Maximum Metric Score Training for Coreference Resolution

Shanheng Zhao and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
{zhaosh,nght}@comp.nus.edu.sg

Abstract

A large body of prior research on coreference resolution recasts the problem as a two-class classification problem. However, standard supervised machine learning algorithms that minimize classification errors on the training instances do not always lead to maximizing the F-measure of the chosen evaluation metric for coreference resolution. In this paper, we propose a novel approach comprising the use of instance weighting and beam search to maximize the evaluation metric score on the training corpus during training. Experimental results show that this approach achieves significant improvement over the state-of-the-art. We report results on standard benchmark corpora (two MUC corpora and three ACE corpora), when evaluated using the link-based MUC metric and the mention-based B-CUBED metric.

1 Introduction

Coreference resolution refers to the process of determining whether two or more noun phrases (NPs) in a text refer to the same entity. Successful coreference resolution benefits many natural language processing tasks. In the literature, most prior work on coreference resolution recasts the problem as a two-class classification problem. Machine learning-based classifiers are applied to determine whether a candidate anaphor and a potential antecedent are coreferential (Soon et al., 2001; Ng and Cardie, 2002b).

A large body of prior research on coreference resolution follows the same process: during training, they apply standard supervised machine learning algorithms to minimize the number of misclassified training instances; during testing, they maximize either the local or the global probability of the coreferential relation assignments according to the specific chosen resolution method.

However, minimizing the number of misclassified training instances during training does not guarantee maximizing the F-measure of the chosen evaluation metric for coreference resolution. First of all, coreference is a rare relation. There are far fewer positive training instances than negative ones. Simply minimizing the number of misclassified training instances is suboptimal and favors negative training instances. Secondly, evaluation metrics for coreference resolution are based on global assignments. Not all errors have the same impact on the metric score. Furthermore, the extracted training instances are not equally easy to be classified.

In this paper, we propose a novel approach comprising the use of instance weighting and beam search to address the above issues. Our proposed maximum metric score training (MMST) approach performs maximization of the chosen evaluation metric score on the training corpus during training. It iteratively assigns higher weights to the hard-to-classify training instances. The output of training is a standard classifier. Hence, during testing, MMST is faster than approaches which optimize the assignment of coreferential relations during testing. Experimental results show that MMST achieves significant improvements over the baselines. Unlike most of the previous work, we report improved results over the state-of-the-art on all five standard benchmark corpora

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1308–1316,
Beijing, August 2010

(two MUC corpora and three ACE corpora), with both the link-based MUC metric and the mention-based B-CUBED metric.

The rest of this paper is organized as follows. We first review the related work and the evaluation metrics for coreference resolution in Section 2 and 3, respectively. Section 4 describes the proposed MMST algorithm. Experimental results and related discussions are given in Section 5. Finally, we conclude in Section 6.

2 Related Work

Soon *et al.* (2001) proposed a training and testing framework for coreference resolution. During training, a positive training instance is formed by a pair of markables, i.e., the anaphor (a noun phrase) and its closest antecedent (another noun phrase). Each markable (noun phrase) between the two, together with the anaphor, form a negative training instance. A classifier is trained on all training instances, using a standard supervised learning algorithm. During testing, all preceding markables of a candidate anaphor are considered as potential antecedents, and are tested in a back-to-front manner. The process stops if either an antecedent is found or the beginning of the text is reached. This framework has been widely used in the community of coreference resolution.

Recent work boosted the performance of coreference resolution by exploiting fine-tuned feature sets under the above framework, or adopting alternative resolution methods during testing (Ng and Cardie, 2002b; Yang et al., 2003; Denis and Baldridge, 2007; Versley et al., 2008).

Ng (2005) proposed a ranking model to maximize F-measure during testing. In the approach, n different coreference outputs for each test text are generated, by varying four components in a coreference resolution system, i.e., the learning algorithm, the instance creation method, the feature set, and the clustering algorithm. An SVM-based ranker then picks the output that is likely to have the highest F-measure. However, this approach is time-consuming during testing, as F-measure maximization is performed during testing. This limits its usage on a very large corpus.

