Chinese Noun Phrase Coreference Resolution: Insights into the State of the Art

Chen CHEN *Vincent NG*
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688, USA
{yzcchen,vince}@hlt.utdallas.edu

ABSTRACT
Compared to the amount of research on English coreference resolution, relatively little work has been done on Chinese coreference resolution. Worse still, it has been difficult to determine the state of the art in Chinese coreference resolution, owing in part to the lack of a standard evaluation dataset. The organizers of the CoNLL-2012 shared task, Modeling Unrestricted Multilingual Coreference in OntoNotes, have recently addressed this issue by providing standard training and test sets for developing and evaluating Chinese coreference resolvers. We aim to gain insights into the state of the art via extensive experimentation with our Chinese resolver, which is ranked first in the shared task on the Chinese test data.

TITLE AND ABSTRACT IN CHINESE

中文名词短语共指消解：探究研究现状

与大量的英文共指消解研究相比，针对中文的共指研究相对较少。更糟糕的是，中文共指消解很难确定当前的研究现状，部分原因是缺少标准的评测集。CoNLL-2012 共同任务（在 OntoNotes 上对无限制多语言共指消解的建模）的组织者解决了这个问题，他们提供了标准化的训练集和测试集供开发和评定中文共指消解系统。我们的系统在中文测试集的评测中排名第一。本文在原系统的基础上做了广泛的实验，以探究中文共指消解的研究现状。

KEYWORDS: coreference resolution, anaphora resolution, Chinese language processing, OntoNotes, CoNLL shared task.

KEYWORDS IN CHINESE: 共指消解, 指代消解, 中文语言处理, OntoNotes, CoNLL 共同任务.

Proceedings of COLING 2012: Posters, pages 185–194,
COLING 2012, Mumbai, December 2012.

1 Introduction

Coreference resolution is the task of determining which noun phrases (NPs) in a text refer to the same real-world entity. Compared to the amount of research on English coreference resolution, relatively little work has been done on Chinese coreference resolution. Worse still, it has been difficult to determine the state of the art in Chinese coreference resolution. The reason can be attributed in part to the lack of a standard evaluation dataset: while recently developed Chinese resolvers are typically evaluated on the ACE datasets, different researchers have used different splits of the ACE data for training and testing, making performance comparisons difficult.[1] The organizers of the CoNLL-2012 shared task, Modeling Unrestricted Multilingual Coreference in OntoNotes, have recently addressed this issue by providing free access to the training and test sets used in the official evaluation (Pradhan et al., 2012).

Our goal in this paper is to gain a better understanding of the state of the art in Chinese coreference resolution by providing an extensive empirical analysis of our Chinese resolver, which is ranked first on the Chinese subtask of the CoNLL-2012 shared task. Briefly, our resolver adopts a hybrid rule-based/machine learning approach to coreference resolution, extending the successful rule-based multi-pass sieve approach (Raghunathan et al., 2010; Lee et al., 2011) with lexical features that have proven useful in machine learning approaches (Rahman and Ng, 2011a, 2011b). Our analysis is focused on four issues.

