), pages 335-344.

[MUC-7, 1997] MUC-7. 1997. Coreference task definition (v3.0, 13 Jul 97). In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*.

[Ng and Cardie, 2002a] Vincent Ng and Claire Cardie. 2002a. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Pages 104-111.

[Ng and Cardie, 2002b] Vincent Ng and Claire Cardie. 2002b. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proceedings of 19th International Conference on Computational Linguistics(COLING-2002).*

[Ng and Cardie, 2002] Vincent Ng and Claire Cardie. 2002. Combining sample selection and error-driven pruning for machine learning of coreference rules. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-02)*, pp. 55-62 Philadelphia. PA, July, 2002.

[Ng and Cardie, 2003] Vincent Ng and Claire Cardie. 2003. Weakly Supervised Natural Language LearningWithout Redundant Views. Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003), Association for Computational Linguistics, 2003.

[Ng and Cardie, 2003] Vincent Ng and Claire Cardie. 2003. Bootstrapping Coreference Classifiers with Multiple Machine Learning Algorithms. *In Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP-2003), Association for Computational Linguistics, 2003.*

[Quinlan, 1993] Quinlan, John Ross. 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco, CA.

[Siddharthan, 2003] Advaith Siddharthan. 2003. Resolving Pronouns Robustly: Plumbing the Depths of Shallowness. In Proceedings of the Workshop on

Computational Treatments of Anaphora, 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003).

[Sidner, 1979] Candace L. Sidner. 1979. Towards a computational theory of definite anaphora comprehension in English discourse. TR 537, M.I.T. Artificial Intelligence Laboratory, 1979.

[Soon et al., 2001] Wee Meng Soon, Hwee Tou Ng and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics, 27(4),* Page 507-520.

[Vilain, 1995] M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the Sixth Message Understanding Conference (MUC-6),* pages 45-52, San Francisco, CA. Morgan Kaufmann.

[Wagstaff, 2002] Kiri Wagstaff. 2002. Intelligent Clustering with Instance-Level Constraints. *Ph.D. Dissertation*.

[Wagstaff and Cardie, 2000] Kiri Wagstaff and Claire Cardie. 2000. Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML2000)*, p. 1103-1110.

[Yang et al., 2003] Xiaofeng Yang, Guodong Zhou, Jian Su and Chew Lim Tan. 2003. Coreference Resolution Using Competition Learning Approach. In *Proceedings of 41th Annual Meeting of the Association for Computational Linguistics (ACL03),* Pages176-183.