CHAPTER 5
APPLICATION 2: NOUN PHRASE COREFERENCE RESOLUTION

In Chapter 4, we saw how COP-KMEANS performed much better than the regular k-means algorithm on the problem of automatic map refinement. In this chapter, we explore the application of COP-COBWEB to a different real-world problem, that of noun phrase coreference resolution. The major result of this chapter is a demonstration that COP-COBWEB is able to effectively apply constraints to improve its performance on the coreference problem.

In Section 5.1, we define this problem and justify our approach to it. Next, we describe COBWEB's default performance on this problem and discuss how appropriate linguistic constraints can be generated (Section 5.2). Evaluation in this domain is particularly challenging, so we devote some time to a thorough discussion of it in Section 5.3. Finally, we present experimental results on news articles (Section 5.4) and compare COP-COBWEB's performance to that of other systems (Section 5.5).

5.1 Noun Phrase Coreference

The field of Natural Language Processing (NLP) is concerned with developing algorithms to process text written in human (natural) languages. There are many steps involved in "processing" (i.e., understanding) a text, including *parsing* the input text to discover the syntactic structure of the sentences, performing *word sense disambiguation* to match words with their contextual definitions, and higher-level analyses to discover deeper semantic meanings.

One of the steps in text processing is referred to as *noun phrase coreference resolution*. Noun phrases include proper names (e.g., "Frodo Baggins"), common nouns ("a hobbit"), and pronouns ("he"). The purpose of this step is to identify pairs of *coreferent* noun phrases, i.e., those that refer to the same entity. We perform this task automatically each time we encounter a pronoun in written or spoken language: we deduce which other noun phrase(s) it refers to. For example, "it" in the previous sentence refers to "a pronoun." You, as a human, probably deduced that without any conscious effort. Although humans are generally able to perform this resolution step with ease, it presents a formidable challenge to automated NLP systems. Noun phrase coreference is a complex linguistic phenomenon, and successful resolution relies on cues from syntactic structure, semantic constraints, and shared world experience on the part of the author and the reader.

As a concrete example, an excerpt from a news article about space exploration is shown in Figure 5.1 with each noun phrase enclosed in square brackets. Any reader of this text, whether a human or a computer, must be able to determine the coreference relationships to fully understand what the text means. For example, the fact that "Mars" and "the Red Planet" refer to the same entity is a crucial component in interpreting the passage. The figure also visually combines coreferent noun phrases by placing them into three separate groups. This document contains

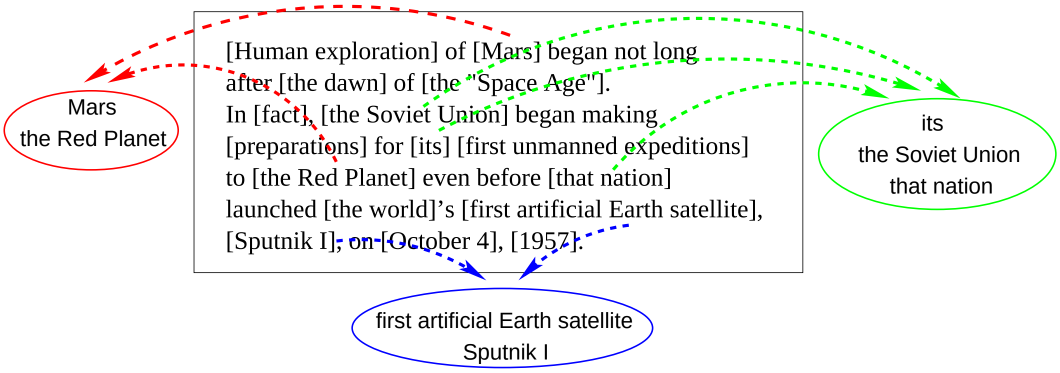

Figure 5.1: An excerpt from a news article with noun phrases bracketed and coreference classes indicated

a total of 12 coreference groups; the nine remaining bracketed noun phrases do not participate in a coreference relationship and are considered singletons.

5.1.1 A Clustering Approach

Coreference is a binary relation over noun phrases. It is an equivalence relation that operates on pairs of noun phrases. Consequently, the task of identifying coreference classes can be naturally phrased as a partitioning, or clustering, problem. With a suitable set of features to describe each noun phrase and a method for measuring the distance between each pair of noun phrases, we can apply any standard clustering algorithm to this task. Each of the resulting clusters corresponds to what we referred to earlier as a "group."

In previous work on this problem, we developed an approach to this problem that uses a clustering algorithm with a highly specialized, hand-tuned distance measure that incorporates constraint information (Cardie and Wagstaff, 1999). However, this algorithm is specific to the problem of noun phrase coreference and cannot be used for other tasks. In contrast, we will present the results of using the multi-purpose COP-COBWEB algorithm, which takes in the data set of noun phrases separately from the set of constraints.

In this section, we will describe in more detail how noun phrase coreference resolution can be cast in terms of a partitioning problem. We will also discuss the features we use to represent each noun phrase. In addition, we will justify our choice of COP-COBWEB for this problem. Finally, we will show an example of the output produced by the system in the absence of constraints.

5.1.2 Coreference Resolution as Partitioning

As discussed above, we can view the coreference resolution problem as a partitioning task. More precisely, the data set $D = \{NP_i\}$ consists of all of the noun phrases in an input text. We have selected six features to represent each noun phrase: $NP_i = < F_1, \ldots, F_6 >$. We seek a partition $P = \{C_1, \ldots, C_k\}$ such that

Table 5.1: Features used to represent noun phrases for coreference resolution

F_i	**Feature**	**Description**
1	POSITION	Position in the document (sequential)
2	HEAD NOUN	Rightmost noun in the noun phrase
3	SEMANTIC CLASS	General class from WordNet
4	NUMBER	singular or plural
5	GENDER	masculine, feminine, either, or neither
6	ANIMACY	animate, inanimate, or neither

each cluster C_j represents an equivalence class of noun phrases with respect to coreference.

Generation of noun phrase features. To create the input data set for a given document, we used the Empire noun phrase finder (Cardie and Pierce, 1998) to locate all noun phrases in the text. Note that Empire identifies only *base noun phrases*, i.e., simple noun phrases that contain no other smaller noun phrases within them. For example, "Human exploration of Mars" is too complex to be a base noun phrase. It contains two base noun phrases, "Human exploration" and "Mars," connected by a preposition.

