Algorithms for Scoring Coreference Chains

Amit Bagga
Box 90129, Dept. of Computer Science
Duke University
Durham, NC 27708-0129. USA
Phone: (919) 660-6507
Fax: (919) 660-6519
Email: amit@cs.duke.edu

Breck Baldwin
Institute for Research in Cognitive Science
3401 Walnut St. 400C
Philadelphia, PA 19104. USA
Phone: (215) 898-0329
Fax: (215) 573-9247
Email: breck@linc.cis.upenn.edu

Abstract

Scoring the performance of a system is an extremely important aspect of coreference algorithm performance. The score for a particular run is the single strongest measure of how well the system is performing and it can strongly determine directions for further improvements. In this paper, we present several different scoring algorithms and detail their respective strengths and weaknesses for varying classes of processing. We also demonstrate that tasks like information extraction have very different needs from information retrieval in terms of how to score the performance of coreference annotation.

Introduction

Scoring the performance of a system is an extremely important aspect of coreference algorithm performance. The score for a particular run is the single strongest measure of how well the system is performing and it can strongly determine directions for further improvements. In this paper, we present several different scoring algorithms and detail their respective strengths and weaknesses for varying classes of processing. In particular, we describe and analyze the coreference scoring algorithm used to evaluate the coreference systems in the sixth Message Understanding Conference (MUC-6)(MUC-6, 95). We also present two shortcomings of this algorithm. In addition, we present a new coreference scoring algorithm, our B-CUBED algorithm, which was designed to overcome the shortcomings of the MUC-6 algorithm. Moreover, we demonstrate that tasks like information extraction have very different needs from information retrieval in terms of how to score the performance of coreference annotation.

The MUC-6 Algorithm: Vilain et al.

Prior to Vilain et al.'s coreference scoring algorithm (Vilain, 95) there had been a graph based scoring algorithm (Sundheim et al.) which produced unintuitive results for even very simple cases. (Vilain, 95) substituted a model-theoretic scoring algorithm which produced very intuitive results for the type of scoring desired in MUC-6. This algorithm computes computes the recall error by taking each equivalence class S (defined by the links in the answer key) and determining the number of coreference links m that would have to be added to the response to place all the entities in S into the same equivalence class in the response. Recall error then is the sum of m's divided by the number of links in the key. Precision error is computed by reversing the roles of the answer key and the response.

The full details of the algorithm are discussed next.

The Model Theoretic Approach To The MUC-6 Algorithm[1]

In the description of the model theoretic algorithm, the terms "key," and "response" are defined in the following way:

key refers to the manually annotated coreference chains (the truth).

response refers to the coreference chains output by a system.

An equivalence set is the transitive closure of a coreference chain. The algorithm 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

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

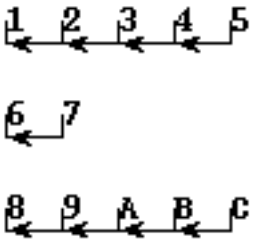

Figure 1: Truth

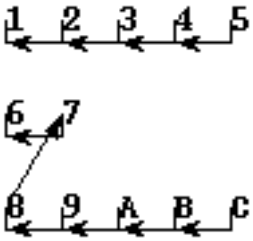

Figure 2: Response: Example 1

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,

$$m(S) = (|p(S)| - 1) \ .$$

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

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

Finally, extending this measure from a single key equivalence class to an entire set T simply requires summing over the key equivalence classes. That is,

$$R_T = \frac{\sum(|S_i| - |p(S_i)|)}{\sum(|S_i| - 1)} \ . \qquad (2)$$

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

Example

For example, let the key contain 3 equivalence classes as shown in Figure 1. Suppose Figure 2 shows a response. From Figure 3(I), the three equivalence classes in the truth, S_1, S_2, and S_3, are {1, 2, 3, 4, 5}, {6, 7}, and {8, 9, A, B, C} respectively. And the partitions $p(S_1)$, $p(S_2)$, and $p(S_3)$, with respect to the response, shown in Figure 3(II), are {1, 2, 3, 4, 5}, {6, 7}, and {8, 9, A, B, C} respectively. Using equation 2, the recall can now be calculated in the following way:

$$\text{Recall} = \frac{(5-1) + (2-1) + (5-1)}{(5-1) + (2-1) + (5-1)} = 9/9 = 100\% \ .$$