1. **Mention detection.** Previous work has shown that the quality of the extracted *mentions* (i.e., the NPs participating in a coreference chain) plays an important role in the performance of a resolver. To what extent is the performance of our resolver limited by the *recall* and *precision* of our mention detector? To improve the precision of our mention detector, we need to improve its *mention pruning* strategy, but to what extent is its precision limited by our current mention pruning strategy? To improve the recall of our mention detector, we need to improve the extraction of mentions from syntactic parse trees, but to what extent is its recall limited by the *mention extraction* strategy versus the quality of the syntactic parses?
2. **Preprocessing.** After mention detection, we need to compute *features* based on the extracted mentions using preprocessing tools such as syntactic parsers and named entity (NE) recognizers. To what extent is the performance of our resolver limited by the correctness of the output produced by these tools?
3. **The coreference algorithm.** To better understand our hybrid approach, we focus on three questions. First, do we really need a hybrid approach? In other words, will our approach work equally well without the learning component? Second, how much does each sieve in the multi-pass sieve approach contribute to overall performance? Third, how important is the ordering of the sieves as far as performance is concerned?
4. **Comparison with classifier-based approaches.** In the shared task, our resolver outperformed those systems that adopted the popularly-used mention-pair (MP) model (Soon et al., 2001), a classifier trained to determine whether two given NPs are coreferent. However, we cannot claim that our coreference algorithm is superior to the MP model because we do not know which component(s) of our resolver (e.g., mention detection, feature computation, resolution) contributed to the superiority. In fact, much of the previous work focuses on comparing *systems* rather than *models*/*methods*. We determine whether our resolution *method* is better than the MP model if both are given the same set of mentions and features.

[1] One may wonder why researchers did not simply follow the same train-test split used in the official ACE evaluations. The reason is that only the training sets used in the official evaluations are released to the public.

	Docs	Mentions	Chains	Mentions/Chain
Train	1391	102854	28257	3.6
Development	172	3875	14183	3.7
Test	166	3559	12801	3.6

Table 1: Statistics of the training, development and test sets.

2 Datasets and Evaluation Measures

The training, development and test sets that we use in our experiments are the same as those in the official CoNLL-2012 shared task evaluation (see Table 1 for their statistics). As we will see in the next section, we use the training set for learning probabilities (e.g., how likely is it that two mentions are coreferent?), the development set for tuning thresholds, and the test set for evaluating our resolver.

We follow the method adopted by the shared task for evaluating a resolver. Specifically, the score of a resolver is the unweighted average of the F-measure scores computed by three scoring programs (MUC, B^3, and entity-based CEAF), whose implementation is provided by the shared task organizers. It is worth mentioning that (1) a resolver is not rewarded for correctly identifying singleton mentions; and (2) a mention is considered correctly extracted if and only if there is an exact phrase match between the gold mention and the extracted mention. In addition, elided pronouns, copulars, and appositive constructions are excluded in this (and the official shared task) evaluation.

3 Our Coreference Resolver

In this section, we give an overview of our two-step resolver as used for the shared task (see Chen and Ng (2012) for details). Note that linguistic annotations such as word segmentation and syntactic parses come with the shared task datasets and do not need to be computed separately.

Step 1: Mention Detection. To build a mention detector, we employ a two-step approach. First, in the *extraction* step, we extract mentions from all the NP and QP nodes in syntactic parse trees. Then, in the *pruning* step, we identify and filter erroneously extracted mentions by employing two types of pruning. In *heuristic pruning*, we use simple heuristics to prune erroneous mentions. For instance, we prune a candidate mention m_k if it is an interrogative pronoun or an NE that is a PERCENT or QUANTITY. In *learning-based pruning*, we prune m_k if the probability that it is a mention in the training data is less than t_C, where t_C is a threshold whose value is determined using the development set.

Step 2: Sieve-Based Coreference Resolution. A *sieve* is composed of one or more manually-designed rules for establishing the coreference relation between two mentions. A sieve-based resolver is composed of a set of *sieves* ordered by their precision, with the most precise sieves appearing first. When given a text, the resolver makes multiple passes over it: in the i-th pass, it uses only the rules in the i-th sieve to establish coreference relations.