Each noun phrase in the input text is represented by a set of six features, which are listed in Table 5.1. The choice of features was inspired by heuristics proposed by research on coreference in the field of computational linguistics (Mitkov, 1999). All feature values are automatically generated by simple heuristics and, therefore, are not always perfect. Because Empire is a partial parser, we do not have access to the full parse structure of the document. This additional information could potentially increase the accuracy of the noun phrase feature values, or allow the use of additional features, such as the grammatical *role* of the noun phrase in the sentence (e.g., subject or object).

As an example, the values for feature 4 (NUMBER) were determined by checking whether the noun phrase ended with an 's'. If so, it was labeled plural; otherwise, it was singular. Feature 3 (SEMANTIC CLASS) relies on the availability of a semantic network. We used WordNet (Fellbaum, 1998) to classify the head noun into one of several general categories: animal, city, human, time, object. If none of these applied, we used the noun phrase's immediate parent in WordNet's class hierarchy. Separate heuristics were used to identify a noun phrase as a number, money, or a company. If the system was unable to assign a specific feature value, the noun phrase under consideration received a value of unknown for that feature. Further details on our feature generation methods for this problem are discussed by Cardie and Wagstaff (1999).

Selection of COP-COBWEB. We selected the COP-COBWEB algorithm for this problem for two main reasons. First, COP-COBWEB automatically determines the number of clusters in the data set. Second, its incremental, order-sensitive nature is a good fit to our model of how humans process documents. Since documents are presumably written to facilitate human processing, we can expect that COP-COBWEB will benefit from this structure.

In contrast to our experiments with automatic map refinement, there is no easy way to estimate a good value for k, the number of noun phrase classes present in a document. As explained in Chapter 4, when clustering to find lanes in GPS data, it was reasonable to try a small set of different k values and select the best resulting partition. This works because roads rarely have more than four or five distinct lanes. However, with noun phrase coreference, the number of classes could vary from 1 to n, the number of noun phrases in the document. Documents may contain hundreds or even thousands of noun phrases. Therefore, COP-COBWEB, which automatically determines the number of clusters present, is a better choice for this problem.

Most documents (in particular, news articles) are written with the assumption that a human will be reading them. Humans generally read sequentially, from the beginning to the end of the document. We therefore process the text in an incremental fashion, continually building and refining our mental comprehension of the document as we progress. This means that our interpretation of what we encounter at any given point in the document is influenced by what we have already seen (but not by the remainder of the document).

COBWEB was originally developed based on observations about human conceptual processing (Fisher, 1987). It was intentionally designed to be incremental (and therefore sensitive to the order of the input observations) since that ability is essential in real-world situations. If a learner is limited to batch processing, then it cannot form abstractions or draw conclusions about its observations until it is entirely done processing them. Our modified version, COP-COBWEB, retains this incremental quality. With respect to the problem of text processing, incremental processing is very important. Because humans are the expected readers, documents written *by* humans are likely structured and optimized for an incremental reader. Our choice of COP-COBWEB exploits this inherent structure to enable better processing of the noun phrases in a document.

5.1.3 COBWEB Performance on Coreference Resolution

Let us return to the example text presented in Figure 5.1. Standard (unconstrained) clustering of the data using COBWEB produces the output shown in Figure 5.2 (for clarity, the arrows from noun phrases to the largest cluster have been omitted). In this case, COBWEB identified a total of three coreference clusters. We can see that the algorithm is overly generous in linking noun phrases together; the correct answer (according to Figure 5.1) contains 12 clusters. COBWEB is generally biased towards the creation of fewer clusters, since it seeks a

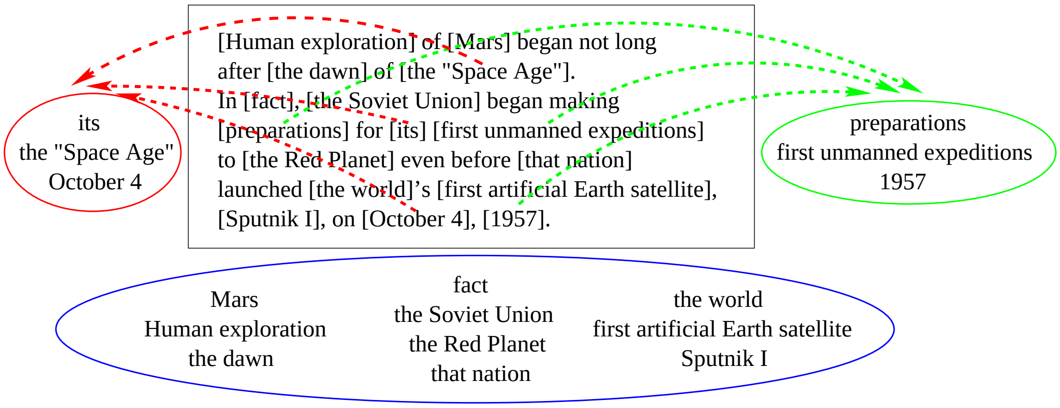

Figure 5.2: Coreference output for the text in Figure 5.1 when clustering without constraint information. Arrows from noun phrases to the largest cluster have been omitted.

partition of the data of minimal complexity. After discussing the kind of constraints that are useful for this problem, we will show the output of COP-COBWEB, using constraints, on this example in Section 5.2.1.

5.2 Generating Coreference Constraints

Once again, we make use of the constraint types defined in Chapter 3 to express our domain knowledge about noun phrase coreference. This problem has been studied extensively by linguists and has led to several heuristic rules that describe how coreference resolution can be done.

Table 5.2 lists the ten heuristics, HC_1 to HC_9 and HM_1, that we used to encode domain knowledge about coreference in the form of instance-level constraints.[1] These heuristics, like the feature set, were inspired by work in computational linguistics and by our own previous experience with noun phrase coreference (Cardie and Wagstaff, 1999). Each heuristic in the table takes as input two noun phrases, NP_i and NP_j, and generates either a cannot-link constraint (for HC_h) or a must-link constraint (for HM_1) if the heuristic's conditions are satisfied. Some of the heuristics compare feature values for NP_i and NP_j, while others use additional non-clustering features. A *non-clustering feature* is a feature that expresses useful information but is not used during the clustering process. In this case, the non-clustering features are useful for the constraint generation phase. We use three main kinds of heuristics to generate cannot-link constraints: *compatibility*, *article*, and *semantic class* heuristics.

[1]For simplicity, we here omit checks for `unknown` values in attributes; the heuristics apply only to feature values that are known. Also, we abbreviate ANIMACY to ANIM and SEMANTIC CLASS to SEMCL.