In the community of machine learning, researchers have proposed approaches for learning a model to optimize a chosen evaluation metric other than classification accuracy on all training instances. Joachims (2005) suggested the use of support vector machines to optimize nonlinear evaluation metrics. However, the approach does not differentiate between the errors in the same category in the contingency table. Furthermore, it does not take into account inter-instance relation (e.g., transitivity), which the evaluation metric for coreference resolution cares about.

Daume III (2006) proposed a structured learning framework for coreference resolution to approximately optimize the ACE metric. Our proposed approach differs in two aspects. First, we directly optimize the evaluation metric itself, and not by approximation. Second, unlike the incremental local loss in Daume III (2006), we evaluate the metric score globally.

In contrast to Ng (2005), Ng and Cardie (2002a) proposed a rule-induction system with rule pruning. However, their approach is specific to rule induction, and is not applicable to other supervised learning classifiers. Ng (2004) varied different components of coreference resolution, choosing the combination of components that results in a classifier with the highest F-measure on a held-out development set during training. In contrast, our proposed approach employs instance weighting and beam search to maximize the F-measure of the evaluation metric during training. Our approach is general and applicable to any supervised learning classifiers.

Recently, Wick and McCallum (2009) proposed a partition-wise model for coreference resolution to maximize a chosen evaluation metric using the Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970). However, they found that training on classification accuracy, in most cases, outperformed training on the coreference evaluation metrics. Furthermore, similar to Ng (2005), their approach requires the generation of multiple coreference assignments during testing.

Vemulapalli *et al.* (2009) proposed a document-level boosting technique for coreference resolution by re-weighting the documents that have the lowest F-measures. By combining multiple classifiers generated in multiple iterations, they

achieved a CEAF score slightly better than the baseline. Different from them, our approach works at the instance level, and we output a single classifier.

3 Coreference Evaluation Metrics

In this section, we review two commonly used evaluation metrics for coreference resolution.

First, we introduce the terminology. The gold standard annotation and the output by a coreference resolution system are called key and response, respectively. In both the key and the response, a coreference chain is formed by a set of coreferential mentions. A *mention* (or markable) is a noun phrase which satisfies the markable definition in an individual corpus. A *link* refers to a pair of coreferential mentions. If a mention has no links to other mentions, it is called a *singleton*.

3.1 The MUC Evaluation Metric

Vilain *et al.* (1995) introduced the link-based MUC evaluation metric for the MUC-6 and MUC-7 coreference tasks. Let S_i be an equivalence class generated by the key (i.e., S_i is a coreference chain), and $p(S_i)$ be a partition of S_i relative to the response. Recall is the number of correctly identified links over the number of links in the key.

$$Recall = \frac{\sum(|S_i| - |p(S_i)|)}{\sum(|S_i| - 1)}$$

Precision, on the other hand, is defined in the opposite way by switching the role of key and response. F-measure is a trade-off between recall and precision.

$$F = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}$$

3.2 The B-CUBED Evaluation Metric

Bagga and Baldwin (1998) introduced the mention-based B-CUBED metric. The B-CUBED metric measures the accuracy of coreference resolution based on individual mentions. Hence, it also gives credit to the identification of singletons, which the MUC metric does not. Recall is computed as

$$Recall = \frac{1}{N} \sum_{d \in D} \sum_{m \in d} \frac{|O_m|}{|S_m|}$$

where D, d, and m are the set of documents, a document, and a mention, respectively. S_m is the equivalence class generated by the key that contains m, while O_m is the overlap of S_m and the equivalence class generated by the response that contains m. N is the total number of mentions in D. The precision, again, is computed by switching the role of key and response. F-measure is computed in the same way as the MUC metric.

4 Maximum Metric Score Training

Before explaining the algorithm, we describe our coreference clustering method used during testing. It is the same as most prior work in the literature, including Soon *et al.* (2001) and Ng and Cardie (2002b). The individual classification decisions made by the coreference classifier do not guarantee that transitivity of coreferential NPs is obeyed. So it can happen that the pair A and B, and the pair B and C are both classified as coreferential, but the pair A and C is not classified as coreferential by the classifier. After all coreferential markable pairs are found (no matter by closest-first, best-first, or resolving-all strategies as in different prior work), all coreferential pairs are clustered together to form the coreference output. By doing so, transitivity is kept: a markable is in a coreference chain if and only if it is classified to be coreferential to at least one other markable in the chain.

