Machine Learning for Coreference Resolution: From Local Classification to Global Ranking

Vincent Ng
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
vince@hlt.utdallas.edu

Abstract

In this paper, we view coreference resolution as a problem of ranking candidate partitions generated by different coreference systems. We propose a set of partition-based features to learn a ranking model for distinguishing good and bad partitions. Our approach compares favorably to two state-of-the-art coreference systems when evaluated on three standard coreference data sets.

1 Introduction

Recent research in coreference resolution — the problem of determining which noun phrases (NPs) in a text or dialogue refer to which real-world entity — has exhibited a shift from knowledge-based approaches to data-driven approaches, yielding learning-based coreference systems that rival their hand-crafted counterparts in performance (e.g., Soon et al. (2001), Ng and Cardie (2002b), Strube et al. (2002), Yang et al. (2003), Luo et al. (2004)). The central idea behind the majority of these learning-based approaches is to recast coreference resolution as a binary classification task. Specifically, a classifier is first trained to determine whether two NPs in a document are co-referring or not. A separate clustering mechanism then coordinates the possibly contradictory pairwise coreference classification decisions and constructs a partition on the given set of NPs, with one cluster for each set of coreferent NPs.

Though reasonably successful, this "standard" approach is not as robust as one may think. First, design decisions such as the choice of the learning algorithm and the clustering procedure are apparently critical to system performance, but are often made in an ad-hoc and unprincipled manner that may be suboptimal from an empirical point of view.

Second, this approach makes no attempt to search through the space of possible partitions when given a set of NPs to be clustered, employing instead a greedy clustering procedure to construct a partition that may be far from optimal.

Another potential weakness of this approach concerns its inability to directly optimize for clustering-level accuracy: the coreference classifier is trained and optimized independently of the clustering procedure to be used, and hence improvements in classification accuracy do not guarantee corresponding improvements in clustering-level accuracy.

Our goal in this paper is to improve the robustness of the standard approach by addressing the above weaknesses. Specifically, we propose the following procedure for coreference resolution: given a set of NPs to be clustered, (1) use n pre-selected learning-based coreference systems to generate n candidate partitions of the NPs, and then (2) apply an *automatically acquired ranking model* to rank these candidate hypotheses, selecting the best one to be the final partition. The key features of this approach are:

Minimal human decision making. In contrast to the standard approach, our method obviates, to a large extent, the need to make tough or potentially suboptimal design decisions.[1] For instance, if we

[1] We still need to determine the n coreference systems to be employed in our framework, however. Fortunately, the choice of n is flexible, and can be as large as we want subject to the

Proceedings of the 43rd Annual Meeting of the ACL, pages 157–164,
Ann Arbor, June 2005.

cannot decide whether learner A is better to use than learner B in a coreference system, we can simply create two copies of the system with one employing A and the other B, and then add both into our pre-selected set of coreference systems.

Generation of multiple candidate partitions. Although an exhaustive search for the best partition is not computationally feasible even for a document with a moderate number of NPs, our approach explores a larger portion of the search space than the standard approach via generating multiple hypotheses, making it possible to find a potentially better partition of the NPs under consideration.

Optimization for clustering-level accuracy via ranking. As mentioned above, the standard approach trains and optimizes a coreference classifier without necessarily optimizing for clustering-level accuracy. In contrast, we attempt to optimize our ranking model with respect to the target coreference scoring function, essentially by training it in such a way that a higher scored candidate partition (according to the scoring function) would be assigned a higher rank (see Section 3.2 for details).

Perhaps even more importantly, our approach provides *a general framework* for coreference resolution. Instead of committing ourselves to a particular resolution method as in previous approaches, our framework makes it possible to leverage the strengths of different methods by allowing them to participate in the generation of candidate partitions.

We evaluate our approach on three standard coreference data sets using two different scoring metrics. In our experiments, our approach compares favorably to two state-of-the-art coreference systems adopting the standard machine learning approach, outperforming them by as much as 4–7% on the three data sets for one of the performance metrics.

2 Related Work

As mentioned before, our approach differs from the standard approach primarily by (1) explicitly learning a ranker and (2) optimizing for clustering-level accuracy. In this section we will focus on discussing related work along these two dimensions.