Table 5.2: Heuristics for generating constraints between NP_i and NP_j for coreference resolution

Compatibility		**Article**	
HC_1	NP_i.NUMBER $\neq$ NP_j.NUMBER	HC_5	$i < j$ and NP_j.ARTICLE $=$ indefinite and NP_i.PROPER $=$ no and NP_i.PRONOUN $=$ none
HC_2	NP_i.ANIM $\neq$ NP_j.ANIM	HC_6	$i < j$ and NP_i.ARTICLE $=$ none and NP_j.ARTICLE $=$ definite and NP_i.PROPER $=$ no and NP_i.PRONOUN $=$ none and NP_i.SEMCL $\neq$ NP_j.SEMCL
HC_3	NP_i.GENDER $\neq$ either and NP_j.GENDER $\neq$ either and NP_i.GENDER $\neq$ NP_j.GENDER	HC_7	$i < j$ and NP_i.ARTICLE $\neq$ none and NP_j.ARTICLE $=$ none and NP_j.PROPER $=$ no and NP_j.PRONOUN $=$ no
HC_4	NP_i.GENDER $=$ either and NP_j.GENDER $=$ neither		
Semantic class		**Appositive**	
HC_8	NP_i.SEMCL $\neq$ NP_j.SEMCL and (NP_i.SEMCL $=$ company or NP_i.SEMCL $=$ human)	HM_1	NP_j.APPOSITIVE $=$ yes and $i = j - 1$
HC_9	NP_i.SEMCL $\neq$ NP_j.SEMCL and $NP_{i,j}$.SEMCL $\neq$ company and $NP_{i,j}$.SEMCL $\neq$ human		

Compatibility constraints. Heuristics HC_1 through HC_4 check for compatibility between NP_i and NP_j. A mismatch on NUMBER or ANIMACY is an immediate indication that a cannot-link constraint should be created between the two noun phrases. The heuristics for handling the GENDER feature are a little more complicated, since a value of either can match any of masculine, feminine, or either.

Article constraints. Heuristics HC_5 through HC_7 encode rules that examine the articles used in NP_i and NP_j. The ARTICLE present in a noun phrase (definite, indefinite, or none) is a non-clustering feature, because it cannot be profitably used in clustering. We do not want the similarity of two noun phrases to increase simply because they match on this feature; the fact that two noun phrases both have indefinite articles does not make them more likely to be coreferent. Two other non-clustering features are used by these heuristics: PRONOUN (nominative, accusative, possessive, demonstrative, ambiguous, or none) and PROPER NAME (yes or no). The constraints generated by these heuristics are truly complementary to the data set D, since they are based on information not available in D.

We will use the following two sentences to illustrate the article heuristics.

(1) [The cat] climbed [my tree].
(2) [A cat] crossed [the road].

HC_5 asserts that a noun phrase with an indefinite article cannot link backwards to a noun phrase that is not a proper name or pronoun. In general, an indefinite article signals the start of a new coreference class; "A cat" in sentence 2 cannot link to "The cat" in sentence 1. HC_6 claims that a noun phrase with a definite article cannot link backwards to a noun phrase with no article, unless it is a proper name or a pronoun or the head nouns match. This is because noun phrases with definite articles tend to link to other definite or indefinite noun phrases. For example, "the road" cannot link to "my tree." Finally, HC_7 claims that a noun phrase with no article cannot link backwards to a noun phrase with an article, unless it is a proper name or pronoun. Once a noun phrase has been introduced with an article, it is very unusual to refer to it without an article. For example, after encountering sentence 1, we would not expect to later see just "cat" or any non-pronominal noun phrase referring to the same cat (unless it were a proper name).

Semantic class constraints. Heuristics HC_8 and HC_9 provide special handling for noun phrases classified as human or company. A mismatch on semantic class usually indicates that NP_i and NP_j cannot be coreferent. However, because many proper names are not present in WordNet, we allow humans and companies to potentially link to noun phrases of unknown classification.

Appositive constraint. Heuristic HM_1 indicates that an appositive noun phrase must be linked to its immediate predecessor in the document. An *appositive noun phrase* is a noun phrase that explains or further defines the preceding noun phrase. In English, appositive noun phrases are usually enclosed by commas or parentheses. For example, in the sentence "I spoke with Mr. Brown, my landlord, yesterday," "my landlord" is an appositive noun phrase that refers to "Mr. Brown." In such cases, we generate a must-link constraint between the two noun phrases. Note that it is possible for the output of this heuristic to conflict with some of the HC_h heuristics. For example, in the sentence "My teacher, a famous biologist, gave an excellent talk today," "a famous biologist" is an appositive noun phrase that refers to "my teacher." However, HC_5 will claim that the two noun phrases cannot be linked, because "My teacher" is not a pronoun or a proper noun, and "a famous biologist" begins with an indefinite article. Consequently, we permit HM_1 to override all of the HC_h heuristics, because it is an extremely reliable heuristic. We will later present a comparative evaluation of the accuracy of each heuristic.

Constraint generation. We have defined ten constraint heuristics. To generate the constraint sets, $Con_=$ and $Con_{\neq}$, we apply each heuristic to every pair of noun phrases in a given document. Each application of a heuristic to a pair of noun phrases may result in a must-link constraint, a cannot-link constraint, or no constraint (if the heuristic's conditions are not met). We take the transitive closure of the resulting set of constraints ($Con_= \bigcup Con_{\neq}$), as explained in Section 3.1.1, and provide the closed versions ($Con'_=$ and $Con'_{\neq}$) to COP-COBWEB.

Other constraint heuristics are possible. It is important to note, however, that this domain knowledge is being represented by hard constraints. Language is infamous for its exceptions to rules, and these heuristic rules are no exception. If a rule is violated by the actual text, but encoded as a hard constraint, then the clustering algorithm is forced to make an incorrect decision. We will analyze the accuracy of the constraint sets these heuristics create by comparing their decisions with the true (human-generated) coreference annotations. We will see that good performance on the coreference resolution problem can be obtained, even with the inherent restriction imposed by expressing knowledge as hard constraints. However, we expect that performance will be even higher, and that a larger range of domain knowledge can be encoded, if we instead make use of soft constraints to represent this knowledge. Soft constraints will be investigated more fully in Chapter 6.

5.2.1 Constraints on a Real Example

Let us once again return to the example text presented in Figure 5.1. Using the heuristics described in the previous section, and taking the transitive closure of the resulting constraints, we obtain a set of 111 pairwise constraints. Five of these constraints are presented in Table 5.3.

These constraints allow COP-COBWEB to avoid the mistakes that it makes without access to the domain knowledge encoded in the constraints. If we run COP-COBWEB with the indicated set of constraints, we obtain the output partition

Table 5.3: Sample constraints generated on the text in Figure 5.1

Constraint	**Heuristic**
"first unmanned expedition" cannot link to "preparations"	HC_1
"the dawn" cannot link to "preparations"	HC_7
"the dawn" cannot link to "the Red Planet"	HC_9
"the Soviet Union" cannot link to "Human exploration"	HC_6, HC_9
"Sputnik I" must link to "first artificial Earth satellite"	HM_1
...	...

