Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI), Barcelona, Spain, July 2011, pp. 1884--1889.

Ensemble-Based Coreference Resolution

Altaf Rahman and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{altaf,vince}@hlt.utdallas.edu

Abstract

We investigate new methods for creating and applying ensembles for coreference resolution. While existing ensembles for coreference resolution are typically created using different learning algorithms, clustering algorithms or training sets, we harness recent advances in coreference modeling and propose to create our ensemble from a variety of supervised coreference models. However, the presence of pairwise and non-pairwise coreference models in our ensemble presents a challenge to its application: it is not immediately clear how to combine the coreference decisions made by these models. We investigate different methods for applying a model-heterogeneous ensemble for coreference resolution. Empirical results on the ACE data sets demonstrate the promise of ensemble approaches: all ensemble-based systems significantly outperform the best member of the ensemble.

1 Introduction

Noun phrase (NP) coreference resolution is the task of determining which NPs in a text refer to the same real-world entity. Recent years have seen the proposal of a number of supervised coreference models. While recently-proposed models all report better performance than earlier ones, an intriguing question is: is there a single model that is truly better than the others in that it addresses all the weaknesses inherent in other models, or do existing models have complementary strengths and weaknesses? We hypothesize that the latter is true, and this motivates our investigation of ensemble approaches to coreference resolution. More specifically, our goal in this paper is to investigate new methods for *creating* and *applying* ensembles of supervised coreference systems.

As far as *creating* ensembles is concerned, we will explore two methods. The first method, as suggested above, involves employing different supervised *models* of coreference resolution. In addition to employing the influential *mention-pair* model [Aone and Bennett, 1995; McCarthy and Lehnert, 1995], which is a classifier that determines whether two NPs are co-referring, we will consider two recently-developed models, a ranking-based model known as the *mention-ranking* model [Denis and Baldridge, 2008], and a hybrid entity- and ranking-based model known as the *cluster-ranking* model [Rahman and Ng, 2011]. Employing different coreference models to create ensembles bears resemblance to Pang and Fan's [2009] approach, where an ensemble of pairwise models is applied to Chinese coreference resolution, but contrasts with the vast majority of existing approaches, where an ensemble of coreference systems is typically created by employing different learning algorithms [Munson *et al.*, 2005] or clustering algorithms [Ng, 2005], or perturbing the training set using meta-learning techniques such as bagging and boosting [Ng and Cardie, 2003; Kouchnir, 2004; Vemulapalli *et al.*, 2009].

Our second method for creating ensembles involves employing different *feature sets* to train coreference models. The features used by the majority of existing coreference systems are derived primarily from Ng and Cardie's [2002] and Bengtson and Roth's [2008]'s feature sets, which comprise a fairly large number of conventional features. In addition to employing a conventional feature set, we also experiment with a feature set that abandons some of these conventional features (e.g., gender and number agreement) and instead includes a kind of features that is not exploited by the majority of existing supervised coreference models: *word pairs* that are composed of the head nouns of the two NPs under consideration. Intuitively, these word pairs contain useful information. For example, they may help improve the *precision* of a model, by allowing a learner to learn that "it" only has a moderate probability of being anaphoric, and that "the contrary" taken from the phrase "on the contrary" is never anaphoric. They may also help improve its recall, by allowing the learner to determine, for instance, that "airline" and "carrier" can be coreferent. Hence, they offer a convenient means to attack one of the major problems in coreference research: identifying lexically dissimilar but semantically related pairs of common nouns. We believe that we can increase the diversity of the ensemble by including models trained on different feature sets, including conventional features as well as these word-pair-based features.

Another contribution of our work lies in our investigation of different methods for *applying* ensembles for coreference resolution. The major challenge in applying ensembles arises from the heterogeneity of coreference models in our ensemble. Specifically, since our ensemble comprises both pairwise classifiers (e.g., the mention-pair model) and non-pairwise models (e.g., the cluster-ranking model), it is not immediately clear how to combine the coreference decisions made by different models. This contrasts with existing ensemble approaches, where all members of the ensemble employ pairwise coreference models. The use of pairwise models across the board makes it easy to apply the resulting ensemble: the ensemble determines whether two NPs are co-referring simply by tallying the votes casted by its members. Experimental results on the ACE 2005 data sets demonstrate the promise of ensemble approaches to coreference resolution: all of our ensemble-based systems significantly outperform the cluster-ranking model, the best-performing coreference model in the ensemble; in particular, the best ensemble-based system outperforms the cluster-ranking model by 4 points in F-measure.

