Simple Coreference Resolution with Rich Syntactic and Semantic Features: Is it enough?

Research report submitted in partial requirements of the MS with Distinction in Research degree

Karthik Raghunathan
Department of Computer Science
Stanford University

Advisors:

Daniel Jurafsky
Professor of Linguistics
Professor by Courtesy of Computer Science
Stanford University

Christopher Manning
Associate Professor of Computer Science
Associate Professor of Linguistics
Stanford University

Table of Contents

Abstract

Haghighi and Klein (EMNLP 2009) showed that even a simple deterministic coreference resolution system with rich syntactic and semantic features can outperform all unsupervised systems and most supervised ones. Their results show that they get the highest gain in performance by adding the knowledge of semantically compatible head pairs to their system, which they say can be easily mined from unannotated corpora using a simple bootstrapping technique. Their approach, however, required that the system know the test sentences in advance, and used the entities in the test set to guide the bootstrapping algorithm. By presenting a generalized test set-independent implementation of their system, we show that their bootstrapping approach does not work in general, and that semantic knowledge mined in this way in fact hurts the system. Our results thus show that their algorithm is probably not applicable in the general case when the test set is not known in advance. We instead propose a set of easily reproducible enhancements to the Haghighi and Klein system which enable our system to be almost as competitive as theirs, despite not using any externally mined semantic knowledge. Our system not only does away with the need to know the test data in advance, it also removes the dependence on large amounts of parsed data for mining semantic head pairs. Additionally, by replacing the unreliable bootstrapping algorithm with more straightforward techniques, we provide a system that is just as strong, but is even easier to reproduce as a baseline.

1 Introduction

A highly common phenomenon in natural language discourse is for a new entity to be introduced in a particular way, but then get referred to, via different *referring expressions*. For instance, consider the following piece of text:

*It is not a secret that **Sony Corp.** is looking at 8mm as an all-purpose format. More important to the future of 8mm is **Sony**'s success in the $2.3 billion camcorder market. **The Japanese company** already has 12% of the total camcorder market, ranking **it** third behind the RCA and Panasonic brands.*

Here, the author has first introduced the new entity "Sony Corporation" by using the phrase "Sony Corp.". He next drops the "Corp." suffix and refers to the company as just "Sony". He then stops using the name altogether and instead uses a descriptive noun phrase ("The Japanese company") first, and then a pronoun ("it"). *Coreference resolution* is the task of finding all such expressions that refer to the same entity in a given piece of text. In the above example, "Sony Corp.", "Sony", "The Japanese company" and "it" are the different referring expressions for the same entity – the company named Sony Corporation. These expressions are said to *corefer* or be *coreferent* to each other, and the set of all four expressions together is called a *coreference chain*. The goal of a coreference resolution system is essentially to identify sets of coreferring expressions in a document and cluster them into separate *coreference chains*.

Humans while reading, can correlate referring expressions (i.e. do coreference resolution) almost naturally without any problems, using their experience and knowledge of the world around them. However, this is a daunting task for a machine. As can be seen from the above example, the referring expressions might not even have any substring in common, e.g. "The Japanese company" and "Sony". Thus, the system needs to be cleverer than just relying on surface similarity between different expressions for finding coreference.

An effective coreference resolution system is an important component in any NLP pipeline that deals with language understanding tasks such as question answering, document summarization, information extraction, etc. Being a crucial prerequisite for creating a machine that can comprehend natural language text, this problem has received a lot of attention in the past, starting with Jerry Hobbs' seminal paper (1977) on pronominal anaphora resolution. Pronominal anaphora resolution can be thought of as a subpart of the more general problem of coreference resolution. It refers to the task of finding an antecedent (i.e. a referring expression that has occurred earlier in text) for a given pronoun. For instance:

*Despite the heat of competition, Mr. Gerson of This Week in Consumer Electronics thinks **Sony** can succeed in the home-video market in the same way **it** has thrived in the highly competitive color-television-set market.*

Here, "Sony" is the correct antecedent for the anaphor "it". Hobbs tackled this problem using a simple deterministic algorithm that involved heuristically traversing parse trees of the given text and applying some basic constraints to weed out improbable antecedents. Assuming the availability of correct parses for the text, this algorithm was simple to implement and quite robust for doing pronominal anaphora resolution. However, most of the recent research on coreference resolution has been using statistical methods as opposed to using heuristic deterministic systems. These approaches mainly revolve around leveraging different pieces of evidence like syntactic constraints, semantic constraints and other discourse phenomenon for finding coreference. Most of the recent work in this field has therefore been focused on how to best combine and model all these various factors together. There have been both supervised (Culotta et al., 2007; Bengston and Roth, 2008; Finkel and Manning, 2008) and unsupervised (Bean and Riloff, 2004; Poon and Domingos, 2008) machine learning approaches to this problem using an inventory of large feature sets.

Despite all recent research being machine learning-oriented, Haghighi and Klein (2009) broke away from this tradition and went back to the Hobbs' way of tackling this problem. They presented an unsupervised, modular and deterministic system that only made use of a few rich features. Even with a simplistic approach, their system managed to beat the state-of-the-art unsupervised coreference resolution systems and was within striking distance of the supervised systems. One of their major contributions was a bootstrap technique to harvest semantically compatible head pairs (the head nouns of the referring expressions) from unlabeled data in an unsupervised way. Adding semantic information mined using this technique gave their system the highest boost in performance (as compared to other constraints or heuristics) – a gain of 3 F1 points on one test set and 9.2 F1 points on another.

Haghighi and Klein say that they offer their approach as "a very strong, yet easy to implement, baseline". Their system definitely offers a pretty strong baseline and even though most of their system is easy to reproduce, the same cannot be said about their semantic knowledge module. Bootstrapping techniques can often be unreliable and unless carefully controlled, can give totally unexpected results. Haghighi and Klein used the knowledge of the entities present in the test set to constrain their bootstrapping algorithm and make it work favorably for their purposes. However, this raises questions over the generality of their approach since it is not always possible to know the test set beforehand. We saw this as a major drawback in their system and tried addressing it by attempting to build a test set-independent semantic compatibility module, based on Haghighi and Klein's bootstrapping approach. Implementing an unbiased algorithm for semantic knowledge extraction is considerably tougher than a bootstrapping algorithm that is guided by the test set (see Section 4). Since our attempts at generalizing Haghighi and Klein's semantic module were unfruitful, we explored an array of different enhancements to the system which could offset the information lost by not having a semantic module in our system. Our final system not only performs just as well as the Haghighi and Klein (2009) system, but is considerably simpler to implement, takes lesser time and resources, and works without knowing the test set in advance.

The major contributions of this work are (i) detailed error analysis with examples and statistics regarding the different kinds of errors that show up in a basic coreference resolution system; (ii) discussion regarding the various challenges involved in implementing an effective semantic knowledge extraction system that works independent of the test set; (iii) a baseline which is almost as strong as Haghighi and Klein (2009), but more reliable and reproducible in practice; and (iv) introducing the notions of using a multi-pass system and a machine-learning based reranker to further improve the system's performance.

2 System Overview

We first give an overview of Haghighi and Klein's (2009) simple coreference resolution system, since their system is the starting point for our own experiments. They built their system in an incremental fashion, refining the features and constraints at each stage, and reporting the increase in performance on the ACE2004-ROTH-DEV corpus (see section 2.2):

i. Their initial system allowed proper and nominal mentions to only corefer with antecedents that have the same head, but allowed pronominal mentions to corefer with any antecedent. The mention heads were found using the Stanford parser (Klein and Manning, 2003) and using the Collins head rules (Collins, 1999). The probable antecedents were then ordered by raw text distance and the nearest one chosen as being coreferent. This system had a fairly low pairwise F1 score of 48.9.

ii. **Adding syntactic salience:** Next, they parsed all the sentences in the document using the Stanford Parser (Klein and Manning, 2003) and mapped each mention in the document to a node in the parse tree. The Hobbs tree distance between mention nodes (based on the tree traversal heuristic described by Jerry Hobbs (1977)) was now used as a measure of syntactic salience to select the closest antecedent, instead of raw text distance. This change by itself yielded a pairwise F1 of 51.7.

iii. **Adding agreement constraints:** Their next step was to add information about person, number and entity type for each mention and use them to filter out antecedents which did not agree in these attributes with the mention in question. The number feature was obtained using the POS tag of the head word. For entity types, they used the Stanford NER (Finkel et al., 2005) to annotate the NPs wherever possible and used a predefined set of compatible NER labels to annotate the pronominal mentions. Applying agreement constraints in the coreference decision gave them a pairwise F1 score of 53.4.

iv. **Adding syntactic configuration constraints and detecting appositions:** They further tightened the filter for candidate antecedents by adding the i-within-i syntactic constraint, i.e. a parent NP cannot be coreferent with any of its child NPs. However, appositives are an exception to this rule since the parent NP of an appositive node is in fact coreferent to it. They therefore implemented apposition detection to directly detect coreference between such nodes and then applied the i-within-i constraint for the remaining non-appositive nodes. This system got a substantially improved pairwise F1 of 55.4.

v. **Detecting predicate nominatives:** The predicative nominative construction is another highly reliable coreference pattern. They deterministically annotated and used predicative nominatives for coreference, analogous to the appositives. This yielded a minor improvement to 55.5 F1.

vi. **Using semantic knowledge:** Finally, in order to resolve the remaining cases of non-pronominal reference (that could not be detected using appositive or predicate nominative constructions), Haghighi and Klein resorted to using semantic knowledge about the various entities to disambiguate the references. They used a bootstrapping approach starting with reliable coreference seed patterns (appositions and predicate nominatives) and using them to discover a large set of semantically compatible head pairs from the BLIPP and WIKI datasets, which were then used to augment the coreference system. Non-pronominal mentions were now matched not only with the antecedents having the same head, but also with those having a semantically compatible head. Bootstrapping-driven extraction algorithms are usually prone to noise that needs to be filtered out. In order to make the bootstrapping algorithm work successfully, Haghighi and Klein did two clever tricks –
   a. They did not use complete documents or Wikipedia pages, but only the first paragraph in each article. The first paragraph, where most of the new entities are introduced is rich in appositives, predicate nominatives and similar helpful patterns.
   b. They used the knowledge of the entities present in the test set to select only those Wikipedia pages that contained one of the test mentions.

   These tricks helped them to ensure that they did not extract a lot of noisy patterns or head pairs. The addition of semantic knowledge using the bootstrapping approach gave them the highest boost in performance – an increase of 3 F1 points with the final system getting a pairwise F1 score of 58.5.

2.1 Overview of our work

We first attempted to implement Haghighi and Klein's (2009) system as a baseline using the Stanford JavaNLP[1] framework. JavaNLP is a suite of tools written in Java for various NLP tasks, developed and maintained by the Stanford NLP group. It includes both the parser and the NER system used by Haghighi and Klein. Our initial plan was to faithfully replicate their system and then attempt to enhance their semantic knowledge module by adding semantic patterns (automatically mined patterns that are highly indicative of coreference) to the system in addition to the semantically compatible head pairs (see section 4.3). However, looking at their semantic module's dependence on the test data, we decided to build a generic semantic extraction algorithm instead – i.e., one that does need to know the mentions in the test data beforehand. This turned out to be a rather non-trivial task. Thus, even though we could easily reproduce steps i – v mentioned above, we had a hard time in getting step vi to work favorably for us (explained in detail in section 4), since our version of the step vi was trying to address a tougher problem than theirs.