Ranking candidate partitions. Although we are not aware of any previous attempt on training a ranking model using global features of an NP partition, there is some related work on partition ranking where the score of a partition is computed via a heuristic function of the probabilities of its NP pairs being coreferent.[2] For instance, Harabagiu et al. (2001) introduce a greedy algorithm for finding the highest-scored partition by performing a beam search in the space of possible partitions. At each step of this search process, candidate partitions are ranked based on their heuristically computed scores.

Optimizing for clustering-level accuracy. Ng and Cardie (2002a) attempt to optimize their rule-based coreference classifier for clustering-level accuracy, essentially by finding a subset of the learned rules that performs the best on held-out data with respect to the target coreference scoring program. Strube and Müller (2003) propose a similar idea, but aim instead at finding a subset of the available features with which the resulting coreference classifier yields the best clustering-level accuracy on held-out data. To our knowledge, our work is the first attempt to optimize a ranker for clustering-level accuracy.

3 A Ranking Approach to Coreference

Our ranking approach operates by first dividing the available training texts into two disjoint subsets: a training subset and a held-out subset. More specifically, we first train each of our n pre-selected coreference systems on the documents in the training subset, and then use these resolvers to generate n candidate partitions for each text in the held-out subset from which a ranking model will be learned. Given a test text, we use our n coreference systems to create n candidate partitions as in training, and select the highest-ranked partition according to the ranking model to be the final partition.[3] The rest of this section describes how we select these n learning-based coreference systems and acquire the ranking model.

3.1 Selecting Coreference Systems

A learning-based coreference system can be defined by four elements: the *learning algorithm* used to train the coreference classifier, the *method of creating training instances* for the learner, the *feature set*

available computing resources.

[2] Examples of such scoring functions include the Dempster-Shafer rule (see Kehler (1997) and Bean and Riloff (2004)) and its variants (see Harabagiu et al. (2001) and Luo et al. (2004)).

[3] The ranking model breaks ties randomly.

used to represent a training or test instance, and the *clustering algorithm* used to coordinate the coreference classification decisions. Selecting a coreference system, then, is a matter of instantiating these elements with specific values.

Now we need to define the set of allowable values for each of these elements. In particular, we want to define them in such a way that the resulting coreference systems can potentially generate good candidate partitions. Given that machine learning approaches to the problem have been promising, our choices will be guided by previous learning-based coreference systems, as described below.

Training instance creation methods. A training instance represents two NPs, NP_i and NP_j, having a class value of COREFERENT or NOT COREFERENT depending on whether the NPs co-refer in the associated text. We consider three previously-proposed methods of creating training instances.

In **McCarthy and Lehnert's method**, a positive instance is created for each anaphoric NP paired with each of its antecedents, and a negative instance is created by pairing each NP with each of its preceding non-coreferent noun phrases. Hence, the number of instances created by this method is quadratic in the number of NPs in the associated text. The large number of instances can potentially make the training process inefficient.

In an attempt to reduce the training time, **Soon et al.'s method** creates a smaller number of training instances than McCarthy and Lehnert's. Specifically, a positive instance is created for each anaphoric NP, NP_j, and its closest antecedent, NP_i; and a negative instance is created for NP_j paired with each of the intervening NPs, NP_{i+1}, NP_{i+2}, ..., NP_{j-1}.

Unlike Soon et al., **Ng and Cardie's method** generates a positive instance for each anaphoric NP and its *most confident* antecedent. For a non-pronominal NP, the most confident antecedent is assumed to be its closest non-pronominal antecedent. For pronouns, the most confident antecedent is simply its closest preceding antecedent. Negative instances are generated as in Soon et al.'s method.

Feature sets. We employ two feature sets for representing an instance, as described below.

Soon et al.'s feature set consists of 12 surface-level features, each of which is computed based on one or both NPs involved in the instance. The features can be divided into four groups: lexical, grammatical, semantic, and positional. Space limitations preclude a description of these features. Details can be found in Soon et al. (2001).

Ng and Cardie expand Soon et al.'s feature set from 12 features to a deeper set of 53 to allow more complex NP string matching operations as well as finer-grained syntactic and semantic compatibility tests. See Ng and Cardie (2002b) for details.