Our resolver is composed of 10 sieves. The **Chinese Head Match** (CHM) sieve identifies the coreference relation between two same-head mentions where one is embedded within the other in newswire articles. The **Discourse Processing** (DP) sieve resolves mentions, especially first- and second-person pronouns, in dialogues. The **Exact String Match** (ESM) sieve resolves a non-pronominal mention to a mention with the same string. The **Precise Constructs** (PC) sieve posits two mentions as coreferent based on lexical and syntactic information, such as whether one is an acronym of the other and whether the mentions are in an appositive construction. In addition, we incorporated Chinese-specific rules for determining whether one mention is an abbreviation of

the other based on NE information. **Strict Head Match A−C** (SHMA−C) are three head match sieves that contain progressively less precise coreference rules based on head matching. The **Proper Head Match** (PHM) is a relaxed version of Strict Head Match C applicable only to proper nouns. The **Pronouns** (Pro) sieve contains rules for resolving pronouns based on features that we learned from the training set, such as gender and number. Finally, the **Lexical Pair** (LP) sieve identifies coreference relations based on lexical features. For example, one rule specifies that two mentions are coreferent if the probability that their heads are coreferent according to the training data is greater than t_{HPU}, where t_{HPU} is a threshold determined using the development set.

Note that the usual linguistic constraints on non-coreference are applied before a rule in any of the sieves posits two mentions as coreferent. These constraints are implemented as a single *non-coreference* rule, which specifies that two mentions m_i and m_j cannot be coreferent if one of the following five conditions holds: (1) they satisfy the i-within-i constraint (Haghighi and Klein, 2009); (2) they refer to different speakers in a dialogue despite being the same string; (3) they are in a copular construction; (4) m_i is composed of two NPs connected by an "and" and m_j is one of the conjuncts; and (5) the probability that m_i and m_j are coreferent (as calculated from training data) falls below a certain threshold.

Step 3: Postprocessing. We postprocess the coreference partition before sending it to the scoring program. Specifically, we remove from it (1) all coreference links between two mentions in appositive constructions and (2) singleton clusters.

4 Evaluation

In this section, we conduct extensive experimentation with our resolver in an attempt to shed light on the four issues raised in the introduction.

4.1 Mention Detection and Preprocessing

We begin by describing the experimental setup related to the first two issues.

The first issue concerns how the performance of our resolver is affected by the quality of the mentions. Stoyanov et al. (2009) show that for English coreference, results obtained using gold mentions (i.e., mentions taken directly from gold-standard coreference annotations) are substantially better than those produced using system mentions (i.e., automatically extracted mentions). While we will explore gold mentions in our Chinese experiments, we seek to gain a better understanding of this issue by considering three types of system mentions. The first type, *system mentions from system parses with imperfect pruning*, is typically what researchers use to produce end-to-end coreference results. It is composed of mentions extracted from system parse trees (i.e., automatically generated parse trees) and pruned using the method described in Step 1 of Section 3. The second type, *system mentions from system parses with perfect pruning*, is the same as the first type except that an oracle is used to prune all the erroneous mentions. The third type, *system mentions from gold parses with imperfect pruning*, is the same as the first type except that the mentions are extracted from gold-standard parse trees. Experiments involving the last two types of mentions will enable us to determine the role of pruning and syntactic parsing in coreference resolution.

The second issue concerns the impact of preprocessing. In particular, we address two questions. First, to what extent will the performance of our resolver be affected if the linguistic features used to create the rules in the sieves are computed based on system rather than gold parse trees? Second, to what extent will its performance be affected if the features are computed based on system rather