Similar to Haghighi and Klein, we built our system in an incremental manner, recording its performance on the MUC-6-TRAIN and ACE-ROTH-DEV development sets (see section 2.2) at each stage. We started with a basic system that included notions of syntactic salience in terms of tree distance, the different agreement constraints and the i-within-i syntactic configuration constraint. This would roughly correspond to step iii above with the addition of the i-within-i constraint from step iv. We analyzed the errors from this initial system to explore the various sources of errors and get some rough error distribu-

[1] http://nlp.stanford.edu/javanlp/

tion statistics (see section 3). This exercise helped us in two ways. First, it reaffirmed our faith in the Haghighi-Klein system since implementing steps iv through vi definitely seemed to be part of the solution for addressing three out of our top five concerns – namely appositive and predicate nominative detection, synonymy and hypernymy detection, and lack of external semantic/pragmatic knowledge or world knowledge. Secondly, it gave us insights into other generic error patterns that could guide us towards improving on the Haghighi-Klein baseline. Our system architecture is explained in detail in section 4. We next talk about the different data resources that we used for our experiments.

2.2 Data sets

Development:
We used these data sets as our development data for carrying out various experiments and recording the boost we got from each addition to our system.

- MUC-6-TRAIN (195 documents; 9,853 mentions): This is the formal training split from the MUC-6 conference.
- ACE2004-ROTH-DEV (68 documents; 4,536 mentions): This is development set split of the ACE 2004 training data utilized in Haghighi and Klein (2009) and Bengston and Roth (2008).

Testing:
We evaluated our final system on the following test set to compare our performance with previous work.

- MUC-6-TEST (30 documents; 2,141 mentions): This is the formal evaluation set from the MUC-6 conference used in Haghighi and Klein (2009).

Unlabeled:
Due to the availability of readily parsed data, we used the APW and NYT sections of the Gigaword corpus for automatically extracting semantic knowledge, in contrast to the BLIPP and WIKI datasets used by Haghighi and Klein.

3 Error Analysis

We tested our initial coreference system on the MUC-6 train set and used its corresponding gold annotations to analyze the results. Both precision and recall errors were classified according to their occurrences and reasons of occurrence.

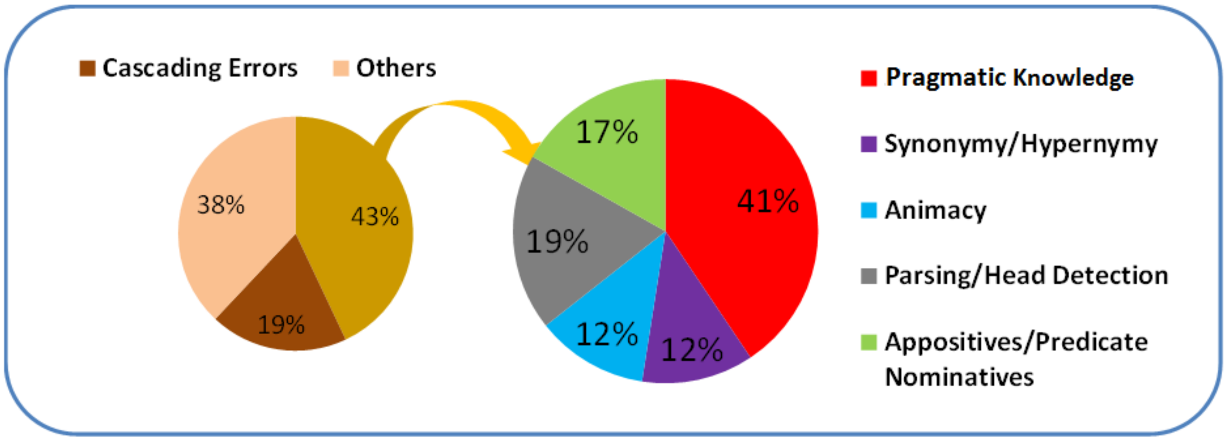


Figure 1: *Error distribution from a random sampling of errors made by the system on MUC-6 training data*

We randomly chose 137 errors from the full error set and got the distribution shown in figure 1. The errors fell into three general categories: “Definite errors” (43%) for whom the reasons were well defined, “Others” (38%) which were very specific errors which couldn’t be easily generalized and “Cascading errors” (19%) which were caused as a side-effect of the errors identified in the previous two sections. “Definite errors” was just a label we came up with for error patterns that occurred enough number of times to justify giving them a class of their own. The different subclasses within definite errors are discussed in detail later in this section. On the other hand, each of the error types in the “Others” category occurred only in one or two cases in our sample set. Thus it made more sense to lump of all of these cases into one big category instead of trying to label each type individually. A “cascading error” is a special kind of error which is caused due to a wrong decision made by the coreference system for an earlier mention. Consider the following text snippet:

Investor Harold Simmons and NL Industries Inc. offered to acquire ***the commodity chemicals concern*** *for $50 a share, or about $1.1 billion**. Georgia Gulf** rebuffed that offer in September and said* ***it*** *would study other alternatives.*

Coreference systems are evaluated based on the number of links (between mentions) that they get right. In the above example, for the mention “it”, the gold annotation would have two links:

1. *it -> Georgia Gulf*
2. *it -> the commodity chemicals concern (through it -> Georgia Gulf -> the commodity chemicals concern)*

Our system was able to detect the first link correctly. However, since it did not have the world knowledge that “Georgia Gulf” is a “commodity chemicals concern” (see section 3.1), it failed to link “Georgia Gulf” and “the commodity chemicals concern” as being coreferent. This missing link broke the coreference chain and the system thus could not detect the second link for “it”. The recall error for link 2 was a cascading error caused due to an earlier error (in this case, because of the lack of semantic knowledge) committed by the system.

The main intuition behind categorizing the errors into these three broad classes was to identify the set of errors that seemed most promising to address first in terms of performance gain for the system. Since each of the subclasses within the definite errors had a considerable representation within our sample set, our belief was that addressing any of these error types would have a widespread influence over the system’s performance on the data set and significantly reduce our error rate. In other words, finding the solution to a particular error pattern within the definite category was more likely to address a large number of error cases at once, than a pattern in the others category. Also, a considerable part of the cascading errors were due to the definite errors and thus correcting definite errors would in effect help us in solving more than half of the total errors made by the system. We came across seven main types of definite errors, as discussed in the remainder of this section.

3.1 Lack of Pragmatic/World Knowledge

Pragmatic knowledge/World knowledge helps the system to factor in real world information into its operation towards an informed decision on coreference. Consider the following example:

Norman Ricken, 52 years old and former president and chief operating officer of Toys “R” Us Inc., and Frederick Deane Jr., 63, chairman of Signet Banking Corp., were elected directors of ***Circuit City Stores Inc****. They succeed Daniel M. Rexinger, retired executive vice president, and Robert R. Glauber, U.S. Treasury undersecretary, on the 12-member board of* ***the consumer electronics and appliances retailing chain****.*

The real world knowledge of “Circuit City Stores Inc.” being a “consumer electronics and appliances retailing chain” would easily help in immediate coreference, which didn’t happen as the initial system lacked this knowledge. Similarly, the semantic knowledge of “the Video Walkman” being a “product” would have made the system deem them coreferent in the following example:

Sony Corp. has heavily promoted the ***Video Walkman*** *since* ***the product****'s introduction last summer, but Bob Gerson, video editor of This Week in Consumer Electronics, says Sony conceives of 8mm as a “family of products, camcorders and VCR decks,”' whose sales will reinforce one another.*

Lack of pragmatic knowledge contributed to the major bulk (41%) of the definite errors.

3.2 Head Detection Errors

These errors arose because of incorrect head detection by the Stanford parser in the mention phrases. For instance, consider:

More important to the future of 8mm is Sony's success in ***the $2.3 billion camcorder market****. The Japanese company already has 12% of* ***the total camcorder market****, ranking it third behind the RCA and Panasonic brands.*

Our system correctly detected “market” as the head of the first mention (“the $2.3 billion camcorder market”), but incorrectly detected “camcorder” as the head of the second mention (“the total camcorder market”). They were thus not deemed to be coreferent when in fact they were. Another peculiar head detection error was caused by the selection of punctuation or a formatting token (like parenthesis) in the mention as the head. This was noticed on several occasions. For example:

To compete with Video 8, ***Matsushita Electric Co.'s 51%-owned JVC unit*** *introduced in 1986 a compact version of its dominant VHS format, called VHS-C that is nearly as small as an 8mm videocassette.* ***JVC*** *says that an enhanced Super VHS version of VHC-C due this fall would be long enough to show a feature film in the extended-play mode with little detectable picture deterioration.*

Our system incorrectly chose the “’s” in the first mention (“Matsushita Electric Co.**’s** 51%-owned JVC unit”) as its head, which did not match with “JVC”, which was the head of the second mention (“JVC”). Incorrect head detection contributed to 19% of the definite errors.

3.3 Lack of Apposition Detection

An apposition is a grammatical construct consisting of two connected noun phrases, where one of the noun phrases merely serves to describe the other phrase. In most cases, the noun phrases are connected by a comma (“,”) between them. For example:

Sony Corp. has heavily promoted the Video Walkman since the product's introduction last summer, but ***Bob Gerson****,* ***video editor of This Week in Consumer Electronics****, says Sony conceives of 8mm as a ‘family of products, camcorders and VCR decks’, whose sales will reinforce one another.*

The comma in between the noun phrases “Bob Gerson” and “video editor of This Week in Consumer Electronics” indicates an apposition construction and hence, coreference between the two. Another example is:

The Texas oilman has acquired a 26.2% stake valued at more than $1.2 billion in ***an automotive-lighting company****,* ***Koito Manufacturing Co.***

Here “an automotive-lighting company” and “Koito Manufacturing Co.” are coreferent to each other. But our system failed to recognize coreferences like these due to lack of apposition detection.

3.4 Lack of Predicate Nominative Detection

Predicate nominatives are a phenomenon similar to appositives, but are considerably less common. They are grammatical constructs where a noun phrase follows a linking verb and renames or describes the subject noun phrase. The linking verb is mostly some form of the verb "be". For example:

*"Started three years ago, **Chemical's interest-rate options group** was **<u>a leading force</u>** in the field."*

This is a predicate nominative construction where the linking verb "was" indicates a coreference between the two noun phrases "Chemical's interest-rate options group" and "a leading force". Another example:

*"**Fed fund** is **the rate banks charge each other on overnight loans**; the Fed influences the rate by adding or draining reserves from the banking system."*

The coreference in this case is indicated by the linking verb "is". Our initial system could not detect the presence of a predicate nominative construction and thus failed to find coreference in such cases. The lack of successful detection of both appositives and predicate nominatives together constituted 19% of the definite errors.

3.5 Lack of Synonymy Detection

Mentions which are synonymous with one another are most often coreferent with one another. In the following example, knowing that "tapes" and "cassettes" are synonyms of each other would have helped in establishing coreference.

