Entity-Based Cross-Document Coreferencing Using the Vector Space Model

Amit Bagga
Box 90129
Dept. of Computer Science
Duke University
Durham, NC 27708-0129
amit@cs.duke.edu

Breck Baldwin
Institute for Research in Cognitive Sciences
University of Pennsylvania
3401 Walnut St. 400C
Philadelphia, PA 19104
breck@unagi.cis.upenn.edu

Abstract

Cross-document coreference occurs when the same person, place, event, or concept is discussed in more than one text source. Computer recognition of this phenomenon is important because it helps break "the document boundary" by allowing a user to examine information about a particular entity from multiple text sources at the same time. In this paper we describe a cross-document coreference resolution algorithm which uses the Vector Space Model to resolve ambiguities between people having the same name. In addition, we also describe a scoring algorithm for evaluating the cross-document coreference chains produced by our system and we compare our algorithm to the scoring algorithm used in the MUC-6 (within document) coreference task.

1 Introduction

Cross-document coreference occurs when the same person, place, event, or concept is discussed in more than one text source. Computer recognition of this phenomenon is important because it helps break "the document boundary" by allowing a user to examine information about a particular entity from multiple text sources at the same time. In particular, resolving cross-document coreferences allows a user to identify trends and dependencies across documents. Cross-document coreference can also be used as the central tool for producing summaries from multiple documents, and for information fusion, both of which have been identified as advanced areas of research by the TIPSTER Phase III program. Cross-document coreference was also identified as one of the potential tasks for the Sixth Message Understanding Conference (MUC-6) but was not included as a formal task because it was considered too ambitious (Grishman 94).

In this paper we describe a highly successful cross-document coreference resolution algorithm which uses the Vector Space Model to resolve ambiguities between people having the same name. In addition, we also describe a scoring algorithm for evaluating the cross-document coreference chains produced by our system and we compare our algorithm to the scoring algorithm used in the MUC-6 (within document) coreference task.

2 Cross-Document Coreference: The Problem

Cross-document coreference is a distinct technology from Named Entity recognizers like IsoQuest's NetOwl and IBM's Textract because it attempts to determine whether name matches are actually the same individual (not all John Smiths are the same). Neither NetOwl or Textract have mechanisms which try to keep same-named individuals distinct if they are different people.

Cross-document coreference also differs in substantial ways from within-document coreference. Within a document there is a certain amount of consistency which cannot be expected across documents. In addition, the problems encountered during within document coreference are compounded when looking for coreferences across documents because the underlying principles of linguistics and discourse context no longer apply across documents. Because the underlying assumptions in cross-document coreference are so distinct, they require novel approaches.

3 Architecture and the Methodology

Figure 1 shows the architecture of the cross-document system developed. The system is built upon the University of Pennsylvania's within document coreference system, CAMP, which participated in the Seventh Message Understanding Conference (MUC-7) within document coreference task (MUC-7 1998).

Our system takes as input the coreference processed documents output by CAMP. It then passes these documents through the SentenceExtractor module which extracts, for each document, all the sentences relevant to a particular entity of interest. The VSM-Disambiguate module then uses a vector space model algorithm to compute similarities between the sentences extracted for each pair of documents.