The rest of the paper is organized as follows. Sections 2 and 3 describe our methods for *creating* and *applying* ensembles for coreference resolution, respectively. We evaluate our ensemble approaches in Section 4 and conclude in Section 5.

2 Ensemble Creation

We create an ensemble of coreference systems by employing different *supervised models* and different *feature sets*. Below we provide the details of these models and feature sets.

2.1 Feature Sets

We use three feature sets to characterize a pair of NPs: *Conventional*, *Lexical*, and *Combined*.

The *Conventional* Feature Set

Our *Conventional* feature set consists of 39 commonly-used coreference features, which are described in Table 1 of Rahman and Ng [2011]. Linguistically, our features can be divided into four categories: string-matching, grammatical, semantic, and positional. These features can also be categorized based on whether they are relational or not. Specifically, relational features capture the relationship between the two NPs under consideration, whereas non-relational features are computed using exactly one of the two NPs. Since space limitations preclude a description of these features, we refer the reader to Rahman and Ng [2011] for details.

The *Lexical* Feature Set

Recall that we propose to use the word pairs collected from coreference-annotated documents as features for training a supervised model. Our second feature set, the *Lexical* feature set, comprises primarily features derived from these word pairs (henceforth *lexical* features).

For lexical features to be effective, we need to address *data sparseness*, as many word pairs in the training data may not appear in the test data. To improve generalization, we extract different kinds of lexical features from an annotated text.

Before computing these features, we first preprocess a *training* text by randomly replacing 10% of its common nouns with the label UNSEEN. If an NP NP_k is replaced with UNSEEN, all NPs that have the same string as NP_k will also be replaced with UNSEEN. A *test* text is preprocessed differently: we simply replace all NPs whose strings are not seen in the training data with UNSEEN. Hence, artificially creating UNSEEN labels from a training text will allow a learner to learn how to handle unseen words in a test text, potentially improving generalizability.

Below are the four types of features in the *Lexical* feature set. Each feature is computed based on two NPs, NP_j and NP_k.

Unseen feature. If both NP_j and NP_k are UNSEEN, we determine whether they are the same string. If so, we create an UNSEEN-SAME feature; otherwise, we create an UNSEEN-DIFF feature. If only one of them is UNSEEN, no feature will be created from any of the four feature groups, since features involving an UNSEEN NP are likely to be misleading for a learner in the sense that they may yield incorrect generalizations from the training set.

Lexical feature. We create a lexical feature between NP_j and NP_k, which is an ordered pair comprising their heads. For a pronoun or a common noun, the head is taken to be the last word of the NP; for a proper name, the head is the entire NP.

Semi-lexical features. These features aim to improve generalizability. Specifically, if exactly one of NP_j and NP_k is tagged as a named entity (NE) by the Stanford NE recognizer [Finkel *et al.*, 2005], we create a semi-lexical feature that is identical to the lexical feature described above, except that the NE is replaced with its NE label (i.e., PERSON, LOCATION, ORGANIZATION). If both NPs are NEs, we check whether they are the same string. If so, we create the feature *NE*-SAME, where *NE* is replaced with the corresponding NE label. Otherwise, we check whether they have the same NE tag *and* a word-subset match (i.e., whether all word tokens in one NP appears in the other's list of tokens). If so, we create the feature *NE*-SUBSAME, where *NE* is replaced with their NE label. Otherwise, we create a feature that is the concatenation of the NE labels of the two NPs.

Old features. To further improve generalizability, we incorporate two easy-to-compute *Conventional* features: *name alias*, which determines whether one NP is an acronym or an abbreviation of the other, and *distance*, which encodes the distance between the two NPs in sentences.

The *Combined* Feature Set

The *Combined* feature set is the union of the *Conventional* features and the *Lexical* features.

2.2 Models

We employ three supervised models: the mention-pair model, the mention-ranking model, and the cluster-ranking model.

Mention-Pair Model

The mention-pair (MP) model is a classifier that determines whether two NPs are co-referring or not. Each instance $\boldsymbol{i}(\text{NP}_j, \text{NP}_k)$ corresponds to NP_j and NP_k, and is represented by one of three sets of features described in the previous subsection. We follow Soon *et al.*'s [2001] method for creating training instances. Specifically, we create (1) a positive instance for each anaphoric noun phrase NP_k and its closest antecedent NP_j; and (2) a negative instance for NP_k paired with each of the intervening NPs, $\text{NP}_{j+1}, \text{NP}_{j+2}, \ldots, \text{NP}_{k-1}$. The class value of a training instance is either positive or negative, depending on whether the two NPs are coreferent in the associated text. To train the MP model, we use the SVM learner from SVM^{light} [Joachims, 2002].