Learning algorithms. We consider three learning algorithms, namely, the **C4.5** decision tree induction system (Quinlan, 1993), the **RIPPER** rule learning algorithm (Cohen, 1995), and **maximum entropy classification** (Berger et al., 1996). The classification model induced by each of these learners returns a number between 0 and 1 that indicates the likelihood that the two NPs under consideration are coreferent. In this work, NP pairs with class values above 0.5 are considered COREFERENT; otherwise the pair is considered NOT COREFERENT.

Clustering algorithms. We employ three clustering algorithms, as described below.

The **closest-first clustering** algorithm selects as the antecedent of NP_j its *closest* preceding coreferent NP. If no such NP exists, then NP_j is assumed to be non-anaphoric (i.e., no antecedent is selected).

On the other hand, the **best-first clustering** algorithm selects as the antecedent of NP_j the closest NP with the highest coreference likelihood value from its set of preceding coreferent NPs. If this set is empty, then no antecedent is selected for NP_j. Since the *most likely* antecedent is chosen for each NP, best-first clustering may produce partitions with higher precision than closest-first clustering.

Finally, in **aggressive-merge clustering**, each NP is merged with *all* of its preceding coreferent NPs. Since more merging occurs in comparison to the previous two algorithms, aggressive-merge clustering may yield partitions with higher recall.

Table 1 summarizes the previous work on coreference resolution that employs the learning algorithms, clustering algorithms, feature sets, and instance creation methods discussed above. With three learners, three training instance creation methods, two feature sets, and three clustering algorithms, we can produce 54 coreference systems in total.

Learning algorithm	Decision tree learners (C4.5/C5/CART)	Aone and Bennett (1995), McCarthy and Lehnert (1995), Soon et al. (2001), Strube et al. (2002), Strube and Müller (2003), Yang et al. (2003)
	RIPPER	Ng and Cardie (2002b)
	Maximum entropy	Kehler (1997), Morton (2000), Luo et al. (2004)
Instance creation method	McCarthy and Lehnert's	McCarthy and Lehnert (1995), Aone and Bennett (1995)
	Soon et al.'s	Soon et al. (2001), Strube et al. (2002), Iida et al. (2003)
	Ng and Cardie's	Ng and Cardie (2002b)
Feature set	Soon et al.'s	Soon et al. (2001)
	Ng and Cardie's	Ng and Cardie (2002b)
Clustering algorithm	Closest-first	Soon et al. (2001), Strube et al. (2002)
	Best-first	Aone and Bennett (1995), Ng and Cardie (2002b), Iida et al. (2003)
	Aggressive-merge	McCarthy and Lehnert (1995)

Table 1: Summary of the previous work on coreference resolution that employs the learning algorithms, the clustering algorithms, the feature sets, and the training instance creation methods discussed in Section 3.1.

3.2 Learning to Rank Candidate Partitions

We train an SVM-based ranker for ranking candidate partitions by means of Joachims' (2002) SVM^{light} package, with all the parameters set to their default values. To create training data, we first generate 54 candidate partitions for each text in the held-out subset as described above and then convert each partition into a training instance consisting of a set of *partition-based* features and *method-based* features.

Partition-based features are used to characterize a candidate partition and can be derived directly from the partition itself. Following previous work on using *global* features of candidate structures to learn a ranking model (Collins, 2002), the global (i.e., partition-based) features we consider here are simple functions of the *local* features that capture the relationship between NP pairs.

Specifically, we define our partition-based features in terms of the features in the Ng and Cardie (N&C) feature set (see Section 3.1) as follows. First, let us assume that f_i is the i-th nominal feature in N&C's feature set and v_{ij} is the j-th possible value of f_i. Next, for each i and j, we create two partition-based features, P_{ij0} and P_{ij1}. P_{ij0} is computed over the set of *coreferent* NP pairs (with respect to the candidate partition), denoting the probability of encountering $f_i = v_{ij}$ in this set when the pairs are represented as attribute-value vectors using N&C's features. On the other hand, P_{ij1} is computed over the set of *non-coreferent* NP pairs (with respect to the candidate partition), denoting the probability of encountering $f_i = v_{ij}$ in this set when the pairs are represented as attribute-value vectors using N&C's features. One partition-based feature, for instance, would denote the probability that two NPs residing in the same cluster have incompatible gender values. Intuitively, a good NP partition would have a low probability value for this feature. So, having these partition-based features can potentially help us distinguish good and bad candidate partitions.