Similarly, if the roles of the key and the response are reversed, then the equivalence classes in the truth, S_1, and S_2, are {1, 2, 3, 4, 5} and {6, 7, 8, 9, A, B, C}], and the partitions, $p(S_1)$, and $p(S_2)$, are {1, 2, 3, 4, 5} and [{6, 7} {8, 9, A, B, C}] respectively (Figure 3(III)). The precision can now be calculated as:

$$\text{Precision} = \frac{(5-1) + (7-2)}{(5-1) + (7-1)} = 9/10 = 90\% \ .$$

Shortcomings of the MUC-6 Algorithm

Despite the advances of the model-theoretic scorer, it yields unintuitive results for some tasks. 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 coreference annotation of not identifying those entities that are markable as possibly coreferent with other entities in the text.

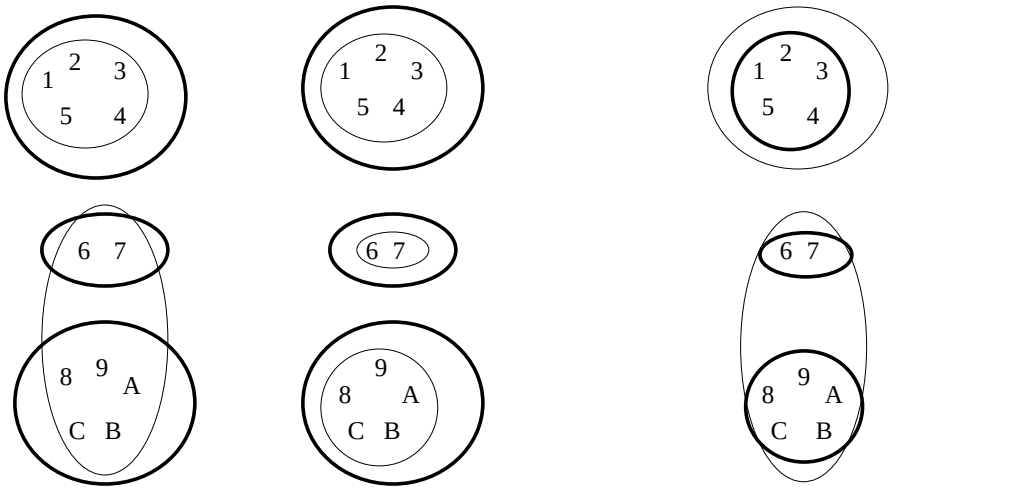

Figure 3: Equivalence Classes and Their Partitions For Example 1

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 1), and suppose Figures 2 and 4 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 2). The second error occurs when the two large coreference chains are related by the errant coreferent link (Figure 4). 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 6).

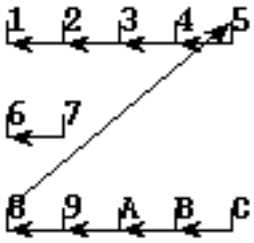

Figure 4: Response: Example 2

Revisions to the Algorithm: Our B-CUBED Algorithm[2]

Our B-CUBED algorithm was designed to overcome the two shortcomings of the MUC-6 algorithm. Instead of looking at the links produced by a system, our algorithm looks at the presence/absence of entities relative to each of the other entities in the equivalence classes produced. Therefore, we compute the precision and recall numbers for each entity in the document, which are then combined to produce final precision and recall numbers for the entire output. The formal model-theoretic version of our algorithm is discussed in the next section.