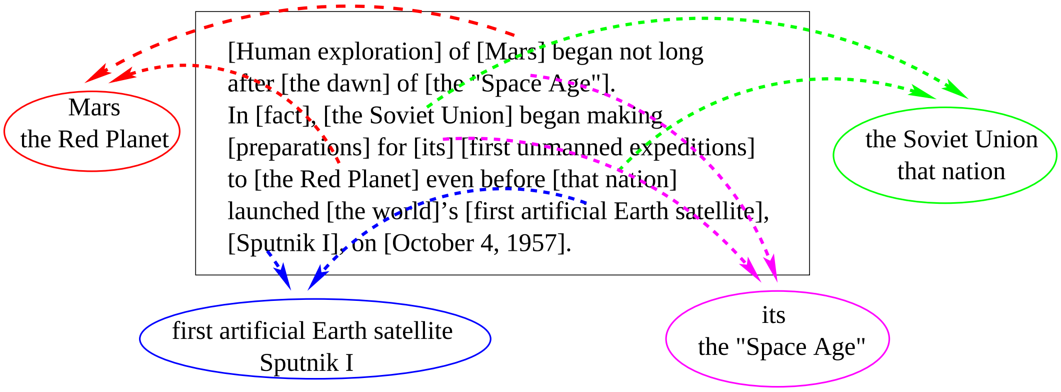

Figure 5.3: Coreference output for the text in Figure 5.1 when clustering with constraints

shown in Figure 5.3 (for clarity, singleton classes are not shown). The system identified 12 coreference clusters. The only mistake it made was in deciding which noun phrase "its" refers to. Pronoun resolution is a notoriously difficult sub-problem in this task, especially for "it" (e.g., Dagan and Itai (1991) and others have done work exclusively on resolving "it"). Although the output does not perfectly correspond to the true coreference relationships in the text, it is a significant improvement over the output of the clustering algorithm without constraints. In the next section, we will see how to quantitatively evaluate this output.

5.3 Evaluation

To determine whether our encoding of linguistic knowledge and use of COP-COBWEB is an effective approach to the noun phrase coreference resolution problem, we need a method for evaluating the accuracy of the noun phrase partitions that we create. Unfortunately, there is little agreement in the NLP community about the best method for evaluating performance on this problem. We have identified eight different metrics that can be used, each of which assesses performance in a different way. A detailed analysis of the benefits and disadvantages of each metric is beyond the scope of this dissertation. However, we will describe all eight metrics briefly and present the results obtained using each metric. Although the numbers generated by the metrics differ, the general conclusions we can draw from them do not. In this section, we provide some terminology that will aid in interpreting our results. We introduce three common components in coreference evaluation metrics: *recall, precision,* and *f-measure.*

Recall and precision. Recall and precision are two criteria commonly used for evaluation in Information Extraction and Information Retrieval. In particular, several of the proposed noun phrase coreference metrics make use of them. *Recall* calculates how many of the desired items (in this case, coreference links) were correctly identified, and *precision* computes the percentage of items returned that are correct. In other words, recall is concerned with coverage, and precision is concerned with accuracy. Recall is defined as $\frac{N_c}{N_t}$ and precision is $\frac{N_c}{N_r}$, where N_c refers to the number of returned items that are correct, N_t is the total number of correct items that could be returned, and N_r is the number of (correct and incorrect) items that are returned.

F-measure. A trivial way to get 100% recall would be to put all of the noun phrases into one cluster, since this would definitely cover all of the correct coreference links (i.e., $N_c = N_t$). However, precision would be very poor since so many *incorrect* coreference links would be implied by that partition (in particular, precision would be $2 \cdot \frac{N_t}{n(n-1)}$). For evaluation purposes, these two numbers are usually combined into a single value called the *f-measure*:

$$\text{f-measure} = \frac{(\beta + 1.0)PR}{\beta P + R}. \tag{5.1}$$

where R and P refer to recall and precision, respectively. The parameter β varies between 0 and 1; it allows the evaluator to indicate the relative importance of the two values. We use $\beta = 1$, which gives equal weighting to recall and precision.

We calculate recall and precision by comparing two partitions of the input. The correct partition is referred to as the *key*, and the output being evaluated is the *response*. To evaluate the response, we must first decide what the "items" being returned are. One method is annotate coreference by linking each anaphoric noun phrase[2] to each of its coreferent noun phrases. If we designate the set of coreference links (over which a transitive closure has been taken) in the key as CL_k and the set of links in the response as CL_r, we then have

$$N_r = |CL_r| \tag{5.2}$$
$$N_t = |CL_k| \tag{5.3}$$
$$N_c = |CL_k \bigcap CL_r| \tag{5.4}$$

[2]Anaphoric noun phrases are those that have a coreferent antecedent (not all noun phrases in a document do).

This intuitive notion of agreement corresponds exactly to the Jaccard method of comparing two partitions, which we will present in more detail shortly. First, we discuss the methods that are most commonly used by NLP researchers for evaluating coreference performance. Some methods tend to view coreference groups as *chains* instead of clusters (these views are equivalent, but they have led to different evaluation methods). Chains are specified by linking each noun phrase to its closest coreferent antecedent. Thus, a group of g mutually coreferent noun phrases will have $g-1$ links under this scheme, while the same group would have $\frac{1}{2}g(g-1)$ links according to Equation 5.2.

Coreference evaluation metrics. The most common method for evaluating coreference chains involves determining the minimum number of links that would need to be added (to eliminate recall errors) or removed (to eliminate precision errors) (Vilain et al., 1995). We refer to this approach as the **Vilain metric**. However, there are important drawbacks to this method which have been discussed by several authors. The first is that the Vilain metric only evaluates accuracy on noun phrases that are involved in some coreference relationship, so correctly determining that a noun phrase is *not* coreferent with any other noun phrase is unrewarded. Second, the Vilain metric can report inappropriately high performance numbers for some partitions, because it only calculates the minimum number of changes that would be needed to convert the response into the key. Bagga and Baldwin (1998) instead propose the **B-CUBED** evaluation method, which scores each noun phrase individually for recall and precision, including those not involved in a coreference relationship. Popescu-Belis and Robba (1998) suggest matching each noun phrase class in the response to a class in the key (**COR**). They count each noun phrase in the response that is not attached to any key class as a recall error and each noun phrase attached to an incorrect key class as a precision error. They also suggest a refinement to COR that identifies an exclusive (one-to-one) mapping from key classes to response classes (**XCOR**). Trouilleux et al. (2000) further expand on the ideas of Popescu-Belis and Robba by establishing different *specificity classes* of noun phrases (proper name, lexical head, and pronoun). These classes are used to improve the one-to-one mapping of key classes to response classes. We refer to their method as **TROU**.