Method-based features, on the other hand, are used to encode the identity of the coreference system that generated the candidate partition under consideration. Specifically, we have one method-based feature representing each pre-selected coreference system. The feature value is 1 if the corresponding coreference system generated the candidate partition and 0 otherwise. These features enable the learner to learn how to distinguish good and bad partitions based on the systems that generated them, and are particularly useful when some coreference systems perform consistently better than the others.

Now, we need to compute the "class value" for each training instance, which is a positive integer denoting the rank of the corresponding partition among the 54 candidates generated for the training document under consideration. Recall from the introduction that we want to train our ranking model so that higher scored partitions according to the target coreference scoring program are ranked higher. To this end, we compute the rank of each candidate partition as follows. First, we apply the target scoring program to score each candidate partition against the correct partition derived from the training text. We then assign rank i to the i-th lowest scored partition.[4] Effectively, the learning algorithm learns what a good partition is from the scoring program.

[4]Two partitions with the same score will have the same rank.

	Training Corpus		Test Corpus	
	# Docs	# Tokens	# Docs	# Tokens
BNEWS	216	67470	51	18357
NPAPER	76	71944	17	18174
NWIRE	130	85688	29	20528

Table 2: Statistics for the ACE corpus.

4 Evaluation

4.1 Experimental Setup

For evaluation purposes, we use the ACE (Automatic Content Extraction) coreference corpus, which is composed of three data sets created from three different news sources, namely, broadcast news (BNEWS), newspaper (NPAPER), and newswire (NWIRE).[5] Statistics of these data sets are shown in Table 2. In our experiments, we use the training texts to acquire coreference classifiers and evaluate the resulting systems on the test texts with respect to two commonly-used coreference scoring programs: the MUC scorer (Vilain et al., 1995) and the B-CUBED scorer (Bagga and Baldwin, 1998).

4.2 Results Using the MUC Scorer

Baseline systems. We employ as our baseline systems two existing coreference resolvers: our duplication of the Soon et al. (2001) system and the Ng and Cardie (2002b) system. Both resolvers adopt the standard machine learning approach and therefore can be characterized using the four elements discussed in Section 3.1. Specifically, Soon et al.'s system employs a decision tree learner to train a coreference classifier on instances created by Soon's method and represented by Soon's feature set, coordinating the classification decisions via closest-first clustering. Ng and Cardie's system, on the other hand, employs RIPPER to train a coreference classifier on instances created by N&C's method and represented by N&C's feature set, inducing a partition on the given NPs via best-first clustering.

The baseline results are shown in rows 1 and 2 of Table 3, where performance is reported in terms of recall, precision, and F-measure. As we can see, the N&C system outperforms the Duplicated Soon system by about 2-6% on the three ACE data sets.

[5]See `http://www.itl.nist.gov/iad/894.01/tests/ace` for details on the ACE research program.

Our approach. Recall that our approach uses labeled data to train both the coreference classifiers and the ranking model. To ensure a fair comparison of our approach with the baselines, we do not rely on additional labeled data for learning the ranker; instead, we use half of the training texts for training classifiers and the other half for ranking purposes. Results using our approach are shown in row 3 of Table 3. Our ranking model, when trained to optimize for F-measure using both partition-based features and method-based features, consistently provides substantial gains in F-measure over both baselines. In comparison to the stronger baseline (i.e., N&C), F-measure increases by 7.4, 7.2, and 4.6 for the BNEWS, NPAPER, and NWIRE data sets, respectively. Perhaps more encouragingly, gains in F-measure are accompanied by simultaneous increase in recall and precision for all three data sets.

Feature contribution. In an attempt to gain additional insight into the contribution of partition-based features and method-based features, we train our ranking model using each type of features in isolation. Results are shown in rows 4 and 5 of Table 3. For the NPAPER and NWIRE data sets, we still see gains in F-measure over both baseline systems when the model is trained using either type of features. The gains, however, are smaller than those observed when the two types of features are applied in combination. Perhaps surprisingly, the results for BNEWS do not exhibit the same trend as those for the other two data sets. Here, the method-based features alone are strongly predictive of good candidate partitions, yielding even slightly better performance than when both types of features are applied. Overall, however, these results seem to suggest that both partition-based and method-based features are important to learning a good ranking model.