*At present, Sony itself is the primary distributor of prerecorded **8mm tapes** through its mail-order Cinema 8 movie club. Minneapolis-based Musicland Group announced last Friday that it will begin test marketing of prerecorded **8mm cassettes** at 19 of its 745 audio and video outlets this month .*

3.6 Lack of Hypernymy Detection

Similar to mentions being synonymous, a mention can also be a hypernym of the other. And similarly, a hypernym is most often likely to corefer with its hyponym, depending on the context. Here is an example:

*To compete with Video 8, Matsushita Electric Co.'s 51%-owned JVC unit introduced in 1986 a compact version of its dominant VHS format, called VHS-C that is nearly as small as an **8mm videocassette**. An **8mm cassette** can't be played on existing VHS equipment.*

In this case, knowing that cassette is a hypernym of videocassette would have helped in establishing coreference between the mentions "8mm videocassette" and "8mm cassette". Lack of synonym and hypernym detection contributed to 12% of the definite errors. These errors might sound similar to the ones listed in section 3.1 since knowing these relations (synonymy, hypernymy, etc.) in a way means having some sort of pragmatic knowledge about the mentions in question. We have still chosen to classify these errors separately because we believe that these errors could easily be fixed using a resource such as WordNet which contains information about such relations for a lot of common nouns. However, this is not true of the errors in the "Pragmatic Knowledge" category. Those clearly require outside world knowledge or other semantic information not easily available from a hand-built thesaurus.

3.7 Lack of Animacy Detection

Animacy detection corresponds to identifying an entity as being animate or inanimate. It can be useful as filtering constraint in coreference resolution since an animate mention cannot corefer with an inanimate one. For example:

> *"**Ideas** are going over borders and there is no SDI ideological weapon that can shoot them down", **he** told a group of Americans at the U.S. Embassy on Wednesday.*

Our system identified "Ideas" to be coreferent with "he". Clearly, the two could have been easily differentiated on the basis of animacy, however our system failed to do so since it lacked animacy detection. Detecting animacy is not always trivial. Some words are always either animate or inanimate e.g. "he", "she", "Americans", "cassette", "tape", etc. But there can also be cases where the context decides whether a particular word has an animate or an inanimate connotation. Consider the following two examples:

1. *He could develop the beach through a trust, but instead is trying have his grandson become a naturalized Mexican so his **family** gains direct control.*

2. *Bob Gerson, video editor of This Week in Consumer Electronics, says Sony conceives of 8mm as 'a **family** of products, camcorders and VCR decks', whose sales will reinforce one another.*

The mention "family" can thus be perceived as animate (a family of humans) or as inanimate (a family of inanimate objects or products) depending on the context. The lack of animacy detection contributed to 12% of the definite errors.

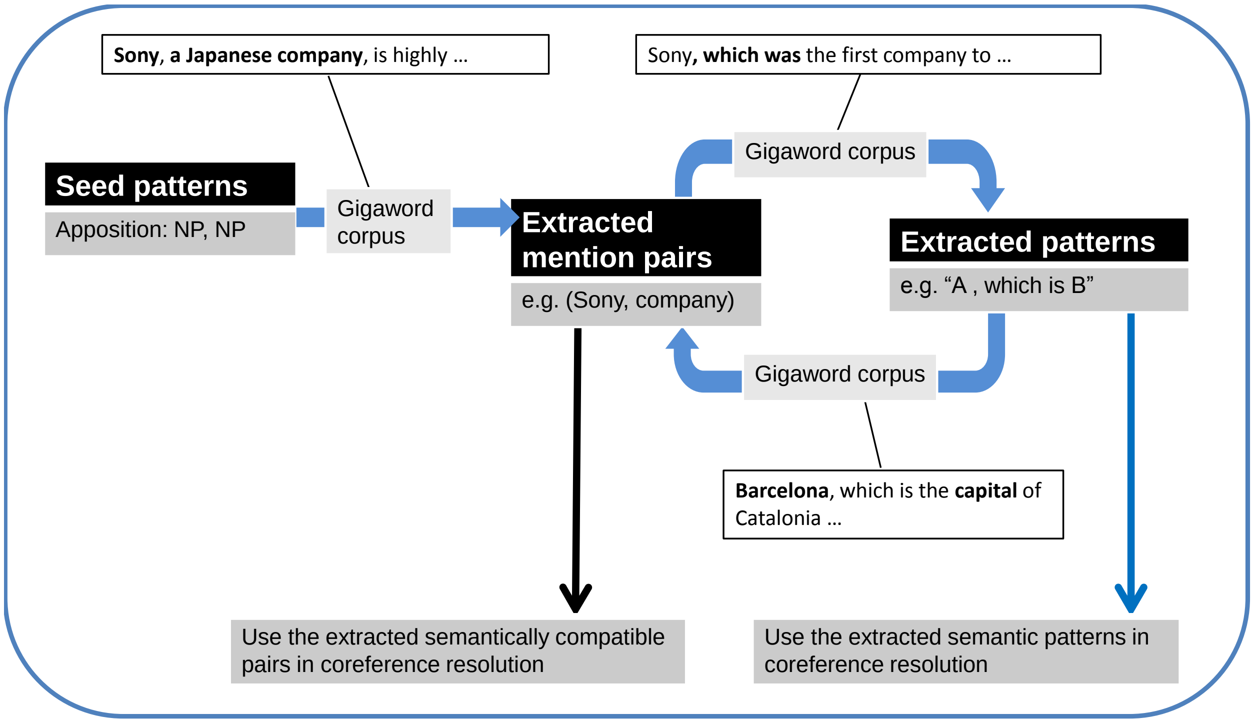
4 Simple Coreference Resolution

As introduced in section 2.1, our first iteration of the system roughly corresponded to steps i – iii and the i-within-i constraint from step iv of the Haghighi-Klein system. The minor detail in which we differed from them was by having gender agreement in addition to person, number and entity type in step iii, which was not of much consequence in the end. Also, in order to make our experiments simpler and faster, we directly used the entity type of the mentions from the gold annotations in the data set, instead of using the named entities provided by the Stanford NER. However, for a fair comparison with previous work, we do report the performance of our system using entity types tagged by the Stanford NER too. We ran our initial system on the MUC-6-TRAIN data set of 9,853 mentions and did error analysis to locate the different sources of errors (section 3). Our error analysis exercise showed us that the path to highest performance boost was via addressing the different kinds of definite errors. We first tried to address the head detection errors by implementing a better heuristic for head finding. In the initial system, the head was found by parsing just the text of the current mention, out of context. We fixed this by parsing the entire sentence containing the mention and then locating the mention span within the parse tree. This led to much better head detection accuracy and gave us our first true baseline scores: a pairwise F1 score of **53.64** on MUC-6-TRAIN and **48.85** on ACE-ROTH-DEV.

The next steps towards improvement involved replicating rest of Haghighi-Klein system to tackle the other definite errors. We implemented the detection of appositives and predicate nominatives as described in Haghighi and Klein (2009) and this improved our system by 1.45 F1 points on MUC-6-TRAIN and by 3.9 points on ACE-ROTH-DEV (See Table 6). We next moved on to step vi from Haghighi and Klein (2009), i.e. adding externally mined semantic information to the system. However, in addition to their idea of using semantically compatible head pairs for helping coreference, we experimented by using automatically mined coreference patterns as an additional source of semantic knowledge for the system. Also, we implemented the semantic extraction module without using any knowledge from the data sets on which the system was going to be tested. The rest of this section is devoted to detailed discussion on the semantic module of our system.

4.1 Adding External Semantic Knowledge

As discussed in section 3.1, there is a sizeable fraction of errors (41% of the definite errors) which can be solved only by using some sort of additional world knowledge or pragmatics. In other words, these are cases which cannot be easily detected by using the usual linguistically motivated selectional constraints, but require external world knowledge to establish the coreference. Haghighi and Klein addressed this problem by mining a large set of semantically compatible head pairs from a big corpus like Wikipedia which could then be used during coreference resolution. Examples of such head pairs are (AOL, company), (Gore, activist), (Clinton, president) and so on. The main intuition behind their automatic extraction algorithm was identifying reliable coreference patterns in addition to the ones they already knew, i.e. appositives and predicative nominatives. They took a bootstrapping approach – starting off with two seed patterns (appositives and predicate nominatives) to extract a bunch of seed head pairs; using them to extract more coreference patterns from the corpus; and finally using those extracted patterns to mine even more compatible head pairs. They then used these compatible pairs in their system by deeming two mentions to be directly coreferent whenever their heads were semantically compatible (i.e. present in the list of extracted compatible head pairs).



Sony, **a Japanese company**, is highly …
Sony**, which was** the first company to …
Gigaword corpus
Seed patterns
Apposition: NP, NP
Gigaword corpus
Extracted mention pairs
e.g. (Sony, company)
Extracted patterns
e.g. "A , which is B"
Gigaword corpus
Barcelona, which is the **capital** of Catalonia …
Use the extracted semantically compatible pairs in coreference resolution
Use the extracted semantic patterns in coreference resolution

Figure 1: *Extracting semantic knowledge via a bootstrapping algorithm. For simplicity, the patterns are represented as a string of words, though they are actually syntactic paths.*

The high-level architecture of the bootstrapping system that we adapted from Haghighi and Klein (2009) is shown in Figure 2. We used the apposition construct as our seed pattern to extract an initial set of semantically compatible head pairs which could be used as seeds for the bootstrapping algorithm. We choose not to use the predicate nominative construction as a seed pattern because in our experiments, we found it to be noisier and less reliable than apposition. Also, we chose to use the much larger Gigaword corpus (APW and NYT sections) parsed using the Stanford Parser (Klein and Manning, 2003), instead of the BLIPP and Wiki data sets used by Haghighi and Klein. In our first pass, we went through the parsed Gigaword corpus and extracted the heads of all mention pairs that participate in an appositive construction. For example:

***Sony, a Japanese company**, is highly respected in the world of electronics and its stock is traded on 23 exchanges around the world.*

In the above sentence, we have the appositive pattern "Sony, a Japanese company", from which we extract the mention pair (Sony, a Japanese company). After extracting the heads from the noun phrases, the pair becomes (Sony, company). We similarly extract lots of other head pairs using the appositive pattern, which serve as seeds for extracting more generalized coreference patterns from the corpus. We go through every sentence in Gigaword again, looking for occurrences of any of our seed head pairs within the same sentence. When we do find such a sentence, we extract the pattern connecting the two mentions whose heads were found in our seed pair set. For instance, we may come across a sentence like:

*Sony**, which was** the first company to sign the "Manufacturers' Commitment to Responsible E-Waste Recycling", is highly dedicated to protecting and improving the environment in all areas of its business operations.*

This sentence contains the seed pair (Sony, company) and the matched heads belong to the noun phrases - "Sony" and "the first company". After mapping the matched heads to noun phrases in the sentence's parse tree, we extract the pattern by traversing the path from the first NP to the second one in the parse tree. In this case, the extracted path would be ",/, which/WDT was/VBD", where the labels to the right of the "/" are the pre-terminal nodes (i.e. the Penn Treebank tags) in the parse tree. We lemmatize all the words and collate finer tag types (e.g., all VB* -> VB) in the extracted pattern to deal with sparsity. Thus, the final extracted pattern is ",/, which/WDT be/VB". We then use these parse-path patterns to extract more head pairs from Gigaword. For each sentence in Gigaword, we first identify all the NPs in the parse tree, and then for each possible NP pair, compute the path between them in the same way as we traversed the tree for pattern extraction. If a computed path is present in our set of semantic patterns, we extract the heads of those two NPs and add them to our list of semantically compatible head pairs. Suppose we have another sentence like:

*Barcelona**, which is** the capital of Catalonia, is located on the Mediterranean coast and is the second largest city in Spain.*

The NPs in the parsed sentence are "Barcelona", "the capital of Catalonia", "the capital", "Catalonia", "the Mediterranean coast", "the second largest city in Spain", "the second largest city" and "Spain". While checking for paths between different NP pairs, we would then find that the NPs "Barcelona" and "the capital of Catalonia" are linked by semantic pattern ",/, which/WDT be/VB". We would therefore add the head pair (Barcelona, capital) to our collection of semantically compatible pairs.