For an entity, i, we define the precision and recall with respect to that entity in Figure 5.

The final precision and recall numbers are computed

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

$$\text{Precision}_\text{i} = \frac{number\ of\ correct\ elements\ in\ the\ output\ chain\ containing\ entity_i}{number\ of\ elements\ in\ the\ output\ chain\ containing\ entity_i}$$

$$\text{Recall}_\text{i} = \frac{number\ of\ correct\ elements\ in\ the\ output\ chain\ containing\ entity_i}{number\ of\ elements\ in\ the\ truth\ chain\ containing\ entity_i}$$

Figure 5: Definitions for Precision and Recall for an Entity i

by the following two formulae:

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

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

where N is the number of entities in the document, and w_i is the weight assigned to entity i in the document. It should be noted that the B-CUBED algorithm implicitly overcomes the first 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).

Different weighting schemes produce different versions of the algorithm. The choice of the weighting scheme is determined by the task for which the algorithm is going to be used.

When coreference (or cross-document coreference) is used for an information extraction task, where information about every entity in an equivalence class is important, the weighting scheme assigns equal weights for every entity i. For example, the weight assigned to each entity in Figure 1 is 1/12. As shown in Figure 6, the precision scores for responses in Figures 2 and 4 are 16/21 (76%) and 7/12 (58%) respectively, using equal weights for all entities. Recall for both responses is 100%. It should be noted that the algorithm penalizes the precision numbers more for the error made in Figure 4 than the one made in Figure 2. As evident from the two examples, this version of the B-CUBED algorithm (using equal weights for each entity) is a precision oriented algorithm i.e. it is sensitive to precision errors.

But, for an information retrieval (IR) task, or a web search task, where an user is presented with classes of documents that pertain to the same entity, the weighting scheme assigns equal weights to each equivalence class. The weight for each entity within an equivalence class is computed by dividing the weight of the equivalence class by the number of entities in that class. Recall is calculated by assigning equal weights to each equivalence class in the truth while precision is calculated by assigning equal weights to each equivalence class in the response. For example, in Figure 2, the weighting scheme assigns a weight of 1/10 to each entity in the first equivalence class, and a weight of 1/14 to each entity in the second equivalence class, when calculating precision. Using this weighting scheme, the precision scores for responses in Figures 2 and 4 are 39/49 (79.6%) and 3/4 (75%) respectively. Recall for both responses is 100%.

Comparing these numbers to the ones obtained by using the version of the algorithm which assigns equal weights to each entity, one can see that the current version is much less sensitive to precision errors. Although the current version of the algorithm does penalize the precision numbers for the error in Figure 4 more than the error made in Figure 2, it is less severe than the earlier version.

The Model Theoretic Approach To The B-CUBED Algorithm

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 with respect to the response, i.e. p(S) is a set of subsets of S formed by intersecting S with those response sets R_i that overlap S. Let p(S) = $\{\text{P}_1, \text{P}_2, \ldots, \text{P}_\text{m}\}$ where each P_j is a subset of S.

- m_j(S) is the number of elements that are missing from each P_j relative to the key set S. Therefore,

$$\text{m}_\text{j}(\text{S}) = (|\text{S}| - |\text{P}_\text{j}|) \ .$$

Since the B-CUBED algorithm looks at the presence/absence of entities relative to each of the other entities, the number of missing entities in an entire equivalence set is calculated by adding the number of missing entities with respect to each entity in that equivalence set. Therefore, the number of missing entities for the entire set S is

$$\sum_{j=1}^{m} \sum_{\text{for each e} \in \text{P}_\text{j}} \text{m}_\text{j}(\text{S}) \ .$$