Random ranking. An interesting question is: how much does supervised ranking help? If all of our candidate partitions are of very high quality, then ranking will not be particularly important because choosing any of these partitions may yield good results. To investigate this question, we apply a *random* ranking model, which randomly selects a candidate partition for each test text. Row 6 of Table 3 shows the results (averaged over five runs) when the random ranker is used in place of the supervised

	System Variation	BNEWS R	BNEWS P	BNEWS F	NPAPER R	NPAPER P	NPAPER F	NWIRE R	NWIRE P	NWIRE F
1	Duplicated Soon et al. baseline	52.7	47.5	50.0	63.3	56.7	59.8	48.7	40.9	44.5
2	Ng and Cardie baseline	56.5	58.6	57.5	57.1	68.0	62.1	43.1	59.9	50.1
3	Ranking framework	62.2	67.9	64.9	67.4	71.4	69.3	50.1	60.3	54.7
4	Partition-based features only	54.5	55.5	55.0	66.3	63.0	64.7	50.7	51.2	51.0
5	Method-based features only	62.0	68.5	65.1	67.5	61.2	64.2	51.1	49.9	50.5
6	Random ranking model	48.6	54.8	51.5	57.4	63.3	60.2	40.3	44.3	42.2
7	Perfect ranking model	66.0	69.3	67.6	70.4	71.2	70.8	56.6	59.7	58.1

Table 3: Results for the three ACE data sets obtained via the MUC scoring program.

ranker. In comparison to the results in row 3, we see that the supervised ranker surpasses its random counterpart by about 9-13% in F-measure, implying that ranking plays an important role in our approach.

Perfect ranking. It would be informative to see whether our ranking model is performing at its upper limit, because further performance improvement beyond this point would require enlarging our set of candidate partitions. So, we apply a *perfect* ranking model, which uses an oracle to choose the best candidate partition for each test text. Results in row 7 of Table 3 indicate that our ranking model performs at about 1-3% below the perfect ranker, suggesting that we can further improve coreference performance by improving the ranking model.

4.3 Results Using the B-CUBED Scorer

Baseline systems. In contrast to the MUC results, the B-CUBED results for the two baseline systems are mixed (see rows 1 and 2 of Table 4). Specifically, while there is no clear winner for the NWIRE data set, N&C performs better on BNEWS but worse on NPAPER than the Duplicated Soon system.

Our approach. From row 3 of Table 4, we see that our approach achieves small but consistent improvements in F-measure over both baseline systems. In comparison to the better baseline, F-measure increases by 0.1, 1.1, and 2.0 for the BNEWS, NPAPER, and NWIRE data sets, respectively.

Feature contribution. Unlike the MUC results, using more features to train the ranking model does not always yield better performance with respect to the B-CUBED scorer (see rows 3-5 of Table 4). In particular, the best result for BNEWS is achieved using only method-based features, whereas the best result for NPAPER is obtained using only partition-based features. Nevertheless, since neither type of features offers consistently better performance than the other, it still seems desirable to apply the two types of features in combination to train the ranker.

Random ranking. Comparing rows 3 and 6 of Table 4, we see that the supervised ranker yields a non-trivial improvement of 2-3% in F-measure over the random ranker for the three data sets. Hence, ranking still plays an important role in our approach with respect to the B-CUBED scorer despite its modest performance gains over the two baseline systems.

Perfect ranking. Results in rows 3 and 7 of Table 4 indicate that the supervised ranker underperforms the perfect ranker by about 5% for BNEWS and 3% for both NPAPER and NWIRE in terms of F-measure, suggesting that the supervised ranker still has room for improvement. Moreover, by comparing rows 1-2 and 7 of Table 4, we can see that the perfect ranker outperforms the baselines by less than 5%. This is essentially an upper limit on how much our approach can improve upon the baselines given the current set of candidate partitions. In other words, the performance of our approach is limited in part by the quality of the candidate partitions, more so with B-CUBED than with the MUC scorer.

5 Discussion

Two questions naturally arise after examining the above results. First, which of the 54 coreference systems generally yield superior results? Second, why is the same set of candidate partitions scored so differently by the two scoring programs?