Using the above algorithm, we can end up with a large collection of semantically compatible head pairs that can be useful for coreference resolution. Haghighi and Klein only used these final extracted head pairs in their system. Unless one has a very large and diverse (almost web-scale) corpus to mine the head pairs from, the efficacy of such an approach would always depend on how close the domain of the test set is to the domain of the data from which the semantic knowledge was extracted. For instance, if the corpus used for this purpose mainly consisted of newswire stories from the United States, then we might extract pairs like (Obama, president), (Republicans, party), (Gates, chairman), (Woods, golfer), etc. While these are definitely important pieces of semantic information, they might not be of much help if the system is doing coreference resolution on a test set containing news articles from a different country or chapters from a fiction novel, etc. Our intuition was that the coreference patterns that were extracted as part of the bootstrapping algorithm could be a more generic source of semantic knowledge than the head pairs themselves. Patterns like "which/WDT be/VB" or "who/WP be/VB" or "know/VB as/IN" can reliably indicate coreference in text, irrespective of the domain and it therefore made sense to include them in the system (see figure 2) rather than discarding them after mining the second set of head pairs as Haghighi and Klein did. To directly use them for coreference, the procedure is almost the same as pattern matching to collect new head pairs, with the difference that we now look for paths between NPs directly in the test set and deem NPs to be coreferent whenever we find a connecting path present in our pattern set.

The bootstrapping approach described here, though intuitive to understand, is not that easy to implement correctly in practice. Moreover, Haghighi and Klein's explanation of the algorithm left a lot of leeway in interpretation. We next discuss some of the challenges we faced while implementing this bootstrapping algorithm for mining semantic knowledge.

4.2 Seed selection - As you sow, so shall you reap

In a bootstrapping paradigm, the final outcome of the algorithm is highly dependent on the quality of the initial seeds. Haghighi and Klein chose two highly reliable patterns for coreference – appositives and predicate nominatives, to extract the initial seed set of head pairs from the data. In order to control noise and ensure good quality seeds, we took an even more conservative approach and used only the appositive pattern to mine the initial head pairs from the Gigaword corpus. The system did extract some useful pairs like (Turkey, country), (Democrats, party) and (capital, Belgrade), but it also ended up extracting a considerable number of noisy head pairs. As expected, when these head pairs were used for extracting new coreference patterns from the corpus, the system extracted a lot of useless patterns and it did not make sense to continue to the next stage of bootstrapping till we ensured good quality output at the first stage. The presence of noise in the initial seeds was handicapping the entire process and it was clear that some sort of post-processing was needed to filter the head pairs extracted by the apposition detector.

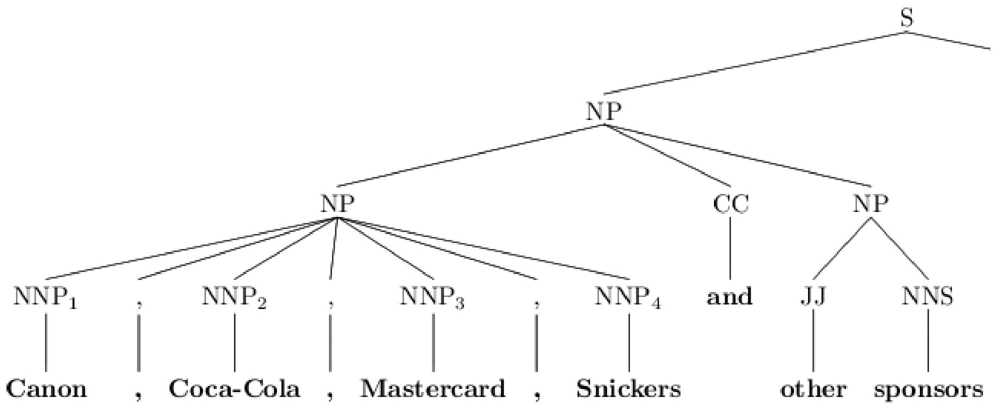
We first tried using a cutoff on the frequency of extracted head pairs as a means to prune the seed pairs. However, this approach did not work because a lot of the noisy pairs themselves turned out to be high frequency ones. In fact, when ordered by decreasing frequency, the mention pair (Serbia, republic), ranked 10th, was the only pair that seemed sensible among the top ten most frequent pairs.

A lot of the noisy head pairs did not seem to be appositives at all, which prompted us to go back to our apposition detection implementation and try to improve the detection heuristic. While doing apposition detection, to ensure that we do not label elements in a list as appositives, we had followed the same heuristic as Haghighi and Klein (2009) of checking for "CC" nodes in the tree of the parent NP. In spite of this, one of the common errors was still the false detection of items in a list as appositive constructions. These were often cases of long comma-separated lists or complex sentences which got parsed wrongly by the Stanford parser. For example:

Sentence:

Canon, Coca-Cola, Mastercard, Snickers and other sponsors are happily hawking their image and their soccer connection to global NNS audiences.

Parse tree:



Here, the parent NP of the nodes marked $NNP_1 – NNP_4$ does not contain a “CC” node and hence the list check would fail here. We would therefore end up getting incorrect head pairs like (Canon, Coca-Cola), (Coca-Cola, Mastercard) and (Mastercard, Snickers) as appositives. We addressed this by adding the constraint that parent NP node should not have more than two “comma” nodes as children, because having three or more commas highly increases the chances of the entities being part of a list.

Another trend that we noticed in our extracted head pairs was that majority of the good pairs seemed to be (proper noun, common noun) or (common noun, proper noun) pairs. On the other hand, the same noun type pairs, i.e. (common noun, common noun) or (proper noun, proper noun) seemed to be much more noisy and contributed to most of the errors. Table 1 contains some example pairs from the NYT section of Gigaword.

Good seed pairs	*Bad seed pairs*
(Common Noun, Proper Noun)	***(Common noun, Common noun)***
(authority, Caltrans)	*(gauge, writers)*
(commissioner, Kaminskas)	*(help, victory)*
(then-president, Yeltsin)	*(beauty, lure)*
(campaigner, Khachigian)	*(museums, novels)*
(engine, google.com)	*(opinions, some)*
(Proper Noun, Common Noun)	***(Proper noun, Proper noun)***
(Hackerson, pilot)	*(Ontario, U.S.)*
(El-Dumeiri, minister)	*(Ryan, Sindelar)*
(Cavonnier, horse)	*(Dornemann, Morgado)*
(Finley, quarterback)	*(Gorbachev, Yeltsin)*
(Stanford, university)	*(Francisco, London)*

Table 1: *Examples of seed head pairs from the Gigaword corpus during one of our initial runs*

To increase the precision of the seeds, we included a strong constraint that the system should accept only (common noun, proper noun) or (proper noun, common noun) patterns. This constraint also helped us fix another frequent source of error (also mentioned by Haghighi and Klein), which was the false detection of location names like “Santiago, Chile” as appositives.

The heuristics so far helped by a decent amount in decreasing the noise in our seed pairs, but there were still some bad pairs that needed to be filtered out. These were mostly cases of problematic tokenization (e.g. ’s, Humphrey), dates (e.g. March, 12), company suffixes (e.g. WorldSources, Inc.), name suffixes (e.g. George, V) and units of measurement (e.g. 5 feet, 2 inches). We filtered these pairs out by using a gazetteer of month names, day names, common suffixes, units of measurement, etc. and using a few additional rules to remove some obvious errors (see Table 2). This led to a much cleaner seed set with most of the high frequency head pairs (8 out of the top 10) being sensible semantically compatible mention pairs. However, 22 out of the top 50 (by frequency) pairs were instances of the kind (City name, capital). We would ideally have wanted to have a more diverse seed pair set, but it turned out that a lot of the high frequency head pairs were quite similar to each other. This could adversely affect the next stage of the bootstrapping algorithm, since we might not be able to extract a wide variety of patterns.

Due to the hard constraint that we had put in place earlier, we were missing out on a lot of common noun head pairs which could have potentially been useful for coreference and have also increased the diversity of the seed set. Among all the noisy pairs, the system had also managed to find some very good (common noun, common noun) pairs such as:

1. *(army, combatants)*	5. *(coat, dress)*
2. *(crime, shootings)*	6. *(recruits, soldiers)*
3. *(cars, vehicles)*	7. *(defendants, lawyers)*
4. *(organization, group)*	8. *(disease, illness)*

We therefore wanted to relax our initial constraint and allow common noun head pairs to be extracted, but at the same time ensure high quality seeds. We approached the problem of filtering the common noun seed pairs in two ways:

- **Using WordNet**

In our first approach, we used the WordNet lexical database to estimate the quality of our candidate common noun head pairs. The distance between the two extracted mentions in the WordNet IS-A (hypernym/hyponym) hierarchy was used as a measure of closeness between them. We allowed only those (common noun, common noun) pairs to be extracted by our system which were related to each other through a IS-A hierarchy of at most depth "k". The intuition behind using the IS-A hierarchy was that hypernyms often corefer with their hyponyms. E.g. "cassette" and "videocassette", as discussed in section 3.6. In addition, we also used a depth cutoff (k), because the number of intervening levels in the hierarchy is an indicator of how generic the hypernym is:

```
videocassette
   => cassette
     => container
       => instrumentality, instrumentation
         => artifact, artefact
           => whole, unit
             => object, physical object
               => physical entity
                 => entity
```

Thus, "cassette" which is just one level above "videocassette" is a more reliable candidate for coreference than something like "artifact" (4 levels above) which is a more generic term that could potentially corefer with lot of different hyponymous nouns. We experimented with different cutoffs for depth. While k = 1 and k = 2 gave us high quality matches, the constraint was too tight and ended up rejecting a lot of good pairs. With higher values of k, we were able to extract larger set of pairs from Gigaword, but this also increased the chances of extracting far too generic pairs which were just as bad as having completely noisy pairs. k = 3 seemed to provide us the best set in terms of coverage vs. noise trade-off and we decided to include those (common noun, common noun) head pairs in our seed set for bootstrapping.

- **Using Dekang Lin's proximity-based Thesaurus**

Dekang Lin's proximity-based thesaurus is a lexical resource which contains entries for most of the words in the English vocabulary. Each thesaurus entry has an English word followed by its 200 "closest" words. The measure of "closeness" is distributional similarity and each similar word gets a similarity score representing how close or similar that word is to the main word. For example, the entry for "videocassette" in the thesaurus is as follows:

videocassette:

1. *cassette 0.342675*
2. *C_VCRs 0.276092*
3. *C_VCR 0.265561*
4. *radio-cassette 0.228062*
5. *C_VHS 0.227428*
6. *videotape 0.226275*
7. *C_Dvd 0.223968*
8. *camcorders 0.211434*
9. *video 0.195067*
10. *audiotape 0.18922*
11. *C_DAT 0.188172*
12. *home-video 0.181384*
13. *videocassettes 0.178622*
14. *…*

Similar to our heuristic based on WordNet, we allowed our system to extract common noun pairs only if one of the nouns was present within the top "k" words in a thesaurus entry of the other noun. For instance in the above example, if k = 10, (cassette, videocassette) or (audiocassette, videocassette) would have been chosen as valid pairs, but not (C_DAT, videocassette). Thus, the cut off "k" gives us a way to control the quality of extracted seeds since the later words in a thesaurus entry are more likely to be noisy. However, we found that entries for some words in the thesaurus just did not have many valid similar words. As a result, even the words within the top ten or twenty in the similarity list seemed like bad matches. But it was also the case that such words (in spite of being ranked high in the list) would have very low similarity scores, much less than the similarity score for the first-ranked word. We therefore imposed an additional constraint, that the distributional similarity score should at least be "p%" of the similarity score for the topmost word in that entry.