Generic partition similarity metrics. Because we view the key and response as partitions of the same set of noun phrases, we can also make use of standard methods for computing the similarity of two partitions, P_1 and P_2. The most common of these is the **Rand** metric, which was described in Section 3.5.1. It views a partition as a set of $\frac{1}{2}n(n-1)$ pairwise decisions between items. We define values a, b, c, and d according to the following table:

		P_1	
		class(d_i) = class(d_j)	class(d_i) $\neq$ class(d_j)
P_2	class(d_i) = class(d_j)	a	b
	class(d_i) $\neq$ class(d_j)	c	d

Table 5.4: COP-COBWEB noun phrase coreference results on the example text shown in Figure 5.1, as assessed by eight metrics. R stands for Recall, P stands for Precision, and F stands for f-measure.

	No constraints			**With constraints**		
Metric	(R	P)	F	(R	P)	F
Vilain	(75.0	23.1)	35.3	**(75.0**	**75.0)**	**75.0**
B-CUBED	(91.7	22.5)	36.1	**(91.7**	**93.8)**	**92.7**
COR	(75.0	7.7)	14.0	**(75.0**	**75.0)**	**75.0**
XCOR	(25.0	25.0)	25.0	**(93.8**	**93.8)**	**93.8**
TROU	(100.0	7.7)	14.3	**(75.0**	**75.0)**	**75.0**
Rand	58.3			**97.5**		
Jaccard	5.7			**50.0**		
FM	18.8			**67.1**		

The columns indicate decisions made by P_1 and the rows indicate decisions made by P_2. The diagonal entries represent situations where both partitions agree; off-diagonal entries represent decisions they disagree on. (This kind of table is also referred to as a *confusion matrix*.) Rand calculates agreement as

$$\text{Rand} = \frac{a+d}{a+b+c+d}. \tag{5.5}$$

In contrast, the **Jaccard** metric does not count decisions where both partitions placed the pair of items in distinct clusters (d):

$$\text{Jaccard} = \frac{a}{a+b+c}. \tag{5.6}$$

Fowlkes and Mallows (1983) calculate a more complicated function of these values (**FM**):

$$\text{FM} = \frac{a}{\sqrt{(a+b)(a+c)}}. \tag{5.7}$$

This metric, like the Jaccard metric, ignores d (the number of pairwise decisions where both partitions placed the items in separate clusters). The sum $b + c$ is the total number of decisions that the two partitions disagree on. By dividing by $\sqrt{(a+b)(a+c)}$, the FM metric calculates a larger penalty for disagreements that are equally divided between b and c (i.e., the disagreements are spread out over the entire partition). Given a fixed number of disagreements $b + c$, FM is maximized when either b or c is zero (this means that a single cluster in one partition was split into multiple clusters in the other partition).

Using the example text in Figure 5.1, we evaluated the coreference system both with and without constraints using each of these eight metrics. The noun phrase bracketing and the feature generation were done by hand. The results are in Table 5.4. The first five metrics calculate recall and precision, while the latter three simply calculate agreement. Recall for the unconstrained output is generally fairly high, but precision is very low. In contrast, the use of constraints boosts precision to high levels, resulting in overall gains in f-measure. The three partition agreement metrics also show large improvements when constraints are used. Although each metric assesses performance in a slightly different way, they all agree that the output produced by COP-COBWEB when using constraints is superior to the output produced by regular clustering. This agrees with our informal impression of the output. The same eight metrics will be used in the next section to analyze the performance of COP-COBWEB on a larger body of evaluation texts.

5.4 Quantitative Evaluation of COP-COBWEB

To determine which of several coreference systems exhibits the best empirical performance, evaluation on a standard, common data set is crucial. The Message Understanding Conference (MUC) provides a common testing ground by putting together collections of manually annotated news articles. In addition to evaluating systems for their performance on other information extraction tasks, MUC offers a coreference resolution track. Each participant builds a coreference system, and then all systems are tested on the same data under the same conditions (MUC-6, 1995). This standardization makes it possible to run our system on the same data and to make precise comparisons.

Each MUC data set consists of a set of news articles. The noun phrases are not identified prior to processing, which means that noun phrase detection is an additional task that is required of each participant. The default scoring system uses the Vilain metric (Vilain et al., 1995) to compare the system's coreference output to the key document, which is annotated with the correct coreference information.

In our opinion, the lack of a standard set of input noun phrases is a big drawback to the MUC evaluations, because a coreference system may be unnecessarily penalized if the noun phrases it is working with differ from the noun phrases identified in the key document. To explicitly measure only a system's coreference ability, we would want to provide each system with the same input set of noun phrases. This has not been done, probably because it is difficult to separate a coreference resolution system from the parsing system it relies on to identify the noun phrases (and possibly their syntactic features). To better evaluate our system's coreference performance, we created a modified set of MUC key documents that use the same set of base noun phrases that our system receives as input. In the next section, our evaluations make use of these "baseNP keys." For quantitative comparisons with other systems (Section 5.5), we fall back on the default MUC keys.

5.4.1 Results on the MUC-6 Dry Run Data Set

The MUC-6 evaluation includes two data sets: the *dry run* data set, used for training and development, and the *formal* data set, used for a blind evaluation of

Table 5.5: COP-COBWEB noun phrase coreference results on baseNP keys for the MUC-6 dry run evaluation, assessed by eight metrics

Metric	No constraints	Constraints	Pruned constraints
Vilain	50.1	**53.0**	53.9
B-CUBED	15.2	**74.3**	74.4
COR	22.3	**41.5**	42.0
XCOR	15.8	**69.9**	70.0
TROU	13.8	**28.9**	29.3
Rand	51.0	**96.9**	96.9
Jaccard	3.5	**20.3**	20.4
FM	15.1	**33.4**	33.5

the systems. Each data set contains 30 documents. The results of running our system on the MUC-6 dry run evaluation data sets are given in Table 5.5. We report f-measure for the first five metrics and agreement values for the others. By comparing the numbers in the second and third columns, we see that each of the metrics shows a significant improvement in performance when constraints are used. This improvement occurs despite the fact that the constraint heuristics only have access to shallow parsing information (base noun phrases). If our method were used in conjunction with a more sophisticated parser, we would be able to encode additional knowledge about coreference in the constraint sets, such as disallowing a coreference link between a subject and an object unless the object is a reflexive pronoun.