4.1 Instance Weighting

Suppose there are m_k and m_r coreferential links in the key and the response, respectively, and a coreference resolution system successfully predicts n correct links. The recall and the precision are then $\frac{n}{m_k}$ and $\frac{n}{m_r}$, respectively. The learnt classifier predicts false positive and false negative instances during testing. For a false positive instance, if we could successfully predict it as negative, the recall is unchanged, but the precision will be $\frac{n}{m_r - 1}$, which is higher than the original precision $\frac{n}{m_r}$. For a false negative instance, it is more subtle. If the two markables in the instance are determined to be in the same coreference chain by the clustering algorithm, it does not matter whether we predict this instance as positive or negative, i.e., this false negative does not

change the F-measure of the evaluation metric at all. If the two markables are not in the same coreference chain under the clustering, in case that we can predict it as positive, the recall will be $\frac{n+1}{m_k}$, which is higher than the original recall $\frac{n}{m_k}$, and the precision will be $\frac{n+1}{m_r+1}$, which is higher than the original precision $\frac{n}{m_r}$, as $n < m_r$. In both cases, the F-measure improves. If we can instruct the learning algorithm to pay more attention to these false positive and false negative instances and to predict them correctly by assigning them more weight, we should be able to improve the F-measure.

In the literature, besides the training instance extraction methods proposed by Soon *et al.* (2001) and Ng and Cardie (2002b) as discussed in Section 2, McCarthy and Lehnert (1995) used all possible pairs of training instances. We also use all pairs of training instances in our approach to keep as much information as possible. Initially all the pairs are equally weighted. We then iteratively assign more weights to the hard-to-classify pairs. The iterative process is conducted by a beam search algorithm.

4.2 Beam Search

Our proposed MMST algorithm searches for a set of weights to assign to training instances such that the classifier trained on the weighted training instances gives the maximum coreference metric score when evaluated on the training instances. Beam search is used to limit the search. Each search state corresponds to a set of weighted training instances, a classifier trained on the weighted training instances minimizing misclassifications, and the F-measure of the classifier when evaluated on the weighted training instances using the chosen coreference evaluation metric. The root of the search tree is the initial search state where all the training instances have identical weights of one. Each search state s can expand into two different children search states s_l and s_r. s_l (s_r) corresponds to assigning higher weights to the false positive (negative) training instances in s. The search space thus forms a binary search tree.

Figure 1 shows an example of a binary search tree. Initially, the tree has only one node: the root (node 1 in the figure). In each iteration, the algorithm expands all the leaf nodes in the beam. For example, in the first iteration, node 1 is expanded to generate node 2 and 3, which corresponds to adding weights to false positive and false negative training instances, respectively. An expanded node always has two children in the binary search tree. All the nodes are then sorted in descending order of F-measure. Only the top M nodes are kept, and the remaining nodes are discarded. The discarded nodes can either be leaf nodes or non-leaf nodes. For example, if node 5 is discarded because of low F-measure, it will not be expanded to generate children in the binary search tree. The iterative algorithm stops when all the nodes in the beam are non-leaf nodes, i.e., all the nodes in the beam have been expanded.

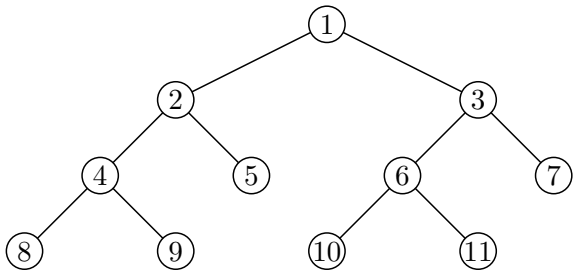

Figure 1: An example of a binary search tree

Figure 2 gives the formal description of the proposed maximum metric score training algorithm. In the algorithm, assume that we have N texts T_1, T_2, ..., T_N in the training data set. m_{ki} and m_{kj} are the ith and jth markable in the text T_k, respectively. Hence, for all $i < j$, $(m_{ki}, m_{kj}, w_{kij})$ is a training instance for the markable pair (m_{ki}, m_{kj}), in which w_{kij} is the weight of the instance. Let L_{kij} and L'_{kij} be the true and predicted label of the pair (m_{ki}, m_{kj}), respectively. Let W, C, F, and E be the set of weights $\{w_{kij}|1 \leq k \leq N, i < j\}$, the classifier, the F-measure, and a boolean indicator of whether the search state has been expanded, respectively. Finally, M is the beam size, and δ controls how much we update the weights in each iteration.