To address the first question, we take the 54 coreference systems that were trained on half of the available training texts (see Section 4) and apply them to the three ACE test data sets. Table 5 shows the best-performing resolver for each test set and scoring program combination. Interestingly, with respect to the

	System Variation	BNEWS R	BNEWS P	BNEWS F	NPAPER R	NPAPER P	NPAPER F	NWIRE R	NWIRE P	NWIRE F
1	Duplicated Soon et al. baseline	53.4	78.4	63.5	58.0	75.4	65.6	56.0	75.3	64.2
2	Ng and Cardie baseline	59.9	72.3	65.5	61.8	64.9	63.3	62.3	66.7	64.4
3	Ranking framework	57.0	77.1	65.6	62.8	71.2	66.7	59.3	75.4	66.4
4	Partition-based features only	55.0	79.1	64.9	61.3	74.7	67.4	57.1	76.8	65.5
5	Method-based features only	63.1	69.8	65.8	58.4	75.2	65.8	58.9	75.5	66.1
6	Random ranking model	52.5	79.9	63.4	58.4	69.2	63.3	54.3	77.4	63.8
7	Perfect ranking model	64.5	76.7	70.0	61.3	79.1	69.1	63.2	76.2	69.1

Table 4: Results for the three ACE data sets obtained via the B-CUBED scoring program.

MUC scorer, the best performance on the three data sets is achieved by the same resolver. The results with respect to B-CUBED are mixed, however.

For each resolver shown in Table 5, we also compute the average rank of the partitions generated by the resolver for the corresponding test texts.[6] Intuitively, a resolver that consistently produces good partitions (relative to other candidate partitions) would achieve a low average rank. Hence, we can infer from the fairly high rank associated with the top B-CUBED resolvers that they do not perform consistently better than their counterparts.

Regarding our second question of why the same set of candidate partitions is scored differently by the two scoring programs, the reason can be attributed to two key algorithmic differences between these scorers. First, while the MUC scorer only rewards correct identification of coreferent links, B-CUBED additionally rewards successful recognition of non-coreference relationships. Second, the MUC scorer applies the same penalty to each erroneous merging decision, whereas B-CUBED penalizes erroneous merging decisions involving two large clusters more heavily than those involving two small clusters.

Both of the above differences can potentially cause B-CUBED to assign a narrower range of F-measure scores to each set of 54 candidate partitions than the MUC scorer, for the following reasons. First, our candidate partitions in general agree more on singleton clusters than on non-singleton clusters. Second, by employing a non-uniform penalty function B-CUBED effectively removes a bias inherent in the MUC scorer that leads to under-penalization of partitions in which entities are over-clustered.

Nevertheless, our B-CUBED results suggest that (1) despite its modest improvement over the baselines, our approach offers robust performance across the data sets; and (2) we could obtain better scores by improving the ranking model and expanding our set of candidate partitions, as elaborated below.

To improve the ranking model, we can potentially (1) design new features that better characterize a candidate partition (e.g., features that measure the size and the internal cohesion of a cluster), and (2) reserve more labeled data for training the model. In the latter case we may have less data for training coreference classifiers, but at the same time we can employ weakly supervised techniques to bootstrap the classifiers. Previous attempts on bootstrapping coreference classifiers have only been mildly successful (e.g., Müller et al. (2002)), and this is also an area that deserves further research.

To expand our set of candidate partitions, we can potentially incorporate more high-performing coreference systems into our framework, which is flexible enough to accommodate even those that adopt knowledge-based (e.g., Harabagiu et al. (2001)) and unsupervised approaches (e.g., Cardie and Wagstaff (1999), Bean and Riloff (2004)). Of course, we can also expand our pre-selected set of coreference systems via incorporating additional learning algorithms, clustering algorithms, and feature sets. Once again, we may use previous work to guide our choices. For instance, Iida et al. (2003) and Zelenko et al. (2004) have explored the use of SVM, voted perceptron, and logistic regression for training coreference classifiers. McCallum and Wellner (2003) and Zelenko et al. (2004) have employed graph-based partitioning algorithms such as correlation clustering (Bansal et al., 2002). Finally, Strube et al. (2002) and Iida et al. (2003) have proposed new edit-distance-based string-matching features and centering-based features, respectively.