The smallest improvement occurs when the output partitions are assessed by the Vilain metric. It is known that this metric is often overly generous in scoring "bad" partitions (Popescu-Belis and Robba, 1998; Trouilleux et al., 2000). Popescu-Belis and Robba state that the Vilain metric is "too indulgent" because it calculates the *minimum* number of links that would need to be corrected to transform the response into the key. Visual inspection of the unconstrained output partitions suggests that this is probably happening. We will explain the contents of the fourth column shortly.

5.4.2 Accuracy of Coreference Constraints

We have noted that these hard constraints are generated by general heuristics and therefore are not guaranteed to be 100% accurate. It is important to get an idea of just how accurate the constraints are. We have calculated, for each heuristic, the percentage of constraints generated by that heuristic that agree with the true coreference annotations on the MUC-6 dry run data set. Table 5.6 presents the computed accuracy of each heuristic. The appositive heuristic is remarkably unreliable, which is a direct consequence of our poor ability to accurately detect appositive noun phrases. Consequently, we did not use this heuristic to generate

Table 5.6: Accuracy of heuristics as evaluated on the MUC-6 dry run data set

ID	**Heuristic**	**Accuracy**
HC_1	Number	98.95%
HC_2	Animacy	99.05%
HC_3 and HC_4	Gender	99.01%
HC_5, HC_6, and HC_7	Articles	99.57%
HC_8 and HC_9	Semantic class	99.20%
HM_1	Appositive	46.15%
All	All	98.96%
All but HM_1	No appositive	98.99%

constraints for the results in Table 5.5. When using all of the heuristics, the total accuracy is 98.96%; without heuristic HM_1, accuracy improves slightly to 98.99% (the effect of HM_1's inaccuracy is small because it is a heuristic that generates relatively few constraints).

We can see that the most reliable heuristics are conditioned upon the articles in the noun phrases. Surprisingly, the heuristic that prevents mismatches in NUMBER is the least reliable. This is probably due to our very simple assignment of singular and plural feature values to the noun phrases, because the heuristic itself is generally a very reliable one in English. As with the appositive heuristic, a more accurate classifier for this feature would improve the accuracy score for HC_1. Its accuracy is still high enough for it to be a useful source of constraints, so we will use it for our experiments.

5.4.3 Clustering with Pruned Constraints

Because the set of input constraints is not 100% accurate, it is difficult to determine whether errors made by COP-COBWEB are due to clustering mistakes or to incorrect information in the constraints. In general, we will not be able to automatically tell which of our heuristically-generated constraints are accurate and which are not. However, we are interested in determining how sensitive COP-COBWEB is to the accuracy of the input constraints. Because we have access to the true answer for each noun phrase, it is possible for us to prune the constraint set by removing each incorrect constraint. In addition to experiments with the heuristically-created $Con_=$ and $Con_{\neq}$, we also experimented with $PCon_=$ and $PCon_{\neq}$, their pruned counterparts:

$$PCon_= = \{(d, d') \in Con_= | d.label = d'.label\} \quad (5.8)$$

$$PCon_{\neq} = \{(d, d') \in Con_{\neq} | d.label \neq d'.label\} \quad (5.9)$$

The *label* attribute is the true class assignment of each noun phrase. The accuracy of $PCon_=$ and $PCon_{\neq}$ is guaranteed to be 100%. The fourth column in Table 5.5

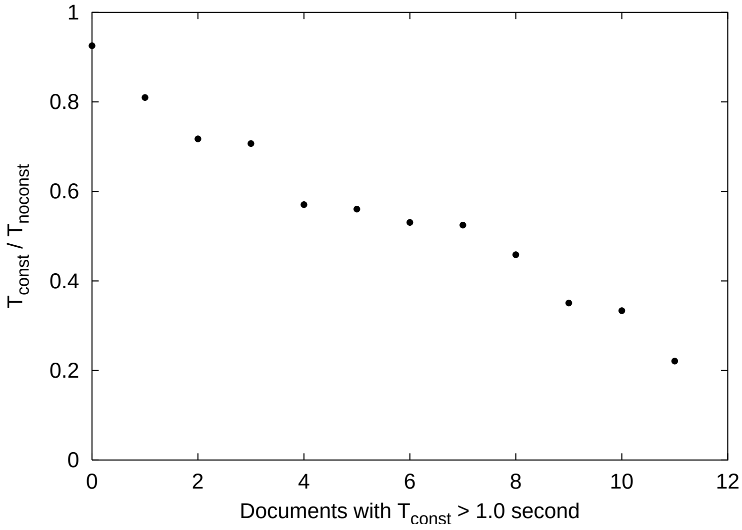

Figure 5.4: Runtime improvements observed when clustering with constraints on the MUC-6 dry run data set. T_{const} and $T_{noconst}$ are the runtimes for clustering with and without constraints, respectively. Only those documents with $T_{const} > 1.0$ second when using constraints are shown.

presents the results we obtain when providing the pruned constraints as input to COP-COBWEB. In each case, clustering accuracy either stays the same or goes up slightly. We conclude that the 1% inaccuracy of the original constraint set, although definitely a factor when applying constrained clustering to this problem, should not have a large impact on future results.

5.4.4 Runtime Improvements

In Chapter 3, we mentioned that empirically, we have observed significant runtime savings when using COP-COBWEB with constraints. Let T_{const} be the time required to run COP-COBWEB with constraints and $T_{noconst}$ be the time required in the absence of constraints. In Figure 5.4, we plot $T_{const}/T_{noconst}$ for several documents from the MUC-6 dry run data set. We have only included the 12 documents for which $T_{const} > 1.0$; the remaining 18 documents were each processed in less than 1.0 second when using constraints. Runtime savings on these 12 documents range from 7 to 78%. Savings for the formal evaluation data set are even more dramatic; they range from 21 to 93%. These improvements in runtime occur because the constraints provided to COP-COBWEB exclude certain areas of the search space from consideration.

We have discussed COP-COBWEB in isolation by comparing its performance with and without constraints and demonstrating that the addition of pairwise constraints is a clear benefit to the algorithm, in terms of accuracy and efficiency.

Table 5.7: Summary of MUC-6 results and baseline performance on the MUC-6 formal data set, using the Vilain metric

Method	Recall	Precision	F-measure
Worst reported in MUC-6	36	44	40
Best reported in MUC-6	59	72	65
Baseline 1: All one class	80	34	47
Baseline 2: Match head noun	46	52	49

Next we will compare COP-COBWEB's performance to that of other state-of-the-art noun phrase coreference systems.