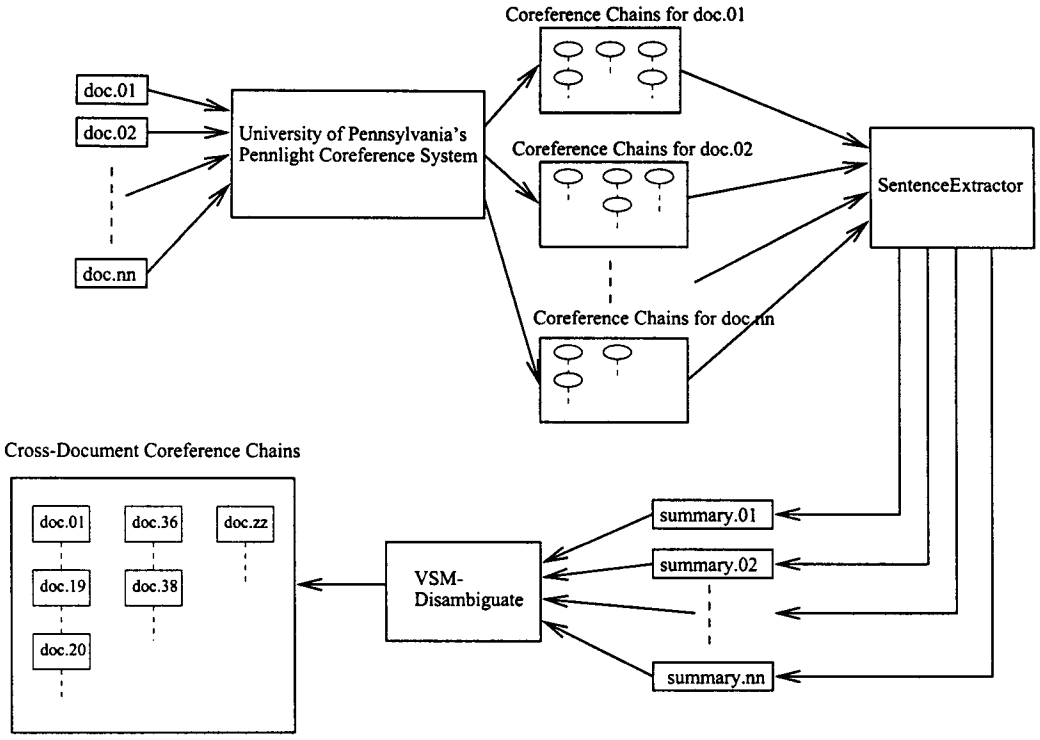

Figure 1: Architecture of the Cross-Document Coreference System

John Perry, of Weston Golf Club, announced his resignation yesterday. He was the President of the Massachusetts Golf Association. During his two years in office, Perry guided the MGA into a closer relationship with the Women's Golf Association of Massachusetts.

Figure 2: Extract from doc.36

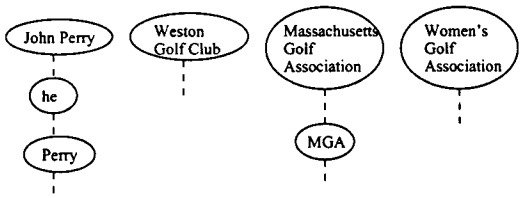

Figure 3: Coreference Chains for doc.36

Details about each of the main steps of the cross-document coreference algorithm are given below.

- First, for each article, CAMP is run on the article. It produces coreference chains for all the entities mentioned in the article. For example, consider the two extracts in Figures 2 and 4. The coreference chains output by CAMP for the two extracts are shown in Figures 3 and 5.

Oliver "Biff" Kelly of Weymouth succeeds John Perry as president of the Massachusetts Golf Association. "We will have continued growth in the future," said Kelly, who will serve for two years. "There's been a lot of changes and there will be continued changes as we head into the year 2000."

Figure 4: Extract from doc.38

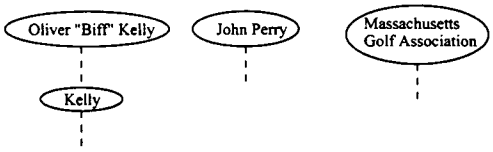

Figure 5: Coreference Chains for doc.38

- Next, for the coreference chain of interest within each article (for example, the coreference chain that contains "John Perry"), the Sentence Extractor module extracts all the sentences that contain the noun phrases which form the coreference chain. In other words, the SentenceExtractor module produces a "summary" of the article with respect to the entity of interest. These summaries are a special case of the query sensitive techniques being developed at Penn using

CAMP. Therefore, for doc.36 (Figure 2), since at least one of the three noun phrases ("John Perry," "he," and "Perry") in the coreference chain of interest appears in each of the three sentences in the extract, the summary produced by SentenceExtractor is the extract itself. On the other hand, the summary produced by SentenceExtractor for the coreference chain of interest in doc.38 is only the first sentence of the extract because the only element of the coreference chain appears in this sentence.