1. Discard if either of the NPs is
   a. a number
   b. a measurement unit, e.g., kg, mm, inch., yard, etc.
   c. part of a date/time, e.g., mon-fri, jan-dec, etc.
   d. a person name suffix, e.g., Jr., Sr., III, etc.
   e. an organizational suffix, e.g. Ltd., Inc., LLC., etc.
   f. an ordinal, e.g. 48th, 21st, etc.
2. Discard if the head of either NP is "**'s**" (a common head detection error for our parser)
3. Discard if either NP contains an underscore "_", e.g., fouls_alabama, finals_1, etc.
4. Discard if the parent NP has more than 2 commas (e.g. parent NP expanding to NP1, NP2, NP3, NP4)
5. Accept only if both mentions are nouns, i.e. the POS tag starts with "NN"
6. Only accept (common noun, proper noun) or (proper noun, common noun) combinations
7. Accept pairs in exception to rule 6., iff
   a. The two mentions are related to each other via a IS-A (hypernymy or hyponymy) relationship in WordNet.
      (OR)
   b. The two mentions have a high distributional similarity score in Dekang Lin's proximity based thesaurus.

Table 2*: Heuristics used in seed pair selection*

We experimented with different "k's" and "p's", but none of the settings were good enough to beat the score we got by using the seeds extracted using the WordNet heuristic. Our most optimal setting was k = 20, p = 10. Apart from looking at the quality of the patterns extracted by using these seeds in the next phase of the bootstrapping algorithm, we also evaluated seed quality by directly adding them as semantically compatible mention pairs in our coreference system. However, using the seed pairs extracted by distributional similarity heuristics was at best only able to match the WordNet-based system in recall, but not in precision.

We also tried a combination of the above two approaches - (i) Using the mention pairs resulting from doing an intersection between all the mention pairs found using the WordNet heuristic and those found using the distributional similarity heuristic (ii) Using the mention pairs from the union of the two approaches. The former was just as good as using WordNet alone, whereas the latter performed worse than WordNet alone, since it had a much lower precision. A summary of the final heuristics used in seed pair selection is outlined in Table 2. After applying these heuristics, almost all of our high frequency pairs were good quality seed pairs, with the low quality ones having very low frequencies of occurrence. We could now therefore successfully impose a frequency cutoff for pruning our set of seed pairs and weed out the low frequency noisy pairs. In practice, we found that keeping the cutoff (for number of occurrences) at 10 or 5 worked almost equally well, with the experiment where cutoff = 5 doing just marginally better (because of slightly higher recall).

However, in spite of putting in a considerable amount of effort in coming up with clever filtering techniques to improve the seed extraction process, we were still extracting a lot of noisy patterns while doing pattern extraction on Gigaword. We reasoned that the high amount of noise in the extracted patterns was due to the relatively large size (in thousands) of the seed set which could have contained some noisy pairs that might have crept in despite our filtering attempts. We therefore chose to take an approach that traditionally been used to good effect in bootstrapping algorithms – that of manually selecting a small number of high quality seeds. So we manually went through our extracted seed pairs and handpicked about hundred head pairs which were diverse and of very high quality (see Appendix A). This helped in decreasing noise in pattern extraction to quite an extent, but we still needed better filtering and ranking techniques for the extracted patterns before they could be used in the final bootstrapping step to extract more compatible mention pairs.

4.3 Pattern Selection

We require patterns that can determine coreference with very high accuracy, because noisy patterns can end up extracting a lot of semantically incompatible mention pairs from the corpus. These head pairs would then adversely affect the system's performance when used as semantic knowledge for coreference resolution. We therefore had to study the extracted patterns carefully and devise various heuristics to filter out the noisy patterns just like we had done for the seed head pairs. We found that patterns that are too long hardly have coreference relations, so we removed patterns longer than 10 (a cut-off also used by Haghighi and Klein). We also filtered out all the empty patterns, which were extracted from cases of role appositives like "President Obama", "Senator McCain", etc.

Also, since the corpus we used was from newswire, we found a lot of instances of the verb 'say' and other reporting verbs like "agree", "report", "debate", etc. across over our pattern set. We therefore discarded any pattern containing a reporting verb. In addition to this filter, we removed any pattern containing a noun because almost all such patterns were incorrect coreference patterns. Finally, we used Haghighi and Klein's constraint on pair coverage (number of unique seed pairs that the pattern occurred with) to remove all patterns that had a coverage less than 100.

Ranking Method	Description
(1) $Hits$	The number of times the pattern was found with a seed head pair
(2) $Word\ Coverage$	The number of unique words from the seed pair set that were covered by this pattern
(3) $Pair\ Coverage$	The number of unique pairs from the seed pair set that were covered by this pattern

(4) $Precision$	The ratio of “good” occurrences to total occurrences, i.e. the number of times the pattern occurred with pairs from the seed set, divided by the number of it occurred with any head pair in the corpus
(5) $Precision \times \log(Hits)$	See (4), (1)
(6) $Precision \times Pair\ Coverage^2$	See (4), (3)

***Table 3**: Different ways to rank the extracted patterns*

Though Haghighi and Klein do not mention any additional thresholds or filtering techniques, we found that using only their pair coverage threshold of 100 did not work well in practice. First of all, we lost a lot of good patterns due to this rather strict threshold. Secondly, despite this constraint, many short patterns (of length = 1) still survived. If a pattern appeared a lot, the total number of occurrences with seed pairs wasn’t useful to filter them out. For example, a pattern like “of/IN” occurred very frequently with various pairs, and wasn’t discarded by the above constraints. We therefore had to devise a better ranking mechanism for patterns than just pair coverage. Table 3 lists the various ranking methods we experimented with. The extracted patterns were ranked using these six metrics and three independent judges rated the quality of the resulting six lists to decide which metric distinguished between the high confidence and low confidence patterns the most. The list of patterns ranked by word coverage and precision seemed to be more promising than the other four. We finally set a cut off of 5 for word coverage and 0.005 for precision and only used those patterns that satisfied both these constraints. Table 4 lists our final set of heuristics used in pattern selection.

Discard the pattern if
1. The path contains an NN* node (i.e. a noun)
2. The path contains a reporting verb (like “say”, “tell, “announce”, etc.)
3. It is empty (path length = 0) or has path length greater than 10.
4. It has a word coverage less than 5
5. It has precision less than 0.005

***Table 4**: Heuristics used in pattern selection*

With the above heuristics, we were able to extract highly-reliable patterns that represented coreference. Table 5 shows some of the useful patterns that we extracted.

X ,/, **Y**	**X** have/VB become/VB **Y**
X which/WDT be/VB **Y**	**X** identify/VB as/IN **Y**
X know/VB as/IN **Y**	**X** describe/VB as/IN **Y**
X who/WP be/VB **Y**	**X** who/WP serve/VB as/IN **Y**
X be/VB consider/VB **Y**	**X** be/VB believe/VB to/TO be/VB **Y**

***Table 5**: Examples of extracted pattern paths; All words are lemmatized. X and Y represent the coreferent NPs.*

Unlike Haghighi and Klein, we wanted to use these patterns not only for extracting more head pairs in the next iteration of bootstrapping, but also for direct use in our coreference system. However, we found that the same set of patterns is not well suited for both tasks. The set of patterns used for bootstrapping can afford to be slightly less precise and more recall-oriented, because the goal here is to harvest as many compatible head pairs as possible from the corpus. Even if the patterns do extract a lot of noisy pairs, we can use the heuristics in Table 2 to prune the space and in addition, find ways to rank the various head pairs and select a suitable threshold to remove the noisy pairs. However, when we used the same set of patterns directly in the coreference system, the noisy patterns matched a lot of false positive heads and gave us a big drop in the F1 score. It was thus necessary to ensure that the set of patterns directly used in the system were as precise as possible. We experimented by tightening the thresholds for word coverage and precision by various amounts, with the most extreme case being the one where we actually handpicked the patterns to use for coreference. However, we could not find any satisfactory setting that helped in improving coreference resolution by a substantial amount.

In the case of handpicked patterns or very precise patterns (i.e. high thresholds for word coverage and precision), the system hardly found any pattern matches in the dev/test sets because of the relatively small size of the pattern set being used. On the other hand, when we loosened the thresholds and used a larger pattern set, the system ended up matching a lot of false positives which negated the effect of increasing recall by lowering the system's precision. Using a large set of patterns surely seems to be the way forward, but clearly more work needs to be done to ensure better quality patterns and to add checks in the pattern matching phase so as to discard incompatible coreference proposals.

4.4 The Final Step - Extracting more head pairs

In the final step, we use the pruned set of patterns (see heuristics for pruning in Table 4) to extract more head pairs from the parsed Gigaword corpus. We used the same set of heuristics that we used for seed pair pruning (Table 2) to filter out noisy head pairs. In addition, we did a ranking exercise similar to the one we had done for patterns. Table 6 describes the different ranking mechanisms that we tried out.

Ranking Method	Description
(1) $Hits$	The number of times the head pair was extracted by one of our patterns
(2) $Coverage$	The number of unique patterns that extracted the head pair
(3) $Precision$	The ratio of "good" occurrences to total occurrences, i.e. the number of times the head pair was extracted by patterns from our set, divided by the total number of times the heads co-occurred in a sentence (in any syntactic configuration)
(4) $Average\ Pattern\ Hits$	The average hit score (Table 4, (1)) of all patterns that extracted this head pair
(5) $Average\ Pattern\ Word\ Coverage$	The average word coverage score (Table 4, (2)) of all patterns that extracted this head pair
(6) $Average\ Pattern\ Pair\ Coverage$	The average pair coverage score (Table 4, (3)) of all patterns that extracted this head pair
(7) $Average\ Pattern\ Precision$	The average precision score (Table 4, (4)) of all patterns that extracted this head pair

Table 6: *Different ways to rank the extracted head pairs*

Similar to our patterns experiment, we found that the list of pairs ranked using coverage (2) and precision (3) were better than the others. However we could still not find any satisfactory setting (threshold on precision and coverage) such that the resulting head pairs helped the coreference system. Using any amount of these semantically compatible pairs (see Table 7) hurt the system's performance. Qualitatively, we found that we got the best set of pairs with a precision cutoff of 0.1 and coverage cutoff of about 10. Even though most of the pairs in that set seemed to be legitimate coreferent heads, the F1 on both MUC-6-TRAIN and ACE-ROTH-DEV still dropped below the F1 for the system without any semantic knowledge.

1. *(opposition, party)*	7. *(capital, city)*
2. *(Democrats, party)*	8. *(association, group)*
3. *(Bush, president)*	9. *(members, people)*
4. *(association, organization)*	10. *(college, university)*
5. *(state, Texas)*	11. *(bill, legislation)*
6. *(squad, team)*	12. *(agreement, deal)*

Table 7: *Examples of the final head pairs extracted by our system*

Error analysis showed that apart from the expected errors caused due to noisy head pairs, a lot of the errors were in fact caused by head pairs which were actually good semantically compatible heads. There were two main reasons for such errors –

(i) Not taking the global information into account.