Since we train the model on all possible pairs, during testing we also test if a potential anaphor is coreferential to each preceding antecedent.

INPUT: $T_1, T_2, \ldots, T_N$
OUTPUT: classifier C
$w_{kij} \leftarrow 1$, for all $1 \leq k \leq N$ and $i < j$
$C \leftarrow$ train($\{(m_{ki}, m_{kj}, w_{kij}) | 1 \leq k \leq N, i < j\}$)
$F \leftarrow$ resolve and evaluate $T_1, \ldots, T_N$ with C
$E \leftarrow false$
BEAM$\leftarrow \{(W, C, F, E)\}$
repeat
  BEAM$' \leftarrow \{\}$
  for all (W, C, F, E) in BEAM **do**
    BEAM$' \leftarrow$ BEAM$' \bigcup \{(W, C, F, true)\}$
    if E=false **then**
      predict all L'_{kij} with C $(1 \leq k \leq N, i < j)$
      cluster into coreference chains based on L'_{kij}
      $W' \leftarrow W$
      for all $1 \leq k \leq N, i < j$ **do**
        if $L_{kij} = false$ and $L'_{kij} = true$ **then**
          $w'_{kij} \leftarrow w'_{kij} + \delta$
        end if
      end for
      $C' \leftarrow$ train($\{(m_{ki}, m_{kj}, w'_{kij}) | 1 \leq k \leq N, i < j\}$)
      $F' \leftarrow$ resolve and evaluate $T_1, \ldots, T_N$ with C'
      BEAM$' \leftarrow$ BEAM$' \bigcup \{(W', C', F', false)\}$
      $W'' \leftarrow W$
      for all $1 \leq k \leq N, i < j$ **do**
        if $L_{kij} = true$ and $L'_{kij} = false$ and
          $Chain(m_{ki}) \neq Chain(m_{kj})$ **then**
          $w''_{kij} \leftarrow w''_{kij} + \delta$
        end if
      end for
      $C'' \leftarrow$ train($\{(m_{ki}, m_{kj}, w''_{kij}) | 1 \leq k \leq N, i < j\}$)
      $F'' \leftarrow$ resolve and evaluate $T_1, \ldots, T_N$ with C''
      BEAM$' \leftarrow$ BEAM$' \bigcup \{(W'', C'', F'', false)\}$
    end if
  end for
  BEAM$\leftarrow$ BEAM$'$
  sort BEAM in descending order of F, keep top M elements
until for all E of all elements in BEAM, $E = true$
return C, from the top element (W, C, F, E) of BEAM

Figure 2: The maximum metric score training (MMST) algorithm

5 Experiments

5.1 Experimental Setup

In the experiments, we used all the five commonly used evaluation corpora for coreference resolution, namely the two MUC corpora (MUC6 and MUC7) and the three ACE corpora (BNEWS, NPAPER, and NWIRE). The MUC6 and the MUC7 corpora were defined in the DARPA Message Understanding Conference (MUC-6, 1995; MUC-7, 1998). The dry-run texts were used as the training data sets. In both corpora, each training data set contains 30 texts. The test data sets for MUC6 and MUC7 consist of the 30 and 20 formal evaluation texts, respectively. The ACE corpora were defined in NIST Automatic Content Extraction phase 2 (ACE-2) (NIST, 2002). The three data sets are from different news sources: broadcast news (BNEWS), newspaper (NPAPER), and newswire (NWIRE). Each of the three data sets contains two portions: training and development test. They were used as our training set and test set, respectively. The BNEWS, NPAPER, and NWIRE data sets contain 216, 76, and 130 training texts, and 51, 17, and 29 test texts, respectively.