- For each article, the VSM-Disambiguate module uses the summary extracted by the SentenceExtractor and computes its similarity with the summaries extracted from each of the other articles. Summaries having similarity above a certain threshold are considered to be regarding the same entity.

4 University of Pennsylvania's CAMP System

The University of Pennsylvania's CAMP system resolves within document coreferences for several different classes including pronouns, and proper names (Baldwin 95). It ranked among the top systems in the coreference task during the MUC-6 and the MUC-7 evaluations.

The coreference chains output by CAMP enable us to gather all the information about the entity of interest in an article. This information about the entity is gathered by the SentenceExtractor module and is used by the VSM-Disambiguate module for disambiguation purposes. Consider the extract for doc.36 shown in Figure 2. We are able to include the fact that the John Perry mentioned in this article was the president of the Massachusetts Golf Association only because CAMP recognized that the "he" in the second sentence is coreferent with "John Perry" in the first. And it is this fact which actually helps VSM-Disambiguate decide that the two John Perrys in doc.36 and doc.38 are the same person.

5 The Vector Space Model

The vector space model used for disambiguating entities across documents is the standard vector space model used widely in information retrieval (Salton 89). In this model, each summary extracted by the SentenceExtractor module is stored as a vector of terms. The terms in the vector are in their morphological root form and are filtered for stop-words (words that have no information content like *a*, *the*, *of*, *an*, ...). If S_1 and S_2 are the vectors for the two summaries extracted from documents D_1 and D_2, then their similarity is computed as:

$$Sim(S_1, S_2) = \sum_{common\ terms\ t_j} w_{1j} \times w_{2j}$$

where t_j is a term present in both S_1 and S_2, w_{1j} is the weight of the term t_j in S_1 and w_{2j} is the weight of t_j in S_2.

The weight of a term t_j in the vector S_i for a summary is given by:

$$w_{ij} = \frac{tf \times \log \frac{N}{df}}{\sqrt{s_{i1}^2 + s_{i2}^2 + \ldots + s_{in}^2}}$$

where tf is the frequency of the term t_j in the summary, N is the total number of documents in the collection being examined, and df is the number of documents in the collection that the term t_j occurs in. $\sqrt{s_{i1}^2 + s_{i2}^2 + \ldots + s_{in}^2}$ is the *cosine normalization factor* and is equal to the Euclidean length of the vector S_i.

The VSM-Disambiguate module, for each summary S_i, computes the similarity of that summary with each of the other summaries. If the similarity computed is above a pre-defined threshold, then the entity of interest in the two summaries are considered to be coreferent.

6 Experiments

The cross-document coreference system was tested on a highly ambiguous test set which consisted of 197 articles from 1996 and 1997 editions of the New York Times. The sole criteria for including an article in the test set was the presence or the absence of a string in the article which matched the "/John.*?Smith/" regular expression. In other words, all of the articles either contained the name *John Smith* or contained some variation with a middle initial/name. The system did not use any New York Times data for training purposes. The answer keys regarding the cross-document chains were manually created, but the scoring was completely automated.

6.1 Analysis of the Data

There were 35 different *John Smith*s mentioned in the articles. Of these, 24 of them only had one article which mentioned them. The other 173 articles were regarding the 11 remaining *John Smith*s. The background of these *John Smith*s , and the number of articles pertaining to each, varied greatly. Descriptions of a few of the *John Smith*s are: Chairman and CEO of General Motors, assistant track coach at UCLA, the legendary explorer, and the main character in Disney's Pocahontas, former president of the Labor Party of Britain.

7 Scoring the Output

In order to score the cross-document coreference chains output by the system, we had to map the cross-document coreference scoring problem to a within-document coreference scoring problem. This

was done by creating a meta document consisting of the file names of each of the documents that the system was run on. Assuming that each of the documents in the data set was about a single *John Smith*, the cross-document coreference chains produced by the system could now be evaluated by scoring the corresponding within-document coreference chains in the meta document.