(ii) Checking the mentions head for semantic compatibility in isolation from its modifiers or surrounding context.

The two reasons are closely related to each other. Consider this example:

Gillette South Africa will sell manufacturing facilities in Springs, South Africa, and its business in toiletries and plastic bags to Twins Pharmaceuticals Ltd., an affiliate of ***Anglo American Corp****.,* ***a South African company****.* ***Gil-lette South Africa*** *employs about 250 people.*

Suppose that "Gillette South Africa" (the one that has been highlighted) is the mention under question. Before this point, the system would have already deemed the mention "a South African company" to be coreferent with the mention "Anglo American Corp." because the former noun phrase is an appositive of the latter. However, since the system might have the head pair (Gillette, company) in its set of semantically compatible heads, it will now mark "Gillette South Africa" to be coreferent with "a South African company". In other words, "Gillette South Africa" ends up being incorrectly included in the same coreference chain as "Anglo American Corp.". This mistake was committed by the system since it always makes local pairwise decisions without using information about other mentions in the documents or the previous coreference decisions that it made itself. Consider another example:

This summer, ***JVC*** *launched an ad campaign emphasizing the VHS-C's compatibility advantage over 8mm, while* ***its sister company****, Panasonic , recently cut list prices $100 to $200 on some of its VHS-C models, to as low as $ 799.*

The mention under question is the noun phrase "its sister company" and the system would choose "JVC" as the coreferent antecedent for this mention since "JVC" and "company" form a semantically compatible head pair. This error was committed since the system does not check the head modifier ("sister" in this case) at all. It might help to have a blacklist of modifiers, e.g. sister, parent, rival, etc. which discourage the system from using a semantic head match when they are present in the mention. Thus, making better use of modifiers and checking for previously proposed antecedents of mentions before declaring them to be coreferent, seem to be two good ways to improve the performance of the system.

4.5 Experimental Results

We used MUC-6-TRAIN and ACE-ROTH-DEV as development sets while implementing the original Haghighi and Klein (2009) system and experimenting with different approaches towards a generalized semantic compatibility module. Table 6 shows the system performance with and without apposition and predicate nominative detection for both the development sets and the MUC-6-TEST test set. Results from our semantic knowledge module are not included, since the numbers highly vary depending on the number of patterns and head pairs being used. In any case, even the best performing system with semantic knowledge was unable to beat the system with only appositives and predicate nominatives.

We get a pairwise F1 score of **65.96** on MUC-6-TEST, whereas Haghighi and Klein (2009) got an F1 of **67.3** on the test set. Our system used no semantic information and was still within 1.3 F1 points of their system. However, our system relied on knowing the gold NE types, in contrast to Haghighi and Klein, who use the Stanford NER to annotate mentions in the test set. In the next section, we remove our system's dependency on gold NE information and discuss methods to make the system competitive despite the lack of any externally mined semantic knowledge.

System	**MUC-6-TRAIN**		
	Pairwise Precision	Pairwise Recall	*Pairwise F1*
Initial System (Agreement and Syntactic constraints)	59.20	49.03	*53.64*
+ Appositive and Predicate Nominatives detection	58.53	52.04	*55.09*
	ACE-ROTH-DEV		
	Pairwise Precision	Pairwise Recall	*Pairwise F1*
Initial System (Agreement and Syntactic constraints)	59.2	41.58	*48.85*
+ Appositive and Predicate Nominatives detection	59.36	47.46	*52.75*
	MUC-6-TEST		
	Pairwise Precision	Pairwise Recall	*Pairwise F1*
Initial System (Agreement and Syntactic constraints)	77.93	55.98	*65.16*
+ Appositive and Predicate Nominatives detection	76.76	57.84	*65.96*

Table 6*: Evaluation results on the various data sets. Note that the system uses gold NE information.*

5 Improving the Haghighi and Klein System

The entire exercise of implementing Haghighi and Klein's coreference resolution system and analyzing the system's errors at each stage was a very important one. It helped us in discovering the various error patterns, the reasons behind them, the limitations of the current system, and gave us insights into possible ways of tackling some of the errors. In this section, we first talk about the improvements that we were able to accomplish over the Haghighi and Klein (2009) system and end with the description of a couple of ongoing experiments that show promise for further improvements.

5.1 Acronym Detection

We noticed that there were a few errors caused due to acronyms which could be fixed by using a fairly high precision heuristic. For instance, consider:

A company spokesman said that talks with the ***Communications Workers of America*** $_1$ *and the* ***International Brotherhood of Electrical Workers*** $_2$ *broke down last weekend. Union spokesmen have said that talks are stalled over a company plan to shift some health-care costs to employees. The* ***CWA*** $_1$ *represents about 40,000 Nynex employees while the* ***IBEW*** $_2$ *represents 20,000.*

In the above example, the system could have easily found coreference between "Communications Workers of America" and "CWA" had it known that one noun phrase was merely the expansion of the other. For some of the popular company names and abbreviations, this information could have potentially come from Haghighi and Klein's semantically compatible mention pairs, but as discussed earlier, it was tough to get the semantic module working correctly. Even if it did work, it is likely that the some of the acronyms might not be present in the list of mined head pairs. So we implemented a fairly straightforward heuristic for acronym detection.

Whenever we have a multiword mention containing more than one capitalized word, we create an acronym out of this phrase by extracting each of the uppercase letters and concatenating them together. Since we only choose the uppercase letters, this method automatically leaves out stop words like "of" which are usually not capitalized. Thus in the above example, "Communication Workers of America" and "International Brotherhood of Electrical Workers" would become "CWA" and "IBEW" respectively. Our system then directly deems two mentions to be coreferent if the acronymized version of one of them matches the other. The heuristic worked and fixed some of our errors and Table 7 summarizes the improvements after adding this heuristic.

Data Set	**Before**			**After**		
	Pairwise Precision	Pairwise Recall	*Pairwise F1*	Pairwise Precision	Pairwise Recall	*Pairwise F1*
MUC-6-TEST	76.76	57.84	*65.6*	76.55	58.64	*66.41*
ACE-ROTH-DEV	59.36	47.46	*52.75*	59.6	47.48	*52.76*

***Table 7**: Improvements from Acronym Detection*

5.2 Relative Pronouns

A relative pronoun links two clauses (a main clause and a relative clause) together to create a single complex clause. The relative pronoun in such cases corefers with noun phrase in the main clause. For example:

*Among other winners Wednesday was **Nippon Shokubai**, <u>**which**</u> was up 80 at 2,410.*

Here, the relative pronoun "which" refers to the NP "Nippon Shokubai". However, our current system did not find this coreference since the two mentions violate the i-within-i constraint. Hence, just like with appositives and predicate nominatives, we added relative pronouns too as an exception to the i-within-i constraint and deemed such constructs as being coreferent, whenever the presence of a relative pronoun was detected. Table 8 shows our system's improvements after adding the deterministic detection of relative pronouns.

Data Set	**Before**			**After**		
	Pairwise Precision	Pairwise Recall	*Pairwise F1*	Pairwise Precision	Pairwise Recall	*Pairwise F1*
MUC-6-TEST	76.55	58.64	*66.41*	76.55	58.64	*66.41*
ACE-ROTH-DEV	59.36	47.48	*52.76*	59.13	49.1	*53.65*

***Table 8**: Improvements from detection of relative pronouns*

The improvements can only be seen on the ACE corpus because hardly any relative pronouns have been annotated in the MUC-6 corpora (see Table 9). But this was a still a good overall enhancement to the system.

	MUC-6-TEST (2,141 mentions)	ACE-ROTH-DEV (4,536 mentions)
who	0	109
whom	0	0
whose	0	12
which	0	46
that	3	30

***Table 9**: Number of times the pronouns "who", "whom", "whose", "which" and "that" are annotated as mentions in the various data sets. These numbers are just for rough comparison purposes – they may also include occurrences of these words when they are not actually functioning as relative pronouns.*

5.3 Better Attribute Assignment

Once the system cannot make a direct coreference decision based on appositions, predicate nominatives, or acronyms, it resorts to agreement constraints for filtering the possible set of antecedents for a given mention. The system checks for the agreement of four attributes – animacy, number, gender, and entity type. Here, we talk about the first three in particular. These attributes were initially assigned for each mention using simple heuristics based on POS tags and predefined lists of nouns and pronouns. However, our existing heuristics turned out to be quite ineffective in assigning a correct animacy, number or gender attribute to the majority of the mentions in the data set. As a result, we had a large number of mentions in our data set which had attributes marked as "UNKNOWN", since the system was not able to correctly predict the animacy, number or gender for the mention. At the time of coreference resolution, such "UNKNOWN" attributes were used as wildcards and could match any attribute value. For instance, a mention having "UNKNOWN" animacy could match with both an animate and an inanimate antecedent. On the other hand, had the animacy attribute been assigned a definite value, say "ANIMATE", then we could have filtered out all the inanimate antecedents and decreased the chances of making an incorrect coreference decision. Unknown attributes thus greatly reduce the efficacy of agreement constraints-based filtering. We therefore decided to leverage existing research in animacy/number/gender detection to improve attribute assignment in our system.

For improving animacy detection, we used a noun-animacy table prepared by Heng Ji and Dekang Lin (2009) which contains a large list of nouns along with their predicted animacies. For improving number and gender detection, we used similar tables prepared by Shane Bergsma and Dekang Lin (2006). Table 10 shows the number of mentions that benefited from using these tables, i.e. these are mentions for which the attribute was "UNKNOWN" earlier, but now had an assigned value.

Data set	Total Mentions	Mentions with new *animacy* assignments	Mentions with new *gender* assignments	Mentions with new *number* assignments
MUC-6-TEST	2,141	1,083	1,546	407
ACE-ROTH-DEV	4,536	1,950	3,341	588

***Table 10**: Number of mentions that benefitted from better attribute assignments*

It is important to note that in contrast to the handcrafted high precision rules that we used earlier for attribute assignments, the attributes assigned using the Ji-Lin or the Bergsma-Lin tables were subject to noise, since these tables had been constructed in an automated fashion. Our hope was that the benefits gained from having a proper attribute assignment for a large number of mentions would outweigh the problems caused by incorrect attribute assignment for a small fraction of the cases. Table 11 shows the system performance after using enhanced attribute assignments.

Data Set	**Before**			**Better Animacy**			**Better Animacy, Gender & Number**		
	P	R	*F1*	P	R	*F1*	P	R	*F1*
MUC-6-TEST	76.6	58.6	*66.4*	77.6	59.4	***67.2***	75.9	59.2	*66.5*
ACE-ROTH-DEV	59.1	49.1	*53.7*	62.6	51.3	*56.4*	63.5	51.4	***56.8***

***Table 11**: Improvements from better attribute assignment*

We got the biggest boost from better animacy detection. A lot of the errors due to animacy mismatch (discussed in section 3.7) were fixed after using the Ji-Lin animacy table. Here are a couple of examples of precision errors that our earlier system used to make, but were fixed in the new system:

__The president of the Association of Professional Flight Attendants__, which represents American's more than 10,000 flight attendants, called the request for mediation "premature" and characterized __it__ as a bargaining tactic that could lead to a lockout.