Unlike some previous work on coreference resolution that assumes that the gold standard markables are known, we work directly on raw text input. Versley *et al.* (2008) presented the BART package[1], an open source coreference resolution toolkit, that accepts raw text input and reported state-of-the-art MUC F-measures on the three ACE corpora. BART uses an extended feature set and tree kernel support vector machines (SVM) under the Soon *et al.* (2001) training and testing framework. We used the BART package in our experiments, and implemented the proposed MMST algorithm on top of it. In our experiments reported in this paper, the features we used are *identical* to the features output by the preprocessing code of BART reported in Versley *et al.* (2008), except that we did not use their tree-valued and string-valued features (see the next subsection for details).

Since we use automatically extracted markables, it is possible that some extracted markables and the gold standard markables are unmatched, or *twinless* as defined in Stoyanov *et al.* (2009). How to use the B-CUBED metric for evaluating twinless markables has been explored recently. In this paper, we adopt the B^3all variation proposed by Stoyanov *et al.* (2009), which retains all twinless markables. We also experimented with their B^30 variation, which gave similar results. Note that no matter which variant of the B-CUBED metric is used, it is a fair comparison as long as the baseline and our proposed MMST algorithm are compared against each other using the same B-CUBED variant.

5.2 The Baseline Systems

We include state-of-the-art coreference resolution systems in the literature for comparison. Since we use the BART package in our experiments,

[1] `http://www.sfs.uni-tuebingen.de/~versley/BART/`

we include the results of the original BART system (with its extended feature set and SVM-light-TK (Moschitti, 2006), as reported in Versley *et al.* (2008)) as the first system for comparison. Versley *et al.* (2008) reported only the results on the three ACE data sets with the MUC evaluation metric. Since we used all the five data sets in our experiments, for fair comparison, we also include the MUC results reported in Ng (2004). To the best of our knowledge, Ng (2004) was the only prior work which reported MUC metric scores on all the five data sets. The MUC metric scores of Versley *et al.* (2008) and Ng (2004) are listed in the row "Versley *et al.* 08" and "Ng 04", respectively, in Table 1. For the B-CUBED metric, we include Ng (2005) for comparison, although it is unclear how Ng (2005) interpreted the B-CUBED metric. The scores are listed in the row "Ng 05" in Table 2.

Tree kernel SVM learning is time-consuming. To reduce the training time needed, instead of using SVM-light-TK, we used a much faster learning algorithm, J48, which is the WEKA implementation of the C4.5 decision tree learning algorithm. (Quinlan, 1993; Witten and Frank, 2005). As tree-valued features and string-valued features cannot be used with J48, in our experiments we excluded them from the extended feature set that BART used to produce state-of-the-art MUC F-measures on the three ACE corpora. All our results in this paper were obtained using this reduced feature set and J48 decision tree learning. However, given sufficient computational resources, our proposed approach is able to apply to any supervised machine learning algorithms.

Our baselines that follow the Soon *et al.* (2001) framework, using the reduced feature set and J48 decision tree learning, are shown in the row "*SNL*-Style Baseline" in Table 1 and 2. The results suggest that our baseline system is comparable to the state of the art. Although in Table 1, the performance of the *SNL*-style baseline is slightly lower than Versley *et al.* (2008) on the three ACE corpora, the computational time needed has been greatly reduced.

Our MMST algorithm trains and tests on all pairs of markables. To show the effectiveness of weight updating of MMST, we built another baseline which trains and tests on all pairs. The performance of this system is shown in the row "*All*-Style Baseline" in Table 1 and 2.

5.3 Results Using Maximum Metric Score Training

Next, we show the results of using the proposed maximum metric score training algorithm. From the description of the algorithm, it can be seen that there are two parameters in the algorithm. One parameter is M, the size of the beam. The other parameter is δ, which controls how much we increase the weight of a training instance in each iteration.

Since the best M and δ for the MUC evaluation metric were not known, we used held-out development sets to tune the parameters. Specifically, we trained classifiers with different combinations of M and δ on a development training set, and evaluated their performances on a development test set. In our experiments, the development training set contained 2/3 of the texts in the training set of each individual corpus, while the development test set contained the remaining 1/3 of the texts. After having picked the best M and δ values, we trained a classifier on the entire training set with the chosen parameters. The learnt classifier was then applied to the test set.