We used two different scoring algorithms for scoring the output. The first was the standard algorithm for within-document coreference chains which was used for the evaluation of the systems participating in the MUC-6 and the MUC-7 coreference tasks.

The shortcomings of the MUC scoring algorithm when used for the cross-document coreference task forced us to develop a second algorithm.

Details about both these algorithms follow.

7.1 The MUC Coreference Scoring Algorithm[1]

The MUC algorithm computes precision and recall statistics by looking at the number of links identified by a system compared to the links in an answer key. In the model-theoretic description of the algorithm that follows, the term "key" refers to the manually annotated coreference chains (the truth) while the term "response" refers to the coreference chains output by a system. An equivalence set is the transitive closure of a coreference chain. The algorithm, developed by (Vilain 95), computes recall in the following way.

First, let S be an equivalence set generated by the key, and let $R_1 \ldots R_m$ be equivalence classes generated by the response. Then we define the following functions over S:

- p(S) is a partition of S relative to the response. Each subset of S in the partition is formed by intersecting S and those response sets R_i that overlap S. Note that the equivalence classes defined by the response may include implicit singleton sets - these correspond to elements that are mentioned in the key but not in the response. For example, say the key generates the equivalence class S = {A B C D}, and the response is simply <A-B>. The relative partition p(S) is then {A B} {C} and {D}.
- c(S) is the minimal number of "correct" links necessary to generate the equivalence class S. It is clear that c(S) is one less than the cardinality of S, i.e., $c(S) = (|S| - 1)$.
- m(S) is the number of "missing" links in the response relative to the key set S. As noted above, this is the number of links necessary to fully reunite any components of the p(S) partition. We note that this is simply one fewer than the number of elements in the partition, that is, $m(S) = (|p(S)| - 1)$.

[1]The exposition of this scorer has been taken nearly entirely from (Vilain 95).

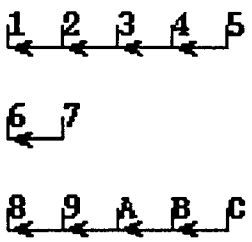

Figure 6: Truth

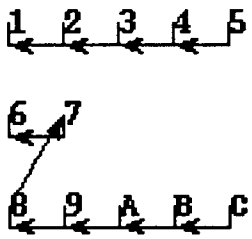

Figure 7: Response: Example 1

Looking in isolation at a single equivalence class in the key, the recall *error* for that class is just the number of missing links divided by the number of correct links, i.e., $\frac{m(S)}{c(S)}$.

Recall in turn is $\frac{c(S)-m(S)}{c(S)}$, which equals

$$\frac{(|S| - 1) - (|p(S)| - 1)}{|S| - 1} .$$

The whole expression can now be simplified to

$$\frac{|S| - |p(S)|}{|S| - 1}.$$

Precision is computed by switching the roles of the key and response in the above formulation.

7.2 Shortcomings of the MUC Scoring Algorithm

While the (Vilain 95) provides intuitive results for coreference scoring, it however does not work as well in the context of evaluating cross document coreference. There are two main reasons.

1. The algorithm does not give any credit for separating out singletons (entities that occur in chains consisting only of one element, the entity itself) from other chains which have been identified. This follows from the convention in

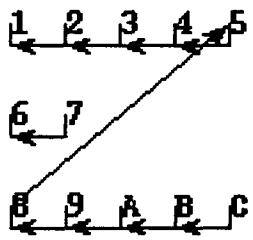

Figure 8: Response: Example 2

coreference annotation of not identifying those entities that are markable as possibly coreferent with other entities in the text. Rather, entities are only marked as being coreferent if they actually are coreferent with other entities in the text. This shortcoming could be easily enough overcome with different annotation conventions and with minor changes to the algorithm, but it is worth noting.