Output	MUC Algorithm	B-CUBED Algorithm (equal weights for every entity)
Example 1	P: $\frac{9}{10}$ (90%)	P: $\frac{1}{12} * [\frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{2}{7} + \frac{2}{7} + \frac{5}{7} + \frac{5}{7} + \frac{5}{7} + \frac{5}{7} + \frac{5}{7}] = \frac{16}{21}$ (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: $\frac{1}{12} * [\frac{5}{10} + \frac{5}{10} + \frac{5}{10} + \frac{5}{10} + \frac{5}{10} + \frac{2}{2} + \frac{2}{2} + \frac{5}{10} + \frac{5}{10} + \frac{5}{10} + \frac{5}{10} + \frac{5}{10}] = \frac{7}{12}$ (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 6: Scores of Both Algorithms on the Examples

The recall *error* is simply the number of missing entities divided by the number of entities in the equivalence set, i.e.,

$$\frac{\mathrm{m_j(S)}}{|\mathrm{S}|} .$$

Since the algorithm looks at each entity in an equivalence set, the recall *error* for that entire set is

$$\frac{1}{|\mathrm{S}|} \sum_{j=1}^{m} \sum_{\text{for each e} \in \mathrm{P_j}} \frac{\mathrm{m_j(S)}}{|\mathrm{S}|} .$$

Recall in turn is

$$1 - \frac{1}{|\mathrm{S}|} \sum_{j=1}^{m} \sum_{\text{for each e} \in \mathrm{P_j}} \frac{\mathrm{m_j(S)}}{|\mathrm{S}|} ,$$

which equals

$$1 - \frac{\sum_{j=1}^{m} \sum_{\text{for each e} \in \mathrm{P_j}} \mathrm{m_j(S)}}{|\mathrm{S}|^2} .$$

The whole expression can now be simplified to

$$1 - \frac{\sum_{j=1}^{m} \sum_{\text{for each e} \in \mathrm{P_j}} |\mathrm{S}| - |\mathrm{P_j}|}{|\mathrm{S}|^2} .$$

Moreover, the measure can be extended from a single key equivalence class to a set $\mathrm{T} = \{\mathrm{S_1}, \mathrm{S_2}, \ldots, \mathrm{S_n}\}$ of equivalence classes. Therefore, the recall $\mathrm{R_i}$ for an equivalence class $\mathrm{S_i}$ equals

$$\mathrm{R_i} = 1 - \frac{\sum_{\mathrm{j=1}}^{\mathrm{m}} \sum_{\text{for each e} \in \mathrm{P_{ij}}} |\mathrm{S_i}| - |\mathrm{P_{ij}}|}{|\mathrm{S_i}|^2} ,$$

where $\mathrm{P_{ij}}$ is the jth element of the partition $\mathrm{p(S_i)}$, and, hence, is a subset of $\mathrm{S_i}$.

The recall numbers calculated for each class can now be combined in various ways to produce the final recall. Different versions of the algorithm are obtained by using different combination strategies. If equal weights are assigned to each class, the version of the algorithm produced is exactly the same as the version of the informal algorithm which assigns equal weights to each class, as described in the previous section. In other words, the final recall is an average of the recall numbers for each equivalence class, i.e.,

$$\mathrm{R_T} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{R_i} .$$

To obtain the version of the informal algorithm which assigns equal weights to each entity, the final recall is computed by calculating the weighted average of the recall numbers for each equivalence class where the weights are decided by the number of entities in each class, i.e.,

$$\mathrm{R_T} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \frac{|\mathrm{S_i}|}{\sum_{\mathrm{j=1}}^{\mathrm{n}} |\mathrm{S_j}|} \mathrm{R_i} .$$

Finally, as in the case of the MUC-6 algorithm, the precision numbers are calculated by reversing the roles of the key and the response in the above formulation.

Task Relative Strengths and Weaknesses of the Two Algorithms

The MUC-6 algorithm is useful for applications/tasks that use single coreference relations at a time rather than resulting equivalence classes. One such obvious application is the MUC-6 coreference task. But, the two shortcomings of this algorithm restrict this algorithm from being useful in a number of other applications/tasks.