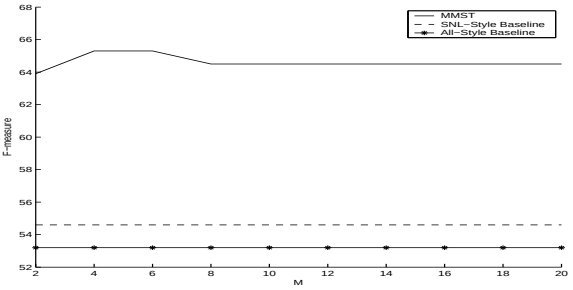

Figure 3: Tuning M on the held-out development set

To limit the search space, we tuned the two parameters sequentially. First, we fixed $\delta = 1$, which is equivalent to duplicating each training instance once in J48, and evaluated $M = 2, 4, 6, \ldots, 20$. After having chosen the best M that corresponded to the maximum F-measure, we fixed the value of M, and evaluated $\delta = 0.1, 0.2, 0.3, \ldots, 2.0$. Take MUC6 as an exam-

Model	MUC6			MUC7			BNEWS			NPAPER			NWIRE		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Versley *et al.* 08	–			–			60.7	65.4	63.0	64.1	67.7	65.8	60.4	65.2	62.7
Ng 04	75.8	61.4	**67.9**	64.2	60.2	62.1	63.1	67.8	65.4	73.5	63.3	**68.0**	53.1	60.6	56.6
SNL-Style Baseline	67.0	49.2	56.7	63.0	54.2	58.3	57.4	64.3	60.7	61.6	67.3	64.3	58.6	66.1	62.1
All-Style Baseline	56.9	69.2	62.5	51.5	73.4	60.6	53.0	76.7	62.7	56.3	75.4	64.4	53.0	74.5	61.9
MMST	73.3	59.9	65.9**††	66.8	59.8	**63.1****††	70.5	61.9	**65.9****†	69.9	64.0	66.8†	64.7	64.7	**64.7****†
	$M = 6,\ \delta = 1.0$			$M = 6,\ \delta = 0.7$			$M = 6,\ \delta = 1.8$			$M = 6,\ \delta = 0.9$			$M = 14,\ \delta = 0.7$		

Table 1: Results for the two MUC and three ACE corpora with MUC evaluation metric

Model	MUC6			MUC7			BNEWS			NPAPER			NWIRE		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Ng 05	–			–			57.0	77.1	65.6	62.8	71.2	66.7	59.3	75.4	66.4
SNL-Style Baseline	57.8	74.4	65.1	57.6	76.5	65.7	62.0	74.7	67.8	61.8	70.4	65.8	65.8	75.9	70.5
All-Style Baseline	51.6	86.3	64.6	49.1	90.1	63.6	61.6	83.7	**71.0**	63.9	74.0	68.6	64.8	80.1	**71.7**
MMST	62.7	81.5	**70.9****††	61.8	73.6	**67.2**††	61.6	83.7	**71.0****	63.1	76.2	**69.1****	64.3	81.0	**71.7**
	$M = 6,\ \delta = 1.0$			$M = 8,\ \delta = 0.8$			$M = 6,\ \delta = 0.9$			$M = 14,\ \delta = 0.5$			$M = 6,\ \delta = 0.1$		

Table 2: Results for the two MUC and three ACE corpora with B^3 evaluation metric

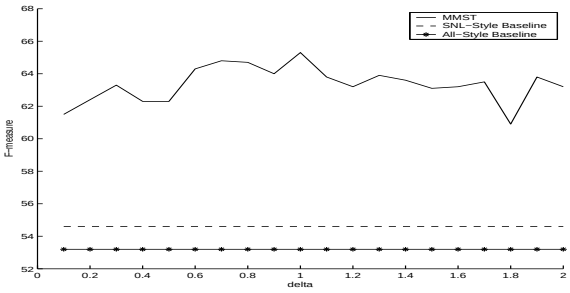

Figure 4: Tuning δ on the held-out development set

ple. The results of tuning M on MUC6 are shown in Figure 3. The maximum F-measure is obtained when $M = 4$ and $M = 6$. On all the different M values we have tried, MMST outperforms both the *SNL*-style baseline and the *All*-style baseline on the development test set. We then fixed $M = 6$, and evaluated different δ values. The results are shown in Figure 4. The best F-measure was obtained when $\delta = 1.0$. Again, on all the different δ values we have tried, MMST outperforms both baselines on the development test set.