After training, the classifier identifies an antecedent for an NP in a test text. Following Soon *et al.* [2001], an NP NP_k selects as its antecedent the closest preceding NP that is classified as coreferent with it. If no preceding NP is classified as coreferent with NP_k, no antecedent will be selected for NP_k.

Mention-Ranking Model

The MP model has a major weakness: since each candidate antecedent for an NP to be resolved (henceforth an *active NP*) is considered independently of the others, it only determines how good a candidate is relative to the active NP, but not how good a candidate antecedent is relative to other candidates. In other words, it fails to answer the question of which candidate is most *probable*. To address this weakness, Denis and Baldridge [2008] propose to train a *mention-ranking* (MR) model to rank all candidates for an active NP *simultaneously*.

To train a ranking model, we use the ranker-learning algorithm from SVM^{light}. Like the MP model, each training instance $\boldsymbol{i}(NP_j, NP_k)$ represents NP_k and a preceding NP NP_j. In fact, the features that represent the instance as well as the method for creating training instances are identical to those of the MP model. The only difference lies in the assignment of labels to training instances. Assuming that S_k is the set of training instances created for anaphoric NP NP_k, the *rank* value for an instance $\boldsymbol{i}(NP_j, NP_k)$ in S_k is the rank of NP_j among competing candidate antecedents, which is HIGH if NP_j is the closest antecedent of NP_k, and LOW otherwise.

After training, the MR model can be applied to rank the candidate antecedents for an active NP in a test text. There is a caveat, however: the ranker cannot determine whether an active NP NP_k is anaphoric or not, since all it does is to rank the candidate antecedents. To address this problem, we learn anaphoricity determination and coreference resolution jointly using the MR model. Specifically, when training the ranker, we provide each active NP NP_k with the option to be non-anaphoric by creating an additional instance that has rank HIGH if it is non-anaphoric and rank LOW otherwise. Its features are computed as follows: if the *Conventional* feature set is used to train the model, the features are the non-relational features computed based on NP_k; if the *Lexical* feature set is used, we simply create one feature NULL-X, where X is the head of NP_k, to help learn that X is likely non-anaphoric; and if the *Combined* feature set is used, we employ both the *Conventional* features and the *Lexical* features, as always.

Next, we apply the MR model to a test text. For each active NP NP_k, we create test instances for it by pairing it with each of its preceding NPs. To allow for the possibility that NP_k is non-anaphoric, we create an additional test instance whose features are created in the same way as in training. All these test instances are then presented to the ranker. If the additional test instance is assigned the HIGH rank by the ranker, then NP_k is classified as non-anaphoric. Otherwise, NP_k is linked to the preceding NP that is assigned the HIGH rank.

Cluster-Ranking Model

Both MP and MR models are limited in expressiveness: the information extracted from two NPs alone may not be sufficient for making an informed coreference decision. To address this problem, the *entity-mention* (EM) model is proposed [Luo *et al.*, 2004; Yang *et al.*, 2008], which determines whether an active NP belongs to a preceding coreference cluster. Hence, an EM model can employ *cluster-level* features (i.e., features that are defined over any subset of the NPs in a preceding cluster), which makes them more expressive than MP and MR models. Recently, we have combined the EM model's expressiveness with the MR model's ability to rank to create the cluster-ranking (CR) model, which *ranks* preceding *clusters* for an active NP [Rahman and Ng, 2011].

We train a CR model to jointly learn anaphoricity determination and coreference resolution in the same way as we train a joint MR model, except that the CR model is trained to rank preceding clusters rather than candidate antecedents. Hence, each training instance $\boldsymbol{i}(c_j, NP_k)$ represents a preceding cluster c_j and an anaphoric NP NP_k.