2. All errors are considered to be equal. The MUC scoring algorithm penalizes the precision numbers equally for all types of errors. It is our position that, for certain tasks, some coreference errors do more damage than others.

   Consider the following examples: suppose the truth contains two large coreference chains and one small one (Figure 6), and suppose Figures 7 and 8 show two different responses. We will explore two different precision errors. The first error will connect one of the large coreference chains with the small one (Figure 7). The second error occurs when the two large coreference chains are related by the errant coreferent link (Figure 8). It is our position that the second error is more damaging because, compared to the first error, the second error makes more entities coreferent that should not be. This distinction is not reflected in the (Vilain 95) scorer which scores both responses as having a precision score of 90% (Figure 9).

7.3 Our B-CUBED Scoring Algorithm[2]

Imagine a scenario where a user recalls a collection of articles about *John Smith*, finds a single article about the particular *John Smith* of interest and wants to see all the other articles about that individual. In commercial systems with News data, precision is typically the desired goal in such settings. As a result we wanted to model the accuracy of the system on a per-document basis and then build a more global score based on the sum of the user's experiences.

[2]The main idea of this algorithm was initially put forth by Alan W. Biermann of Duke University.

Consider the case where the user selects document 6 in Figure 8. This a good outcome with all the relevant documents being found by the system and no extraneous documents. If the user selected document 1, then there are 5 irrelevant documents in the systems output – precision is quite low then. The goal of our scoring algorithm then is to model the precision and recall on average when looking for more documents about the same person based on selecting a single document.

Instead of looking at the links produced by a system, our algorithm looks at the presence/absence of entities from the chains produced. Therefore, we compute the precision and recall numbers for each entity in the document. The numbers computed with respect to each entity in the document are then combined to produce final precision and recall numbers for the entire output.

For an entity, i, we define the precision and recall with respect to that entity in Figure 10.

The final precision and recall numbers are computed by the following two formulae:

$$\text{Final Precision} = \sum_{i=1}^{N} w_i * Precision_i$$

$$\text{Final Recall} = \sum_{i=1}^{N} w_i * Recall_i$$

where N is the number of entities in the document, and w_i is the weight assigned to entity i in the document. For all the examples and the experiments in this paper we assign equal weights to each entity i.e. $w_i = 1/N$. We have also looked at the possibilities of using other weighting schemes. Further details about the B-CUBED algorithm including a model theoretic version of the algorithm can be found in (Bagga 98a).

Consider the response shown in Figure 7. Using the B-CUBED algorithm, the precision for entity-6 in the document equals 2/7 because the chain output for the entity contains 7 elements, 2 of which are correct, namely {6,7}. The recall for entity-6, however, is 2/2 because the chain output for the entity has 2 correct elements in it and the "truth" chain for the entity only contains those 2 elements. Figure 9 shows the final precision and recall numbers computed by the B-CUBED algorithm for the examples shown in Figures 7 and 8. The figure also shows the precision and recall numbers for each entity (ordered by entity-numbers).

7.4 Overcoming the Shortcomings of the MUC Algorithm

The B-CUBED algorithm does overcome the the two main shortcomings of the MUC scoring algorithm discussed earlier. It implicitly overcomes the first

Output	MUC Algorithm	B-CUBED Algorithm (equal weights for every entity)
Example 1	P: $\frac{9}{10}$ (90%)	P: $\frac{1}{12} * [\frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{2}{7} + \frac{2}{7} + \frac{5}{7} + \frac{5}{7} + \frac{5}{7} + \frac{5}{7} + \frac{5}{7}] = 76\%$
	R: $\frac{9}{9}$ (100%)	R: $\frac{1}{12} * [\frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{2}{2} + \frac{2}{2} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5}] = 100\%$
Example 2	P: $\frac{9}{10}$ (90%)	P: $\frac{1}{12} * [\frac{5}{10} + \frac{5}{10} + \frac{5}{10} + \frac{5}{10} + \frac{5}{10} + \frac{2}{2} + \frac{2}{2} + \frac{5}{10} + \frac{5}{10} + \frac{5}{10} + \frac{5}{10} + \frac{5}{10}] = 58\%$
	R: $\frac{9}{9}$ (100%)	R: $\frac{1}{12} * [\frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{2}{2} + \frac{2}{2} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5}] = 100\%$