The rows "MMST" in Table 1 and 2 show the performance of MMST on the test sets, with the tuned parameters indicated. In our experiments, the statistical significance test was conducted as in Chinchor (1995). * and ** stand for $p < 0.05$ and $p < 0.01$ over the *SNL*-style baseline, respectively. † and †† stand for $p < 0.05$ and $p < 0.01$ over the *All*-style baseline, respectively.

For the MUC metric, when compared to the *All*-style baseline, MMST gains 3.4, 2.5, 3.2, 2.4, and 2.8 improvement in F-measure on MUC6, MUC7, BNEWS, NPAPER, and NWIRE, respectively. The experimental results clearly show that MMST gains not only consistent, but also statistically significant improvement over both the *SNL*-style baseline and the *All*-style baseline in all combinations (five data sets and two baselines) on the MUC metric, except that it is not significant ($p = 0.06$) over the *SNL*-style baseline in NPAPER. As for the B-CUBED metric, MMST gains significant improvement in F-measure on MUC6 and MUC7 data sets, while its performance on the three ACE data sets are comparable to the *All*-style baseline.

5.4 Discussion

To see how MMST actually updates the weight, we use the MUC metric as an example. Under the experimental settings, it takes 6 – 9 iterations for MMST to stop on the five data sets. The number of explored states in the binary search tree, including the root, is 33, 39, 25, 29, and 75 on MUC6, MUC7, BNEWS, NPAPER, and NWIRE, respectively. It is instructive to find out the final weight of each instance. Take MUC6 as an example, the number of positive instances with weight 1, 2, 3, and 4 are 5,204, 1,568, 1,379, and 1,844, respectively, while the number of negative instances with weight 1 and 2 are 503,141 and 1,755, respec-

tively. Counting the weighted number of instances (e.g., an instance with weight 2 is equivalent to 2 instances), we have 19,853 positive and 506,651 negative training instances. This changes the ratio of the positive instances from 1.9% to 3.8%. As a by-product, MMST reduces data skewness, while using all possible NP pairs for training to keep as much information as possible.

The change of weights of the training instances is equivalent to the change of distribution of the training instances. This effectively changes the classification hypothesis to the one that tends to yield higher evaluation metric score. Take the following sentence in the MUC6 data set as an example:

> In a news release, ***the company*** said the new name more accurately reflects ***its*** focus on high-technology communications, including business and entertainment software, interactive media and wireless data and voice transmission.

In the above example, the pronoun *its* is coreferential to the antecedent NP *the company*. The baseline classifier gives a probability of 0.02 that the two NPs are coreferential. The pair is classified wrongly and none of the other pairs in the article can link the two NPs together through clustering. However, with MMST, this probability increases to 0.54, which leads to the correct classification. This is because the baseline classifier is not good at predicting in the case when the second markable is a pronoun. In the above example, *its* can have another candidate antecedent *the new name*. There are far more negative training instances than positive ones for this case. In fact, in the induced decision tree by the baseline, the leaf node corresponding to the pair *the company – its* has 7,782 training instances, out of which only 175 are positive. With MMST, however, these numbers decrease to 83 and 45, respectively. MMST also promotes the Anaphor_Is_Pronoun feature to a higher level in the decision tree. Although we use decision tree to illustrate the working of the algorithm, MMST is not limited to tree learning, and can make use of any learning algorithms that are able to take advantage of instance weighting.

It can also be seen that with the B-CUBED metric, MMST gains improvement on MUC6 and MUC7, but not on the three ACE corpora. However, the results of MMST on the three ACE corpora with the B-CUBED evaluation metric are at least comparable with the *All*-style baseline. This is because we always pick the classifier which corresponds to the maximum evaluation metric score on the training set and the classifier corresponding to the *All*-style baseline is one of the candidates. In addition, our MMST approach improves upon state-of-the-art results (Ng, 2004; Ng, 2005; Versley et al., 2008) on most of the five standard benchmark corpora (two MUC corpora and three ACE corpora), with both the link-based MUC metric and the mention-based B-CUBED metric.