5.5 Quantitative Comparison to Other Systems

In this section, we compare our system's performance to other coreference resolution systems. We will continue to make use of the Vilain metric for quantitative comparisons. As mentioned above, several shortcomings of this metric have been identified (Popescu-Belis and Robba, 1998; Bagga and Baldwin, 1998; Trouilleux et al., 2000). However, the metric has been consistently used by participants in the MUC competitions, so it provides the only means for quantitative comparisons. It is important to keep in mind that, for these comparisons, the results do not just evaluate coreference accuracy. As discussed above, each system identified noun phrases in a different way, so errors in noun phrase identification are folded in with errors in coreference resolution in the output of any given system.

5.5.1 Results on the MUC-6 Formal Data Set

The formal MUC-6 evaluation included seven different systems. F-measures ranged from 40% to 65% (see Table 5.7). The best scoring system (Appelt et al., 1995) attained a recall of 59% and a precision of 72%, while the worst scoring system had a recall of 36% and a precision of 44%. Most coreference systems use a set of hand-coded filters or heuristics and algorithms that are highly specialized to the coreference task. The main exception is work that provides labeled training texts to a decision tree learner and outputs a classifier that can automatically label a pair of noun phrases as "coreferent" or "not-coreferent." We will discuss two simple baseline methods, summarize methods that do not use learning techniques, and then compare supervised and unsupervised coreference systems to our experiments with COP-COBWEB.

Baseline 1: All one class. The first baseline, "all one class," places all of the noun phrases into one large class; it claims that every noun phrase is coreferent with every other noun phrase in the document. This baseline is very simple, yet as shown in Table 5.7, its performance is surprisingly high according to the Vilain metric (this is one of the weaknesses of this metric). This baseline would attain a recall of 100% if we had access to the full set of noun phrases used by the human annotators on these documents. Since we are using our own set of noun phrases, the recall obtained by this baseline (80%) instead indicates the upper bound we can expect to obtain for recall. The remaining 20% of coreference links involve noun phrases that do not appear in our input data set. The missing noun phrases include nested noun phrases such as "Enron" in "the Enron executive," since Empire does not detect nested noun phrases, as well as noun phrases that are omitted due to parsing mistakes, such as incorrect part of speech tags. "All one class," as expected, attains a very low precision (34%).

Baseline 2: Match head noun. The "match head noun" baseline is slightly more sophisticated. It posits a coreference link between any two noun phrases that have the same head noun. This very simple rule also performs quite well, attaining 46% recall and 52% precision (49% f-measure).

Non-learning approaches. As mentioned above, most coreference resolution systems make use of hand-crafted filters and heuristics rather than techniques from machine learning. Azzam et al. (1998) evaluated their *focus-based* system, LaSIE, on the MUC-7 formal evaluation texts (MUC-7, 1998). LaSIE tracks the current conceptual *focus* at each point in the text as well as a list possible *actors*. Their system obtains an f-measure of 60.4%, but they also show that a very simple heuristic-based pronoun resolution method outperforms the focus-based method, achieving 62.3%.[3] The authors posit that this is because the focus-based method relies heavily on a sophisticated analysis of the text (to determine what the current focus is). This analysis is difficult to do well automatically, and errors in that analysis cause errors in the coreference output. We have also observed that errors in our computed noun phrase features have an impact on coreference accuracy, but our system is likely to be less sensitive to this effect because we are computing simple features.

Supervised learning approaches. McCarthy and Lehnert (1995) and Aone and Bennett (1995) use supervised learning techniques to determine noun phrase coreference relations. Both systems train a C4.5 decision tree classifier (Quinlan, 1993). The resulting decision tree takes in two noun phrases and outputs whether or not they are coreferent. The decision tree trains on a data set consisting of several noun phrases *pairs* that are manually annotated as "coreferent" or "not coreferent." McCarthy and Lehnert create $\frac{1}{2}n(n-1)$ training instances by pairing each noun phrase with every other noun phrase. Aone and Bennett train only on noun phrase pairs where the second noun phrase is anaphoric. McCarthy and Lehnert participated in the MUC-6 evaluation, so we can compare our scores

[3]These results do not appear in Table 5.7 because they are evaluated on a different data set.

Table 5.8: Quantitative comparison of noun phrase coreference systems that use machine learning techniques, on the MUC-6 formal data set using the Vilain metric

Method	Recall	Precision	F-measure
Supervised methods			
RESOLVE (decision tree) (McCarthy and Lehnert, 1995)	44	51	47
Decision tree, pruning non-coref examples (Soon et al., 2001)	59	67	63
Decision tree, pruning coref and non-coref (Ng and Cardie, 2002)	63	77	70
Unsupervised methods			
NP constrained clustering (Cardie and Wagstaff, 1999)	53	55	54
COP-COBWEB, no constraints	72	31	43
COP-COBWEB, with constraints	50	40	44

directly to theirs. On the formal evaluation corpus, their system, RESOLVE, obtained a recall of 44% and a precision of 51% (f-measure 47%). Table 5.8 presents these results as well as figures for the other learning methods we will discuss.

Although both systems demonstrate the ability to learn from the labeled examples, there are some drawbacks to this approach. The first is that coreference is a transitive relation, so making pairwise coreference decisions in isolation requires a later "coordination" step to resolve any conflicts. For example, if the system decides that NP_i and NP_j are coreferent, and NP_j and NP_k are coreferent, but NP_i and NP_k are *not* coreferent, the coordination step must decide how to resolve this so that transitivity is upheld. McCarthy and Lehnert mention this issue but do not discuss how they solved it. Others have suggested using a clustering algorithm as a post-processing step to ensure that the output is in fact a partition (Soon et al., 2001). Another problem with training on noun phrase pairs, identified by Soon et al., is that the class distribution of the training set is highly skewed: the number of coreferent pairs is much smaller than the number of non-coreferent pairs. This situation is known to cause problems for supervised learners (Cardie and Howe, 1997). Soon et al. directly addressed this problem by intelligently pruning the training set to reduce the number of non-coreferent pairs. In evaluation on the MUC-6 formal data set, their method achieved a recall of 59% and a precision of 67% (f-measure 63%). Ng and Cardie (2002) further showed that intelligent pruning of the coreferent pairs as well can also improve performance, to an f-measure of 70% (see the second and third entries in Table 5.8).