Figure 9: Scores of Both Algorithms on the Examples

$$\text{Precision}_i = \frac{\textit{number of correct elements in the output chain containing entity}_i}{\textit{number of elements in the output chain containing entity}_i} \quad (1)$$

$$\text{Recall}_i = \frac{\textit{number of correct elements in the output chain containing entity}_i}{\textit{number of elements in the truth chain containing entity}_i} \quad (2)$$

Figure 10: Definitions for Precision and Recall for an Entity i

shortcoming of the MUC-6 algorithm by calculating the precision and recall numbers for each entity in the document (irrespective of whether an entity is part of a coreference chain). Consider the responses shown in Figures 7 and 8. We had mentioned earlier that the error of linking the the two large chains in the second response is more damaging than the error of linking one of the large chains with the smaller chain in the first response. Our scoring algorithm takes this into account and computes a final precision of 58% and 76% for the two responses respectively. In comparison, the MUC algorithm computes a precision of 90% for both the responses (Figure 9).

8 Results

Figure 11 shows the precision, recall, and F-Measure (with equal weights for both precision and recall) using the B-CUBED scoring algorithm. The Vector Space Model in this case constructed the space of terms only from the summaries extracted by SentenceExtractor. In comparison, Figure 12 shows the results (using the B-CUBED scoring algorithm) when the vector space model constructed the space of terms from the articles input to the system (it still used the summaries when computing the similarity). The importance of using CAMP to extract summaries is verified by comparing the highest F-Measures achieved by the system for the two cases. The highest F-Measure for the former case is 84.6% while the highest F-Measure for the latter case is 78.0%. In comparison, for this task, named-entity tools like NetOwl and Textract would mark all the *John Smith*s the same. Their performance using our

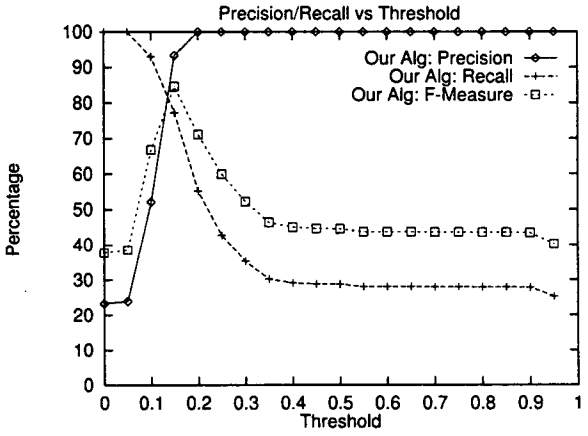

Figure 11: Precision, Recall, and F-Measure Using the B-CUBED Algorithm With Training On the Summaries

scoring algorithm is 23% precision, and 100% recall.

Figures 13 and 14 show the precision, recall, and F-Measure calculated using the MUC scoring algorithm. Also, the baseline case when all the *John Smith*s are considered to be the same person achieves 83% precision and 100% recall. The high initial precision is mainly due to the fact that the MUC algorithm assumes that all errors are equal.

We have also tested our system on other classes of cross-document coreference like names of companies, and events. Details about these experiments can be found in (Bagga 98b).