Finally, our approach performs all the F-measure maximization during training, and is very fast during testing, since the output of the MMST algorithm is a standard classifier. For example, on the MUC6 data set with the MUC evaluation metric, it took 1.6 hours and 31 seconds for training and testing, respectively, on an Intel Xeon 2.33GHz machine.

6 Conclusion

In this paper, we present a novel maximum metric score training approach comprising the use of instance weighting and beam search to maximize the chosen coreference metric score on the training corpus during training. Experimental results show that the approach achieves significant improvement over the baseline systems. The proposed approach improves upon state-of-the-art results on most of the five standard benchmark corpora (two MUC corpora and three ACE corpora), with both the link-based MUC metric and the mention-based B-CUBED metric.

Acknowledgments

We thank Yannick Versley for providing us the BART package and the preprocessed data. This research was done for CSIDM Project No. CSIDM-200804 partially funded by a grant from the National Research Foundation (NRF) administered by the Media Development Authority (MDA) of Singapore.

References

Bagga, Amit and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the LREC1998*, pages 563–566.

Chinchor, Nancy. 1995. Statistical significance of MUC-6 results. In *Proceedings of the MUC-6*, pages 39–43.

Daume III, Hal. 2006. *Practical Structured Learning for Natural Language Processing*. Ph.D. thesis, University of Southern California.

Denis, Pascal and Jason Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of the NAACL-HLT2007*, pages 236–243.

Hastings, W. K. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Joachims, Thorsten. 2005. A support vector method for multivariate performance measures. In *Proceedings of the ICML2005*, pages 377–384.

McCarthy, Joseph F. and Wendy G. Lehnert. 1995. Using decision trees for coreference resolution. In *Proceedings of the IJCAI1995*, pages 1050–1055.

Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092.

Moschitti, Alessandro. 2006. Making tree kernels practical for natural language learning. In *Proceedings of the EACL2006*, pages 113–120.

MUC-6. 1995. Coreference task definition (v2.3, 8 Sep 95). In *Proceedings of the MUC-6*, pages 335–344.

MUC-7. 1998. Coreference task definition (v3.0, 13 Jul 97). In *Proceedings of the MUC-7*.

Ng, Vincent and Claire Cardie. 2002a. Combining sample selection and error-driven pruning for machine learning of coreference rules. In *Proceedings of the EMNLP2002*, pages 55–62.

Ng, Vincent and Claire Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *Proceedings of the ACL2002*, pages 104–111.

Ng, Vincent. 2004. *Improving Machine Learning Approaches to Noun Phrase Coreference Resolution*. Ph.D. thesis, Cornell University.

Ng, Vincent. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *Proceedings of the ACL2005*, pages 157–164.

NIST. 2002. The ACE 2002 evaluation plan. `ftp://jaguar.ncsl.nist.gov/ace/doc/ACE-EvalPlan-2002-v06.pdf`.

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

Soon, Wee Meng, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Stoyanov, Veselin, Nathan Gilbert, Claire Cardie, and Ellen Riloff. 2009. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of the ACL-IJCNLP2009*, pages 656–664.

Vemulapalli, Smita, Xiaoqiang Luo, John F. Pitrelli, and Imed Zitouni. 2009. Classifier combination techniques applied to coreference resolution. In *Proceedings of the NAACL-HLT2009 Student Research Workshop and Doctoral Consortium*, pages 1–6.

Versley, Yannick, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang, and Alessandro Moschitti. 2008. BART: A modular toolkit for coreference resolution. In *Proceedings of the ACL2008:HLT Demo Session*, pages 9–12.

Vilain, Marc, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the MUC-6*, pages 45–52.

Wick, Michael and Andrew McCallum. 2009. Advances in learning and inference for partition-wise models of coreference resolution. Technical Report UM-CS-2009-028, University of Massachusets, Amherst, USA.

Witten, Ian H. and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, second edition.

Yang, Xiaofeng, Guodong Zhou, Jian Su, and Chew Lim Tan. 2003. Coreference resolution using competition learning approach. In *Proceedings of the ACL2003*, pages 176–183.