If the *Conventional* feature set is used to train the model, the instance will consist of features that are computed based solely on NP_k as well as cluster-level features, which describe the relationship between c_j and NP_k. Motivated in part by Culotta *et al.* [2007], we create cluster-level features from the relational features using four predicates: NONE, MOST-FALSE, MOST-TRUE, and ALL. Specifically, for each relational feature X in the *Conventional* feature set, we first convert X into an equivalent set of binary-valued features if it is multi-valued. Then, for each resulting binary-valued feature X_b, we create four binary-valued cluster-level features: (1) NONE-X_b is true when X_b is false between NP_k and each NP in c_j; (2) MOST-FALSE-X_b is true when X_b is true between NP_k and less than half (but at least one) of the NPs in c_j; (3) MOST-TRUE-X_b is true when X_b is true between NP_k and at least half (but not all) of the NPs in c_j; and (4) ALL-X_b is true when X_b is true between NP_k and each NP in c_j.

On the other hand, feature computation is much simpler if the *Lexical* feature set is used to train the model. Specifically, the feature set will be composed of features created between NP_k and each NP in c_j, as described in Section 2.1, and the value of a feature is the number of times it appears in the instance. Encoding feature values as frequency allows us to capture cluster-level information in a shallow manner.

After training, we can apply the resulting cluster ranker to a test text in essentially the same way as we apply the MR model, except that the cluster ranker resolves an active NP to the highest-ranked preceding cluster rather than the highest-ranked candidate antecedent. Note that the clusters preceding NP_k are formed incrementally based on the predictions of the ranker for the first $k-1$ NPs; no gold-standard coreference information is used in their formation.

2.3 The Ensemble

Since each of the three models can be trained in combination with each of the three feature sets, we can create nine coreference systems, which will all be members of our ensemble.

3 Ensemble Application

Now that we have the ensemble, the next step is to describe how to apply it to resolve an NP in a test text. As mentioned before, a major challenge in applying this ensemble stems from the fact that it contains both pairwise and cluster-based models. Below we address this challenge by describing four methods that allow us to apply the ensemble.

3.1 Method 1: Applying Best Per-NP-Type Model

This method is based on the hypothesis that different models are good at resolving different types of NPs. Consequently, for each type of NPs, we identify the member of the ensemble that is best at resolving NPs of this type using a held-out development set. When resolving an NP in a test text, we first identify its NP type, and then resolve it using the model that was determined to be the best at handling this NP type.

Two questions naturally arise. First, how many NP types should be used? Rather than considering only the three major NP types (pronouns, proper nouns, and common nouns), we follow Stoyanov *et al.* [2009] and subdivide each of the three major NP types into three subtypes, which yields nine NP subtypes, but add another pronoun subtype to cover pronouns that do not belong to any of Stoyanov *et al.*'s categories. Due to space limitations, we refer to the reader to Rahman and Ng [2011] for details on these ten NP subtypes.

Second, how can we determine which model performs the best for an NP type on the development set? To compute the score of a model for NP type C, we process the NPs in a development text in a left-to-right manner. For each NP encountered, we check whether it belongs to C. If so, we use the model to decide how to resolve it. Otherwise, we use an oracle to make the correct resolution decision (so that in the end all the mistakes can be attributed to the incorrect resolution of the NPs belonging to C, thus allowing us to directly measure its impact on overall performance). After all the test documents are processed, we compute the F-measure score on only the NPs that belong to C. We repeat this process to compute the score of each model on NP type C, and the model selected for C is the one that achieves the best F-measure score.

3.2 Method 2: Antecedent-Based Voting

In this method, when we apply the ensemble to resolve an active NP NP_k in a test text, each member of the ensemble independently selects an antecedent for NP_k. The candidate antecedent that receives the largest number of votes will be selected as the antecedent of NP_k.

At first glance, it may not be immediately clear how to apply this method to those members that employ the CR model, since they select preceding clusters, not preceding antecedents. One simple fix to this problem would be to just let a CR-based member cast a vote for each NP in the preceding cluster it selects. However, this voting scheme may cause some unfairness to the MP- and MR-based members of the ensemble, since the CR-based members can cast more votes than their MP- and MR-based counterparts.

We propose a simple solution to this problem: we force a CR-based member to select an antecedent instead. Specifically, we assume that the antecedent it selects for NP_k is the last NP in the preceding cluster that it selects (i.e., the NP in this preceding cluster that is closest to NP_k).

Note that each member has the option of casting a vote for the NULL antecedent if it believes that NP_k is non-anaphoric. If the NULL antecedent receives the largest number of votes, the ensemble will posit NP_k as non-anaphoric.