	Mention	Antecedent	Animacy Compatible?	Linked as coreferent?
Earlier:	president (UNKNOWN)	it (INANIMATE)	Yes	Yes
Now:	president (ANIMATE)	it (INANIMATE)	No	No

If __harsh things__ were said by either side in the heat of the 1985 pilot strike, or since, __we__ regret them, and want to forget them.

	Mention	Antecedent	Animacy Compatible?	Linked as coreferent?
Earlier:	things (UNKNOWN)	we (ANIMATE)	Yes	Yes
Now:	things (INANIMATE)	we (ANIMATE)	No	No

However, improving number and gender assignment using the Bergsma-Lin table had mixed results. The system performance increased by 0.4 F1 points (as compared to the increase of 2.7 while using animacy alone) on ACE-ROTH-DEV and decreased by 0.7 F1 points on MUC-6-TEST. There are several reasons for the number and gender enhancements not working as effectively as the animacy enhancement. First of all, we found that the gender and number tables provided by Bergsma and Lin (2006) are noisier compared to the animacy data from Ji and Lin (2009). Also, the gain in information from using number and gender was quite less as compared to knowing the animacy of the mention. Number helped in some cases, but gender was almost never the distinguishing factor between mentions, especially in MUC-6-TEST. The few cases where it seemed like gender could have helped were cases of distinguishing male or female gender from neutral gender (assigned to words that are organizations, locations or other things). However, the system was already getting this information from the animacy attribute (animate = male or female, inanimate = neutral).

Lastly, there were errors committed by the system since it only looked at the head word for attribute assignment, which at times misled it to wrongly assign attributes for the mention. The fraction of such errors happened to be higher in MUC-6-TEST. A particularly good example is a document about "American Airlines" in MUC-6-TEST. Consider the headline for this document:

__American Airlines__ Calls for Mediation In __Its__ Union Talks: By Marj Charlier

	Mention	Antecedent	Number Compatible?	Linked as coreferent?
Earlier:	American Airlines (UNKNOWN	Its (SINGULAR)	Yes	Yes
Now:	American Airlines (PLURAL)	Its (SINGULAR)	No	No

The mistake was committed since the head word "Airlines" is plural according to the Bergsma-Lin list. Since this document was about American Airlines, it was filled with such kind of pronominal references (it, its) to the mention "American Airlines" and the system got each one of them wrong. Each such recall error (missing the link "it/its" -> "American Airlines") was accompanied by a precision error (i.e. system incorrectly linking the "it" or "its" to some other mention), which further contributed to the low performance. Thus, there is certainly room for improvement, but proper attribute assignment for all the mentions in the test set seems to be a good way to improve coreference resolution. In our final system we decided to only use the animacy detection enhancement.

5.4 Attribute Sharing

Section 5.3 showed that the system performance increases considerably after filling in some of the previously unknown attributes with their correct values. Having correct attributes for mentions constraints the list of possible antecedents to ones that are more likely to be truly coreferent. Though we improved attribute assignment a lot by the methods described earlier, there were still a considerable fraction of mentions with one or more unknown attributes. These were words that were not a part of the Ji-Lin or the Bergsma-Lin vocabulary, e.g. rare proper nouns that only occurred in a particular domain. Also, most of our mentions did not have an assigned entity type. Only the named entities had a NE type assigned using the Stanford NER, but neither of the common nouns or pronouns had any such information. We therefore resorted to other methods to decrease the number of "UNKNOWN" attributes.

We used a key insight that we had got from our earlier error analysis – to make use of previous coreference decisions, instead of making isolated pairwise decisions. Even though a given mention might have some of its attributes missing (i.e. "UNKNOWN"), it might be the case that one of the other mentions in its coreference chain (i.e. mentions it has already been found coreferent to) has that information. Thus, whenever we found two mentions to be coreferent, we merged the attribute sets of those two mentions. Consider the following example:

*By proposing a meeting date, **Eastern** moved one step closer toward reopening current high-cost contract agreements with **its** unions.*

Initial attribute set:

Mention	Entity Type	Animacy	Number	Gender
Eastern	ORGANIZATION	INANIMATE	UNKNOWN	UNKNOWN
its	UNKNOWN	INANIMATE	SINGULAR	UNKNOWN

Sharing attributes after coreference has been established:

Mention	Entity Type	Animacy	Number	Gender
Eastern	ORGANIZATION	INANIMATE	SINGULAR	UNKNOWN
its	ORGANIZATION	INANIMATE	SINGULAR	UNKNOWN

Thus, by propagating the set of attributes from one coreferent mention to the other, we were able to fill two of the previously empty attributes (one for each mention). In this particular case, the pronominal mention "its" could have earlier acted as a suitable antecedent for a singular inanimate mention with any entity type – Organization, Date, Money, Location, etc. But now, it is constrained to be coreferent with only those mentions that are organizations (or have entity type UNKNOWN). While doing attribute sharing between two mentions that have been deemed coreferent, there were a few cases when one or more of attributes conflicted each other. In such cases, we were conservative and set the attribute value of both mentions to be a union of the two different values. Such a mention was then allowed to be coreferent with mention of both those attribute types. For instance, a mention having the gender value as "MALE or NEUTRAL" could corefer with both male and neutral gender mentions. Table 12 shows the performance of the system with attribute sharing.

Data Set	**Before**			**After**		
	Pairwise Precision	Pairwise Recall	*Pairwise F1*	Pairwise Precision	Pairwise Recall	*Pairwise F1*
MUC-6-TEST	77.56	59.35	*67.24*	80.33	59.64	***68.45***
ACE-ROTH-DEV	62.55	51.33	*56.38*	65.37	50.69	***57.1***

***Table 12**: Improvements from attribute sharing*

5.5 Discourse Salience

We next tried a simplistic way to incorporate discourse salience into the coreference resolution algorithm. Our aim was to detect mentions that are mostly likely to have been just introduced into the discourse, and thus not have any antecedents. The idea was to make the system skip such mentions instead of incorrectly predicting antecedents for them and causing precision errors. We used three rules for detecting such mentions:

1. Checking if the mention begins with an indefinite article – “a” or “an”. E.g., “**A spokesman** for the company said that …”
2. Checking if the mention is an indefinite pronoun[2] like “some”, “another”, “such”, “several”, “other”, etc. E.g., “**Some** say that …”
3. Checking if the mention begins with an indefinite adjective, i.e. an indefinite pronoun used as an adjective before a noun. E.g., “**Another opinion** on the matter is …”

If a mention satisfied any of the above three conditions, it was highly likely that the entity had just been introduced into the conversation or was a general pronoun which did not corefer with anything earlier in the text. However, using rules 1 and 3 did not work well in practice. We especially found a lot of exceptions to rule 1 (See Table 13). In the case of rule 3, it turned out that not all indefinite adjectives were equally reliable for predicting whether the mention does not have an antecedent.

1. "as-a" pattern:
*Sony conceives of **8mm** as **a “family of products, camcorders and VCR decks,”** whose sales will reinforce one another.*

2. appositives:
*Sony this month will introduce **a full-featured, $ 1,500 palm-sized 8mm model**, **the CCD-TR5 Handycam**, tipping the scales at less than two pounds.*

3. "that-is-a" pattern:
*“It's the proverbial chicken and egg situation,” says George Krieger, president and chief executive officer of **CBS Home Video, a leading videocassette distributor** that is **a joint venture of CBS Inc. and News Corp. 's 20th Century Fox studio** , which isn’t supplying any 8mm tapes at present .*

4. Referring to a mention in the headline from the body:
*Politics & Policy: Business and Labor Reach **a Consensus** On Need to Overhaul Health-Care System – By Ken neth H. Bacon*
*The pocketbook impact of health benefits has driven business and labor to **a surprising consensus**.*

5. Difficult to infer
*Eastern Airlines executives notified union leaders that the carrier wishes to discuss selective wage reductions on **Feb. 3**. Union representatives who could be reached said they hadn't decided whether they would respond. By proposing **a meeting date**, Eastern moved one step closer toward reopening current high-cost contract agreements with its unions.*

***Table 13**: Exceptions to our first discourse salience heuristic*

The second rule however worked with high precision and it was indeed the case that indefinite pronouns were almost never linked to any previous antecedent in the text. We therefore chose to only include the check for indefinite pronouns in our final system. This gave us no improvement on MUC-6-TEST and only a negligible improvement on ACE-ROTH-DEV (see Table 14), since there were not matches for indefinite pronouns in our data.

[2] http://en.wikipedia.org/wiki/Indefinite_pronouns

There are many mentions in our data set which do not have antecedents, but the system tries to find an antecedent for every given mention and hence makes a lot of avoidable precision errors. However, it is not easy to detect such mentions using simple rules like the ones we tried out. Thus, deciding when a given mention cannot have an antecedent seems to be one of the ways to improve coreference performance. The remaining subsections describe some of our ongoing work on further improving the coreference system.

Data Set	**Before**			**After**		
	Pairwise Precision	Pairwise Recall	*Pairwise F1*	Pairwise Precision	Pairwise Recall	*Pairwise F1*
MUC-6-TEST	80.33	59.64	***68.45***	80.33	59.64	***68.45***
ACE-ROTH-DEV	65.37	50.69	*57.1*	65.52	50.66	***57.14***

Table 14*: Improvements from checking of indefinite pronouns*

5.6 Modifier Checking

We talked a little about the problems arising from not checking head modifiers in Section 4.4 while discussing reasons for the inefficacy of Haghighi and Klein's semantic head pairs. This is actually a more general problem, also present in the system without semantic knowledge. Our system declares two nominal mentions to be coreferent only if their heads are the same (unless they have already been deemed coreferent by any of the deterministic rules like apposition detection, predicate nominative detection, etc.). However, while matching the heads of the mention's NPs, it does not take into consideration the other modifiers present in the NP. This may cause the system to make erroneous decisions, e.g. corefering "Stanford University" with "Harvard University", where the heads are same ("university").

Taking the head modifiers into account while doing the head matching is one of the things we are working on. Two of the approaches that we are currently exploring include:

1. Using an antonym dictionary to check if the modifiers of the two mentions are antonymous. E.g., "the **first** contract" vs. "the **second** contract", "**public** school" vs. "**private** school", etc.
2. Checking for the presence of different named entities in the noun phrase. E.g., "University of **Maryland** vs. University of **Southern California**", "the **Bush** campaign" vs. "the **Gore** campaign", etc.

5.7 Multiple Passes

So far, we have discussed a wide range of techniques and experiments aimed at improving coreference resolution. When we studied each individual component in isolation, we found there are some methods that are precision-oriented and there are some that are recall-oriented. The precision-oriented methods give you very high accuracy, but work only for a small subset of mentions. The recall-oriented ones, on the other hand enable you to make coreference decisions for a lot more mentions, but at the expense of finding many false positives. It is therefore a challenge to combine these various methods in a way that they complement each other. The solution that we are currently working on is a multi-pass system which makes multiple passes over the data set, with each pass dedicated to a certain set of functions.

We envision a layered system where the different techniques for finding coreference (apposition detection, head matching, agreement constraint checking, etc.) become individual layers of the system, with each layer feeding forward into the next one. The intuition is to have the highest precision technique as the topmost layer (the first pass) and then add subsequent layers (passes) of decreasing precision, but increasing recall. In any given pass, the system is not obliged to make a coreference decision for all of the mentions in the test set. For instance, in a particular pass, the system might choose to skip all the common noun mentions and/or all pronominal mentions. This allows the system to only link the mentions that it is highly confident of in that pass and leave the rest to the subsequent passes. Also in each pass, the system can only add new links, but cannot disturb the links already constructed in the previous higher confidence passes.