Unsupervised learning approaches. In contrast to the above *supervised* approaches to noun phrase coreference resolution, we previously developed a custom clustering algorithm created specifically for this problem (Cardie and Wagstaff, 1999). This method takes in one parameter, a cluster *radius* threshold, and uses a custom distance measure to determine how close two noun phrases are. Linguistic constraints and preferences (soft constraints) are incorporated in the distance calculation; this system can therefore encompass a wider variety of domain knowledge than our hard constraints can. For the MUC-6 formal evaluation, this system obtained a recall of 53% and a precision of 55% (f-measure of 54%); we refer to this method as "NP constrained clustering" in Table 5.8. This system has the substantial benefit of not requiring any labeled training examples; it deduces coreference relationships solely on the "similarity" of the noun phrases under consideration. In addition, it out-performs the only other learning approach used in the MUC-6 evaluation (RESOLVE). However, it does not outperform the best-scoring system (as mentioned previously, this system had an f-measure of 65%). In addition, the decision tree methods previously discussed also perform significantly better.

Another approach that does not require human intervention is that of Dagan and Itai (1991). They focused exclusively on performing coreference for the pronoun "it," arguably the most difficult noun phrase to resolve. For each occurrence of "it," they collected co-occurrence statistics from a large body of texts that were parsed, but not labeled with coreference information. These statistics were computed for all subject-verb and verb-object pairs and used to filter out implausible antecedents. The approach looks promising, but unfortunately was only evaluated on 74 occurrences of "it" (not from the MUC texts), so it is difficult to compare to other work here. For those 74 examples, performance using their statistical filter improved from 64% to 74% (in terms of how many "it"s were assigned to their correct nearest antecedent).

5.5.2 Comparison of COP-COBWEB to Other Coreference Systems

For these data sets, and using the Vilain metric, COP-COBWEB achieves a recall of 72% and precision of 31% (43% f-measure) without any constraints. Performance improves with the inclusion of constraints to an f-measure of 44%. Because most of the constraints are cannot-link constraints, we expected precision to improve when constraints are used. It is also unsurprising that recall goes down, due to inaccuracies in the feature value computations and in the constraint heuristics themselves. We will explain why the Vilain scores obtained by COP-COBWEB appear low and then discuss the advantages of the COP-COBWEB approach to this problem.

Factors influencing COP-COBWEB's Vilain scores. There are several reasons that we do not observe higher f-measure values for COP-COBWEB, most of which arise from our experimental setup and are independent of the algorithm used. The most significant factor, which we have previously mentioned, is the mismatch between the set of noun phrases we obtain from Empire and the set of noun phrases used in the key documents provided by MUC. Since 20% of the coreference links in the keys involve noun phrases we do not have access to, there is a large impact on recall. Secondly, we have also noted the drawbacks of using the Vilain metric for evaluating noun phrase coreference. Although this metric reports only a small improvement in COP-COBWEB's performance compared to clustering without constraints, the other metrics we have used show much larger improvements (see Table 5.5). Finally, our feature values are probably not very accurate representations of the noun phrases. These three difficulties would arise regardless of which algorithm we used, so their contributions to the low f-measures reported do not tell us anything specifically about COP-COBWEB's abilities.

Other factors are more directly relevant to an assessment of COP-COBWEB. We have previously noted that the constraints used by the algorithm are imperfect, due to their heuristic generation. However, our experiments with pruned constraint sets have shown that this inaccuracy is unlikely to have a large impact in our results. It is also the case that supervised methods often out-perform unsupervised methods when applied to the same task. The supervised methods have specific examples of the desired classifications. In contrast, unsupervised methods simply seek a "good" organization of the data, which may or may not correspond to the specific concept encoded in the labels used for evaluation. COP-COBWEB is a hybrid algorithm; it has more guidance than a purely unsupervised method, in the form of constraints, but it does not have access to explicitly labeled examples of coreferent relationships.

Finally, we expect that our set of constraint heuristics is incomplete. Each heuristic generated constraints that improved the performance of COP-COBWEB, but it is likely that there are other applicable rules about coreference that could (and should) also be added to the set of heuristics. We have demonstrated that COP-COBWEB can effectively apply the constraints it is given; augmenting the constraint set with additional (reliable) information is likely to improve performance further.

Advantages of using COP-COBWEB for coreference resolution. Although the Vilain scores for COP-COBWEB do not exceed those of other algorithms, there is a significant advantage to using COP-COBWEB on this problem. Domain knowledge from a variety of sources can be easily encoded using our constraint formulation. For example, although the Vilain scores for both baselines are higher than those reported for COP-COBWEB, we could emulate the performance of either one by using a suitable set of constraints. Baseline 1, which places all of the noun phrases into a single class, can be duplicated by creating a must-link constraint between each pair of consecutive noun phrases in the document. Baseline 2, which connects each pair of noun phrases with identical head nouns, can be duplicated by creating a must-link constraint between noun phrases with matching head nouns and a cannot-link constraint between every other pair of noun phrases.

In addition, the use of soft constraints, rather than hard constraints, should further improve the performance of this algorithm. Because we could only encode knowledge as hard constraints, we were forced to omit information that functions as a "preference" rather than a "constraint." For example, HC_5 claims that a noun phrase with an indefinite article, such as "a cat," cannot link backwards to a noun phrase that is not a proper name or a pronoun. However, HC_5 is violated by a sentence such as "[Her cat] is [a cat] that literally climbs walls." Consequently, we would like to indicate that HC_5 is *usually* but not *always* reliable. In Chapter 6, we will see how soft constraints, which allow the specification of a *strength* for each constraint, can be formulated and integrated by a constrained clustering algorithm. We have reason to believe that this would lead to improved Vilain scores, due to our previous experience with a clustering algorithm specialized to the problem of coreference resolution (Cardie and Wagstaff, 1999). That algorithm used the same set of features that we have used for our experiments with COP-COBWEB, but it was able to accommodate preferences as well as constraints. The critical reader then asks why one might prefer COP-COBWEB to that algorithm, if we simply duplicate its performance on this problem. The answer highlights another significant advantage of COP-COBWEB: its generality. COP-COBWEB can be applied to any clustering problem, not just that of noun phrase coreference.

5.6 Summary

In this chapter, we applied the COP-COBWEB constrained clustering algorithm to the problem of noun phrase coreference resolution. This challenging problem is far from being considered "solved" by the NLP community, and a diverse array of techniques are being applied to it. We described how background linguistic knowledge about the coreference relation can be encoded as a set of instance-level constraints and provided as input to COP-COBWEB. The results of eight different metrics agree that this is an effective way to improve performance on this problem. We also compared COP-COBWEB's performance to other methods specifically designed with the coreference problem in mind. The best available coreference system, which uses supervised learning techniques and intelligent pruning of the training set, out-performs COP-COBWEB, which is not surprising. We do find, however, that the general COP-COBWEB algorithm, when specialized to this task through the use of constraints, provides an improvement over not using constraint information. In addition, COP-COBWEB allows for the encoding and use of a diverse set of constraints. We expect that applying COP-COBWEB to this problem with soft constraints will demonstrate further improvements.