3.3 Method 3: Cluster-Based Voting

In Method 2, we make it possible to combine the votes casted by MP- and MR-based members and those casted by CR-based members by forcing the latter to select antecedents rather than preceding clusters. A natural alternative would be to force the MP- and MR-based members to select preceding clusters so that their votes can be combined with those casted by the CR-based members. This is the idea behind Method 3.

Method 3 relies crucially on the observation that each model incrementally constructs coreference clusters as it processes the NPs in a test text in a left-to-right manner, regardless of whether it selects preceding antecedents or preceding clusters in the resolution process. Hence, when applying the ensemble to resolve an active NP NP_k in a test text, each member of the ensemble may have a different set of coreference clusters constructed from the first $k - 1$ NPs.

Method 3 utilizes this observation to allow each MP- and MR-based member to cast a vote for a preceding cluster. Specifically, consider again the scenario where the ensemble is applied to resolve an active NP NP_k in a test text. If a MP- or MR-based member selects NP_j as the antecedent of NP_k, we will make it cast one vote for each NP that appears in the same cluster as NP_j. Hence, this cluster-based voting scheme allows each member to cast a vote for more than one NP. As before, the candidate antecedent that receives the largest number of votes will be selected as the antecedent of NP_k.

3.4 Method 4: Weighted Cluster-Based Voting

In Method 3, all the votes casted for a particular candidate antecedent have equal weights. In practice, some members of the ensemble may be more important than the others, so the votes casted by these members should have higher weights.

To model this observation, we first learn the weights associated with each member using a held-out development set. To do so, we employ an iterative algorithm, which initializes the weights of all the members to 1. In the ith iteration, it optimizes the weight of exactly one member, namely member (i mod 9), while keeping the weights of the rest constant. Specifically, it selects the weight from the set $\{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$ that, when used in combination with the weights associated with the other members, maximizes the F-measure score on the development set. We run the algorithm for 10 * (size of ensemble) = 10 × 9 = 90 iterations, keeping track of the F-measure achieved in each iteration. We use the vector that achieves the highest F-measure score among these 90 iterations as our final weight vector.

Testing proceeds in the same way as in Method 3, except that for each candidate antecedent of an active NP NP_k, we compute a weighted sum of the votes, where the weight of a vote is equal to the weight associated with the member that casted the vote. The candidate that receives the largest number of weighted votes is chosen as the antecedent for NP_k.

4 Evaluation

4.1 Experimental Setup

We use the ACE 2005 coreference corpus as released by the LDC, which consists of the 599 training documents used in the official ACE evaluation. The corpus was created by

	MP Models			MR Models			CR Models			Ensembles			
Source	conv	lex	comb	conv	lex	comb	conv	lex	comb	M1	M2	M3	M4
bc	50.8	57.4	55.7	52.9	56.5	54.1	55.1	57.7	58.2	59.1	59.7	60.2	**61.9**
bn	53.4	62.3	62.7	55.8	63.5	63.7	62.7	63.3	62.5	63.9	64.6	65.2	**66.9**
cts	57.0	61.1	61.3	58.6	62.7	61.7	62.5	61.1	64.1	66.0	67.0	67.6	**69.7**
nw	57.7	64.9	60.8	60.2	65.4	61.3	61.5	65.3	64.6	65.1	66.2	66.5	**68.3**
un	53.7	54.8	55.4	55.6	56.3	56.0	56.2	55.7	58.1	58.9	59.2	59.5	**61.4**
wb	63.3	65.2	57.6	65.2	68.7	54.5	67.0	63.3	67.9	69.0	69.5	69.9	**71.5**
Overall	56.2	61.2	58.8	58.2	62.4	61.2	61.2	61.5	62.8	63.7†	64.4†	64.8†	**66.8**†

Table 1: B^3 F-measure scores of the baselines and the ensemble approaches. The strongest result for each data source is boldfaced. A dagger is used whenever an ensemble method is significantly better than CR-comb, the best baseline (paired t-test, $p < 0.01$).

selecting documents from six different sources: Broadcast News (bn), Broadcast Conversations (bc), Newswire (nw), Webblog (wb), Usenet (un), and Conversational Telephone Speech (cts). For each data source, we partition the documents into five folds of roughly the same size, reserving one fold for testing and training on the remaining four folds.

We employ in our evaluation NPs automatically extracted using our in-house mention extractor, which achieves an F-score of 86.7% on our test set when trained on our training texts (see Section 6.1.2 of Rahman and Ng [2011] for details). All coreference models are trained using the default learning parameters provided by SVM^{light}. For scoring, we employ one of the commonly-used coreference scoring programs, B^3 [Bagga and Baldwin, 1998].