Cross-Document Coreference

As described in (Bagga, 98a), we had to map the cross-document coreference scoring problem to a within-

document coreference scoring problem in order to score the cross-document coreference chains output by our system. The mapping was done by creating a meta document consisting of the file names of each of the documents in the domain. The cross-document coreference chains produced by the system were then evaluated by scoring the corresponding within-document coreference chains in the meta document.

Since every filename in the meta document is part of some coreference chain, the MUC-6 algorithm yields very unintuitive results for this task. For example, consider a system which marks all the entities in the meta document as being coreferent. Suppose there are N entities in the meta document which participate in m coreference chains (i.e. the truth consists of m coreference chains). If we want to calculate the precision for this response using the MUC-6 algorithm, then the partition p(S) with respect to the key is the set of equivalence classes in the key because the response is just a single equivalence class containing all the entities in the document. Since |S| = N, the number of entities in the document, and |p(S)| = m, the number of chains in the truth, the precision, using equation 1, equals $\frac{N-m}{N-1}$. But since the number of entities in a document is usually much larger than the number of coreference chains, the fraction $\frac{N-m}{N-1}$ is close to 1 (and so the precision is close to 100%). Moreover, the recall for this response is 100%. This is very unintuitive since precision and recall are usually mutually exclusive at the limits of a task.

Therefore, a naive system which marks all the entities as being coreferent achieves very high precision and recall numbers for this task, when evaluated by the MUC-6 algorithm. In comparison, the B-CUBED algorithm (using equal weights for all entities) severely penalizes the precision numbers assigned to the naive system because its response erroneously links all the entities together, thereby making a large number of entities coreferent that should not be. However, similar to the MUC-6 algorithm, the recall computed by the B-CUBED algorithm for the same response equals 100%.

Information Extraction

In an information extraction task, information about every entity in a document is important. A response which erroneously puts an entity into the wrong coreference chain, or a response which erroneously links two coreference chains can greatly affect the performance of an information extraction system. As a result, an information extraction task requires a precision-oriented coreference scoring algorithm. Therefore, the version of the B-CUBED algorithm which assigns equal weights to each entity is an appropriate algorithm for this task. In contrast, the MUC-6 algorithm is not very useful for such a task because it considers all precision errors to be equal.

Information Retrieval and Web Search Engines

Consider a web search task where a web search engine retrieves documents and presents to the user classes of documents based upon the topic, or the mention of a particular entity. Compared to an information extraction task, this task is less likely to be affected by an error which puts a document in the wrong class, or an error which erroneously links two classes together. However, the latter kind of error should still be penalized more than the former. Therefore, an appropriate algorithm for this task would be the version of the B-CUBED algorithm which assigns equal weights to each equivalence class. Once again, the MUC-6 algorithm is not very useful for this task because it considers all precision errors to be equal.

Conclusions

In this paper we have described two different scoring coreference scoring algorithms. The (Vilain, 95) algorithm suits applications that concern single coreference relations at a time rather than the resulting equivalence classes. Our observation is that some coreference links introduce more error into the resulting equivalence classes that others, and it is important to be able to notice that difference through scoring. Our B-CUBED coreference scoring algorithm overcomes this shortcoming of the MUC-6 algorithm. Applications that benefit from such a scoring algorithm include cross-document coreference, information extraction, and information retrieval.

Acknowledgments

The authors would like to thank Alan W. Biermann of Duke University who initially put forth the main idea of the B-CUBED algorithm.

The first author would also like to thank Greg Keim, and Pavan K. Desikan of Duke University for the insightful discussions during the development of the B-CUBED algorithm.

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

References

Bagga, Amit. How Much Processing Is Required for Cross-Document Coreference?, *this volume.*

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

Proceedings of the Sixth Message Understanding Conference (MUC-6), November 1995, San Mateo: Morgan Kaufmann.