An example pass over the dataset could be just a simple string matching algorithm – For all non-pronominal mentions, go over all the mentions in the document and cluster together the mentions that are the same strings. String matching gave us a precision of **93.4** (with recall = 37.8) on MUC-6-TEST and **96.5** (with recall = 15.6) on ACE-ROTH-DEV. Because of its high precision, string matching could be a very good candidate for being the topmost layer (first pass) of the system. A possible configuration of the subsequent passes could be:

1. Exact String Match
2. Apposition, Predicate Nominative and Acronym Detection
3. Head matching with attribute checking
4. Head matching
5. Original single pass system

The final pass (lowest layer) could be just our original single pass system since it has fairly a high recall and would be able to link many of the mentions missed out by the earlier passes. A multi-pass system such as this would be more resilient to cascading errors than a single pass system. In the example below, one can easily tell at a glance that the two "Paramount's" are coreferent, but the single pass system has to proceed by building up the links in a sequential manner.

*__Paramount$_1$__, which agreed to sell the unit in July, said **it** would realize net proceeds from the sale of $2.6 billion, with an after-tax gain of $1.2 billion. __Paramount$_2$__ said the gain would be recorded in its fourth quarter, which ended yesterday.*

There is a coreference chain of the form *Paramount$_2$ -> it -> Paramount$_1$*. In the single pass system, suppose (Paramount$_1$, it) had been found coreferent, but the system had missed out the link from "Paramount$_2$" to "it", then (Paramount$_1$, Paramount$_2$) would not have been found coreferent. However, in the multi-pass system, "Paramount$_1$" and "Paramount$_2$" would have already been linked in the first pass (string match) and would remain linked irrespective of the links from "it" to either of them. The multi-pass system thus ensures that errors from low confidence decisions do not affect an otherwise high confidence decision. The multi-pass system, if implemented correctly thus holds a lot of promise for improving coreference resolution. We are currently running experiments to empirically determine the optimum number of passes and the identity of each individual pass.

5.8 Reranking

Haghighi and Klein argued that even a simple deterministic coreference system with a few rich syntactic and semantic features can do almost as well as complex machine learning systems which use a large number of features. While we like the notion of a simple modular system not overloaded with features, we think that the deterministic nature of the system creates a handicap. The current system is forced to make a coreference decision the moment it finds an antecedent that has satisfied all the coreference constraints. Moreover, the system is always making a binary Yes/No decision and hence does not have any notion of the "goodness" of a match. Thus, once a "Yes" decision has been made, the system does not look further back in the text for finding more suitable candidates or in other words, better scoring matches. During our initial error analysis (Section 3), we found that a lot of the errors were caused due to the system making a coreference decision too soon, when the true coreferent antecedent actually came earlier in the text (instead of just being the most recent one). For instance:

*Oil production from **Australia's Bass Strait fields** will be raised by 11,000 barrels a day to about 321,000 barrels with the launch of the Whiting field, the first of **five small fields scheduled to be brought into production before the end of 1990**. Esso Australia Ltd., a unit of New York-based Exxon Corp., and Broken Hill Pty. operate **the fields** in a joint venture.*

In this case, the deterministic system will corefer the mention "the fields" with the noun phrase starting "five small fields …". However, the correct antecedent is the noun phrase "Australia's Bass Strait fields" which comes further back in the text. One way of tackling this issue would be to use a reranker that assigns a score to each of the top *n* antecedents that the system deems to be coreferent, and then ranks them using the assigned scores to find the best possible match.

Using a reranker had been one of the earliest ideas that we had conceived, soon after we had finished our initial error analysis. Back then, we had run a couple of experiments to get statistics about the number of cases where such an approach could be helpful. We had modified our original system (the system described in section 4) to not stop at the first match, but continue going backwards and output the four most recent antecedents (including the first match) that it thinks are coreferent with the mention under question. Essentially, it was similar to asking the system to output an n-best list of antecedents where n = 4. We then labeled these antecedents as being correct or wrong based on the true gold annotation. Table 6 shows the fraction of mentions in the MUC-6-TRAIN corpus for whom, the system's first proposal was wrong, but a subsequent proposal was correct.

The correctness of system's proposals, (from most recent to least recent decision)	**Number of Mentions**	**Percentage of Mentions**
False True	464	4.7%
False False True	178	1.8%
False False False True	76	0.8%
Total	718	7.3%

Table 6*: The percentage of mentions (out of the total 9,853) in the development set, for whom the system's first proposal was incorrect but the second, third or fourth proposal was correct instead. This was experiment done at an early stage of the research with the system described in Section 4.*

Thus, 7.3% of all mentions in MUC-6-TRAIN could have potentially benefited had the system looked further after its first match. Since the F1 score of the system is calculated based on the number of correct links (which are significantly larger than the number of mentions) rather than the number of correctly co-referred mentions, we carried out an oracle experiment to find out what the potential gain in F1 would be. Our oracle experiments suggested that there was a potential of improving the system performance by **12 F1 points** on MUC-6-TRAIN and by **9.48 F1** points, if the system were to always choose the correct antecedent from among the top three proposals (instead of the first one). Thus, there certainly seemed to opportunities for improvement using a reranker.

However, there were few reasons why we postponed implementing the idea. First of all, even though we could think of many supervised ways to train the reranker, we could not come up with a convincing unsupervised approach to assign scores to the coreference candidates. We were reluctant to add a supervised machine learning component to our system, which was otherwise entirely unsupervised. Secondly, we wanted our system to be as simple and reproducible as possible and adding a machine learning component would have increased the complexity of the system. Thirdly, there seemed to be many other avenues for improvement (Sections 5.1 - 5.7) which seemed promising and would have still preserved the system as a simple, deterministic and unsupervised system. We therefore decided to start working on other approaches first, with a plan to return to the idea of a reranker later. Reranking seems to be the ideal way to gain performance by using machine learning in the system, but at the same time requiring minimal changes to the original core system and ensuring its modular nature.

5.9 Experimental Results

Table 15 compares the final results of our current system with previous work. Similar to Haghighi and Klein, our performance on MUC-6-TEST is significantly better than on ACE-ROTH-DEV. Our system, when using the named entity tags provided by the Stanford NER is almost as good as Haghighi and Klein's final system on MUC-6-TEST. However, on ACE-ROTH-DEV, our results are comparable to his system without the semantic module. These results are still quite impressive considering the fact that their system actually peeks at the test set for guiding their bootstrapping algorithm, while we neither look at the test set nor use any external semantic information. In fact, if our system was allowed to look at the test set and use the gold NE annotations (instead of using the Stanford NER), our system would beat Haghighi and Klein (2009) by 1.5 F1 points on MUC-6-TEST.

	MUC-6-TEST		
	Pairwise Precision	Pairwise Recall	*Pairwise F1*
Our System with gold entity types	80.3	59.6	***68.5***
Our System with Stanford NER	77.8	58.8	*67.0*
Haghighi and Klein (2009)	80.5	57.8	*67.3*
Poon and Domingos (2008)	63.0	57.0	*60.0*
Finkel and Manning (2008)	74.1	37.1	*49.5*
	ACE-ROTH-DEV		
Our System with gold entity types	65.5	50.7	*57.1*
Our System with Stanford NER	64.3	48.6	*55.3*
Haghighi and Klein (2009)	68.2	51.2	***58.5***
Haghighi and Klein (2009) without semantic module	71.3	45.4	*55.5*

Table 15*: Comparing our system with previous work*

6 Conclusion

We first reviewed the simple coreference resolution system of Haghighi and Klein (2009) and discussed the various challenges in replicating their system, particularly talking in detail about the impractical nature of their semantic compatibility module. The bootstrapping approach used by them for mining semantic knowledge (which was the highest performance booster for their system) works well in practice only if the test set is known in advance. This is a serious handicap of their system, because it is not capable of performing well on previously unseen test sets without running the entire semantic knowledge extraction procedure again. Our attempts at generalizing their bootstrapping algorithm to make it test set-independent were not fruitful, which further supports the claim that their knowledge mining technique is highly unreliable unless done in a very careful and controlled fashion. E.g., by using the test set to prune the pattern set, etc.

We therefore came up with better techniques for improving coreference resolution which are competitive in performance when compared to Haghighi and Klein's system (2009), but do not peek at the test data or use any externally mined semantic information. The salient features of our current system are

(i) It does not need to know the test set in advance. It works equally well on unseen test sets.

(ii) It does not require large amounts of parsed text to mine semantically compatible head pairs.

(iii) It does not have a time-consuming preprocessing phase where it uses the test set for mining semantic information via a bootstrapping algorithm.

(iv) It uses straightforward techniques which are easy to reproduce, in contrast to bootstrapping, which is often very unreliable.

(v) It provides a baseline that is almost as strong as Haghighi and Klein (2009), but at the same time is more flexible (test-set independence), faster (considering preprocessing time) and simpler to implement.

We then introduced the notion of a multi-pass system which is a multilayered system with each layer (or pass) making coreference decisions based only on a limited set of heuristics, while leaving the remaining mentions to be handled by the lower layers (subsequent passes). The topmost layer (or the first pass) is a very high precision one, which only makes high confidence coreference decisions. Each subsequent pass has a higher recall but lower precision than its previous pass. Such a pass structure allows the system to link mentions across the document that it is most confident about, without being forced to make coreference decisions for the intervening mentions. This can help in reducing a lot of the cascading errors.

Finally, we provide some arguments against the deterministic nature of the existing system and prove using oracle experiments that there is potential for building a reranker on top of the existing system. The multi-pass system and the reranker are the current focus of our research. Either of these approaches, if implemented cleverly enough, would not only improve the performance of the original system, but also stay true to Haghighi and Klein's fundament pitch – of having a simple modular coreference resolution system without an overload of features.

Acknowledgments

The research project on coreference resolution has benefited from the helpful comments and code contributions of Mihai Surdeanu, Nathanael Chambers, Jenny Finkel, Heeyoung Lee and Sudarshan Rangarajan.

References

Aria Haghighi and Dan Klein. 2009. Simple Coreference Resolution with Rich Syntactic and Semantic Features. *In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing.*

A Culotta, M Wick, R Hall, and A McCallum. 2007. First-order probabilistic models for coreference resolution. *In NAACL-HLT.*

Eric Bengston and Dan Roth. 2008. Understanding the value of features for coreference resolution. *In Empirical Methods in Natural Language Processing.*

Jenny Finkel and Christopher Manning. 2008. Enforcing transitivity in coreference resolution. *In Association of Computational Linguists (ACL).*

David Bean and Ellen Riloff. 2004. Unsupervised Learning of Contextual Role Knowledge for Coreference Resolution. *In Proceedings of HLT/NAACL 2004.*

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov Logic. *In Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing.*

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. *In Association of Computational Linguists (ACL).*

Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. *Cambridge, MA: MIT Press.*

Dekang Lin. 2005. Proximity-based thesaurus. http://webdocs.cs.ualberta.ca/~lindek/downloads.htm

Heng Ji and Dekang Lin. 2009. Gender and Animacy Knowledge Discovery from Web-Scale N-Grams for Unsu pervised Person Mention Detection. Proc. *PACLIC 2009.*

Shane Bergsma and Dekang Lin. Bootstrapping Path-Based Pronoun Resolution. *In Proceedings of the Conference on Computational Linguistics / Association for Computational Linguistics (COLING/ACL-06), Sydney, Australia, July 17-21, 2006.*