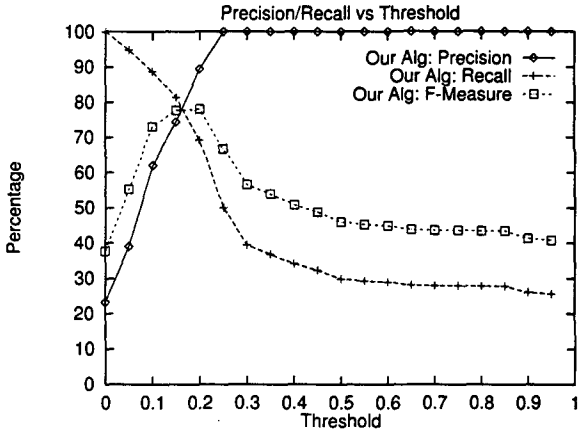

Figure 12: Precision, Recall, and F-Measure Using the B-CUBED Algorithm With Training On Entire Articles

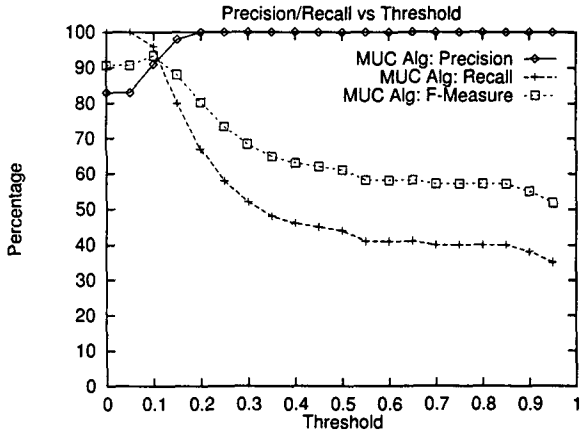

Figure 13: Precision, Recall, and F-Measure Using the MUC Algorithm With Training On the Summaries

9 Conclusions

As a novel research problem, cross document coreference provides an different perspective from related phenomenon like named entity recognition and within document coreference. Our system takes summaries about an entity of interest and uses various information retrieval metrics to rank the similarity of the summaries. We found it quite challenging to arrive at a scoring metric that satisfied our intuitions about what was good system output v.s. bad, but we have developed a scoring algorithm that is an improvement for this class of data over other within document coreference scoring algorithms. Our results are quite encouraging with potential performance being as good as 84.6% (F-Measure).

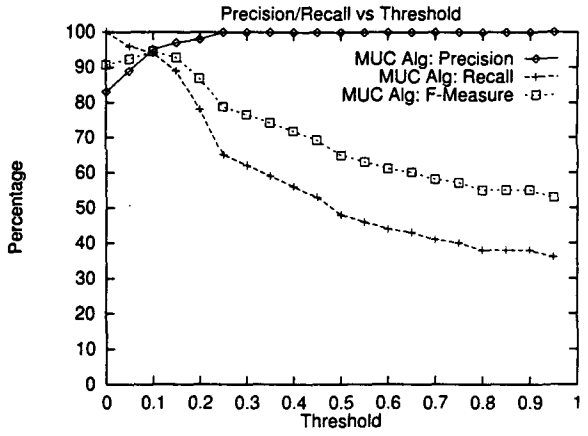

Figure 14: Precision, Recall, and F-Measure Using the MUC Algorithm With Training On Entire Articles

10 Acknowledgments

The first author was supported in part by a Fellowship from IBM Corporation, and in part by the Institute for Research in Cognitive Science at the University of Pennsylvania.

References

Bagga, Amit, and Breck Baldwin. Algorithms for Scoring Coreference Chains. To appear at *The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, May 1998.

Bagga, Amit, and Breck Baldwin. How Much Processing Is Required for Cross-Document Coreference? To appear at *The First International Conference on Language Resources and Evaluation on Linguistics Coreference*, May 1998.

Baldwin, Breck, et al. University of Pennsylvania: Description of the University of Pennsylvania System Used for MUC-6, *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pp. 177-191, November 1995.

Grishman, Ralph. Whither Written Language Evaluation?, *Proceedings of the Human Language Technology Workshop*, pp. 120-125, March 1994, San Francisco: Morgan Kaufmann.

Proceedings of the Seventh Message Understanding Conference (MUC-7), April 1998.

Salton, Gerard. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer, 1989, Reading, MA: Addison-Wesley.

Vilain, Marc, et al. A Model-Theoretic Coreference Scoring Scheme, *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pp. 45-52, November 1995, San Francisco: Morgan Kaufmann.