4.2 Results and Discussion

Baseline Systems

Since the goal of our evaluation is to determine the effectiveness of ensemble approaches, we employ as our baselines non-ensemble approaches. Specifically, we employ nine baseline systems, each of which corresponds to one of the members of our 9-member ensemble. Each baseline is trained on the four training folds and tested on the test fold.

Results of the nine baseline systems on the test set, reported in terms of B^3 F-measure, are shown in the first nine columns of Table 1. Note that the columns labeled 'conv', 'lex', and 'comb' correspond to the *Conventional*, *Lexical*, and *Combined* feature sets, respectively. Each row of the table corresponds to results from one of the six data sources. Results shown in the last row are aggregates of the results from the previous six rows, obtained by applying B^3 to score the test documents from all six data sources.

As we can see from the last row of the table, the best-performing baseline is CR-comb, which achieves an F-measure of 62.8. More interestingly, this baseline does not achieve the best performance on all six domains. In fact, its closest competitor, MR-lex, achieves the best performance on two domains (nw and wb), whereas MR-comb achieves the best performance on the bn domain. In other words, none of the baselines is consistently better than the others. This suggests that it may be beneficial to apply an ensemble approach.

It is interesting to see that for MP and MR, employing the *Lexical* features yields substantially better results than employing the *Conventional* features. In other words, these models achieve better results when conventional features such as gender and number agreement are not used. A closer look at the results reveals that some of these conventional features are in fact misguiding the learner. For instance, while the learner correctly learns that two NPs should not be coreferent when they have incompatible number, it also learns that number compatibility contributes positively to coreference. We believe that this is an instance of over-generalization, since numerous number-compatible NPs are not coreferent. Unlike a *Conventional* feature, which is typically applicable to a large number of NP pairs, a *Lexical* feature is typically applicable to a small number of NP pairs. As a result, a learner that employs *Lexical* features is less likely to over-generalize than one that employs *Conventional* features. When combining the *Conventional* and *Lexical* features into the *Combined* feature set, the performance of MP and MR both drops, presumably due to the over-generalization contributed by the *Conventional* features. On the other hand, the CR model achieves similar performance using the *Conventional* features and the *Lexical* features: the increased complexity of the CR model makes it more robust to the "noise" (i.e., features that are likely to cause over-generalization) in the *Conventional* features than the other models, but at the same time its performance is limited in part by the *Lexical* features, which have data sparseness issues despite our attempts to improve generalizability. Nevertheless, when using the *Combined* features, the CR model improves, suggesting that it can exploit the useful information in both feature sets.

Ensemble Approaches

Results of the ensemble approaches on the test set are shown in the last four columns of Table 1, where M1, M2, M3, and M4 correspond to the four methods for applying ensembles introduced in Section 3. Recall that M1 and M4 require a held-out development set for parameter selection, whereas M2 and M3 do not. To ensure a fair comparison, they all have access to the same amount of coreference-annotated data. In particular, we tune the parameters of M1 and M4 on one training fold, and train their models on the remaining three training folds; on the other hand, we train the models in M2 and M3 on all four training folds.

Two points deserve mention. First, all four ensemble methods yield better performance than CR-comb, the best baseline. In particular, the best ensemble method, M4, achieves an F-measure of 66.8, significantly outperforming CR-comb by 4 absolute F-measure points. One should bear in mind that CR-comb achieves one of the best published results on the ACE 2005 data set, and hence the 4-point improvement

	CR-comb			M1			M2			M3			M4		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Overall	54.4	74.8	62.8	55.1	75.6	63.7	55.5	76.6	64.4	55.7	77.5	64.8	57.6	79.5	**66.8**

Table 2: B^3 recall, precision, and F-measure scores for the best baseline and the ensemble methods.

we observe here is an improvement over the state of the art.[1] A closer look at the table reveals that each ensemble achieves better results on every domain than CR-comb. In fact, the performance difference between CR-comb and each of the four ensemble methods is highly statistically significant (paired t-test, $p < 0.01$). Overall, these results suggest that it is beneficial to apply ensemble methods to coreference resolution.

Second, unlike CR-comb (the best baseline), which does not offer the best performance on each data source among the baselines, M4 (the best ensemble method) achieves the best performance on each data source. In fact, each ensemble method performs better than the one immediately to its left on each data source. These results seem to suggest that the ensemble methods can indeed combine the strengths of the ensemble members.