[6] The rank of a partition is computed in the same way as in Section 3.2, except that we now adopt the common convention of assigning rank i to the i-th *highest* scored partition.

Test Set	Scoring Program	Average Rank	Coreference System			
			Instance Creation Method	Feature Set	Learner	Clustering Algorithm
BNEWS	MUC	7.2549	McCarthy and Lehnert's	Ng and Cardie's	C4.5	aggressive-merge
	BCUBED	16.9020	McCarthy and Lehnert's	Ng and Cardie's	C4.5	aggressive-merge
NPAPER	MUC	1.4706	McCarthy and Lehnert's	Ng and Cardie's	C4.5	aggressive-merge
	B-CUBED	9.3529	Soon et al.'s	Soon et al.'s	RIPPER	closest-first
NWIRE	MUC	7.7241	McCarthy and Lehnert's	Ng and Cardie's	C4.5	aggressive-merge
	B-CUBED	13.1379	Ng and Cardie's	Ng and Cardie's	MaxEnt	closest-first

Table 5: The coreference systems that achieved the highest F-measure scores for each test set and scorer combination. The average rank of the candidate partitions produced by each system for the corresponding test set is also shown.

Acknowledgments

We thank the three anonymous reviewers for their valuable comments on an earlier draft of the paper.

References

C. Aone and S. W. Bennett. 1995. Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proc. of the ACL*, pages 122–129.

A. Bagga and B. Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *Proc. of COLING-ACL*, pages 79–85.

N. Bansal, A. Blum, and S. Chawla. 2002. Correlation clustering. In *Proc. of FOCS*, pages 238–247.

D. Bean and E. Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. In *Proc. of HLT/NAACL*, pages 297–304.

A. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

C. Cardie and K. Wagstaff. 1999. Noun phrase coreference as clustering. In *Proc. of EMNLP/VLC*, pages 82–89.

W. Cohen. 1995. Fast effective rule induction. In *Proc. of ICML*, pages 115–123.

M. Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*, pages 1–8.

S. Harabagiu, R. Bunescu, and S. Maiorano. 2001. Text and knowledge mining for coreference resolution. In *Proc. of NAACL*, pages 55–62.

R. Iida, K. Inui, H. Takamura, and Y. Matsumoto. 2003. Incorporating contextual cues in trainable models for coreference resolution. In *Proc. of the EACL Workshop on The Computational Treatment of Anaphora*.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proc. of KDD*, pages 133–142.

A. Kehler. 1997. Probabilistic coreference in information extraction. In *Proc. of EMNLP*, pages 163–173.

X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *Proc. of the ACL*, pages 136–143.

A. McCallum and B. Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proc. of the IJCAI Workshop on Information Integration on the Web*.

J. McCarthy and W. Lehnert. 1995. Using decision trees for coreference resolution. In *Proc. of the IJCAI*, pages 1050–1055.

T. Morton. 2000. Coreference for NLP applications. In *Proc. of the ACL*.

C. M¨uller, S. Rapp, and M. Strube. 2002. Applying co-training to reference resolution. In *Proc. of the ACL*, pages 352–359.

V. Ng and C. Cardie. 2002a. Combining sample selection and error-driven pruning for machine learning of coreference rules. In *Proc. of EMNLP*, pages 55–62.

V. Ng and C. Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *Proc. of the ACL*, pages 104–111.

J. R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

W. M. Soon, H. T. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

M. Strube and C. M¨uller. 2003. A machine learning approach to pronoun resolution in spoken dialogue. In *Proc. of the ACL*, pages 168–175.

M. Strube, S. Rapp, and C. M¨uller. 2002. The influence of minimum edit distance on reference resolution. In *Proc. of EMNLP*, pages 312–319.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proc. of the Sixth Message Understanding Conference (MUC-6)*, pages 45–52.

X. Yang, G. D. Zhou, J. Su, and C. L. Tan. 2003. Coreference resolution using competitive learning approach. In *Proc. of the ACL*, pages 176–183.

D. Zelenko, C. Aone, and J. Tibbetts. 2004. Coreference resolution for information extraction. In *Proc. of the ACL Workshop on Reference Resolution and its Applications*, pages 9–16.