While we showed that our ensemble approaches improve the best baseline in terms of F-measure, it would be interesting to see whether such improvement was contributed by improvements in recall, precision, or both. To answer this question, we show in Table 2 the recall (R), precision (R), and F-measure (F) scores of the best baseline (CR-comb) and the four ensemble methods. As we move across the table, we can see that the rise in F-measure is always accompanied by a simultaneous rise in recall and precision.

5 Conclusions

We examined new methods for *creating* and *applying* ensembles of learning-based coreference systems. To create ensembles, we proposed using different *supervised models*, including pairwise and cluster-based models, and different *feature sets*, which involve the under-studied word-pair-based features. Due to the presence of pairwise and cluster-based models in our ensemble, we need to address the non-trivial question of how to combine the coreference decisions made by these models. We explored four methods for combining these decisions, all of which are developed specifically for coreference resolution. Experimental results on the ACE 2005 data set show that all four ensemble methods outperform the best baseline (i.e., the cluster-ranking model trained on the *Combined* feature set), which is a state-of-the-art supervised coreference model. In particular, the best ensemble method, which involves casting weighted votes on preceding clusters, surpasses this state-of-the-art model by 4.0 in B^3 F-measure.

Acknowledgments

We thank the three anonymous reviewers for their invaluable comments on an earlier draft of the paper. This work was supported in part by NSF Grant IIS-0812261.

[1] To be specific, Haghighi and Klein's [2010] coreference model, which achieves one of the best results on the ACE data sets, reports a B^3 F-measure of 62.7 on our test set.

References

[Aone and Bennett, 1995] C. Aone and S. W. Bennett. Evaluating automated and manual acquisition of anaphora resolution strategies. In *ACL*.

[Bagga and Baldwin, 1998] A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *COLING/ACL*.

[Bengtson and Roth, 2008] E. Bengtson and D. Roth. Understanding the values of features for coreference resolution. In *EMNLP*.

[Culotta *et al.*, 2007] A. Culotta, M. Wick, and A. McCallum. First-order probabilistic models for coreference resolution. In *NAACL HLT*.

[Denis and Baldridge, 2008] P. Denis and J. Baldridge. Specialized models and ranking for coreference resolution. In *EMNLP*.

[Finkel *et al.*, 2005] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL*.

[Haghighi and Klein, 2010] A. Haghighi and D. Klein. Coreference resolution in an modular, entity-centered model. In *NAACL HLT*.

[Joachims, 2002] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*.

[Kouchnir, 2004] B. Kouchnir. A machine learning approach to German pronoun resolution. In *ACL Student Research Workshop*.

[Luo *et al.*, 2004] X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *ACL*.

[McCarthy and Lehnert, 1995] J. McCarthy and W. Lehnert. Using decision trees for coreference resolution. In *IJCAI*.

[Munson *et al.*, 2005] A. Munson, C. Cardie, and R. Caruana. Optimizing to arbitrary NLP metrics using ensemble selection. In *EMNLP*.

[Ng and Cardie, 2002] V. Ng and C. Cardie. Improving machine learning approaches to coreference resolution. In *ACL*.

[Ng and Cardie, 2003] V. Ng and C. Cardie. Weakly supervised natural language learning without redundant views. In *HLT-NAACL*.

[Ng, 2005] V. Ng. Machine learning for coreference resolution: From local classification to global ranking. In *ACL*.

[Pang and Fan, 2009] W. Pang and X. Fan. Chinese coreference resolution with ensemble learning. In *PACIIA*.

[Rahman and Ng, 2011] A. Rahman and V. Ng. Narrowing the modeling gap: A cluster-ranking approach to coreference resolution. *Journal of Artificial Intelligence Research*, 40:469–521.

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

[Stoyanov *et al.*, 2009] V. Stoyanov, N. Gilbert, C. Cardie, and E. Riloff. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *ACL-IJCNLP*.

[Vemulapalli *et al.*, 2009] S. Vemulapalli, X. Luo, J. F. Pitrelli, and I. Zitouni. Classifier combination techniques applied to coreference resolution. In *NAACL HLT Student Research Workshop*.

[Yang *et al.*, 2008] X. Yang, J. Su, J. Lang, C. L. Tan, and S. Li. An entity-mention model for coreference resolution with inductive logic programming. In *ACL*.