	Feature Computation		MUC			B^3			$CEAF_e$			Avg
Mention Type	Parse Type	NE Type	R	P	F	R	P	F	R	P	F	F
System Mentions from System Parses with Imperfect Pruning	System Parses	System NEs	62.5	67.1	64.7	71.2	78.4	74.6	53.6	49.1	51.3	63.5
		Gold NEs	62.2	67.0	64.5	71.0	78.6	74.6	53.5	48.9	51.1	63.4
		No NEs	59.9	64.7	62.2	69.7	77.8	73.6	53.4	48.7	51.0	62.2
	Gold Parses	System NEs	62.4	67.2	64.8	71.0	78.4	74.6	53.8	49.0	51.3	63.5
		Gold NEs	62.2	67.2	64.6	70.8	78.7	74.5	53.7	48.8	51.1	63.4
		No NEs	60.0	65.0	62.4	69.6	77.9	73.5	53.5	48.7	51.0	62.3
System Mentions from System Parses with Perfect Pruning	System Parses	System NEs	65.4	92.5	76.6	65.8	92.4	76.8	79.1	44.9	57.3	70.3
		Gold NEs	65.4	92.6	76.7	65.6	92.7	76.8	79.2	44.9	57.3	70.3
		No NEs	64.3	91.6	75.6	64.3	92.2	75.8	78.8	44.4	56.8	69.4
	Gold Parses	System NEs	65.5	92.6	76.7	65.9	92.4	76.9	79.2	45.0	57.4	70.3
		Gold NEs	65.5	92.7	76.8	65.7	92.7	76.9	79.3	45.0	57.4	70.4
		No NEs	64.5	91.7	75.7	64.4	92.2	75.9	78.9	44.5	56.9	69.5
System Mentions from Gold Parses with Imperfect Pruning	System Parses	System NEs	73.5	74.3	73.9	76.3	80.5	78.3	58.2	57.3	57.8	70.0
		Gold NEs	73.3	74.4	73.9	76.1	80.9	78.4	58.3	57.1	57.7	70.0
		No NEs	70.8	72.1	71.4	74.4	79.9	77.0	58.0	56.4	57.2	68.5
	Gold Parses	System NEs	74.8	74.9	74.9	77.1	80.8	78.9	58.6	58.5	58.6	70.8
		Gold NEs	74.6	75.0	74.8	76.9	81.2	79.0	58.6	58.1	58.4	70.7
		No NEs	72.1	72.8	72.5	75.3	80.2	77.7	58.4	57.6	58.0	69.4
Gold Mentions	System Parses	System NEs	78.1	93.2	85.0	75.0	91.6	82.5	84.0	59.2	69.4	79.0
		Gold NEs	78.1	93.4	85.0	74.8	92.0	82.5	84.2	59.1	69.4	79.0
		No NEs	76.6	92.4	83.8	73.0	91.4	81.2	83.6	57.9	68.4	77.8
	Gold Parses	System NEs	79.1	93.6	85.7	75.8	91.9	83.1	84.8	60.4	70.6	79.8
		Gold NEs	79.2	93.7	85.8	75.7	92.3	83.2	84.9	60.5	70.6	79.9
		No NEs	77.9	92.9	84.7	74.0	91.7	81.9	84.3	59.5	69.7	78.8

Table 2: Impact of the quality of mentions, parse trees and NEs on coreference performance.

than gold NE information, and what if no NE information is used by the resolver?[2] We hypothesize that coreference performance will drop when gold parses and NEs are replaced with their system counterparts, since these two types of linguistic annotations are used extensively by the rules in our resolver: syntactic parses are used to identify copular and appositive constructions, find speakers in dialogue, and determine the modifier(s) of a mention, whereas NEs are used in the Chinese-specific abbreviation rules in the Precise Constructs sieve, the pronoun resolution rules (for computing the animacy of an NP), and some of the relaxed head matching rules in the Proper Head Match sieve.

Considering both issues together, we have 24 coreference experiments resulting from different combinations of four types of mentions (gold mentions and the three types of system mentions), two types of syntactic parse trees (gold and system), and three types of NE annotations (gold, system, and none). Table 2 shows the test results of these 24 experiments, which are organized as follows. There are four blocks of results corresponding to the four types of mentions (column 1); for each type of mentions, we conduct experiments using two types of parses (column 2) and three types of NEs (column 3). Hence, each row of the table corresponds to one of these 24 experiments. Results are reported in terms of the recall (R), precision (P), and F-score provided by three scoring programs (MUC, B^3, $CEAF_e$) as well as the unweighted average of their F-scores (Avg).

Comparing the first two blocks of results, which differ in terms of whether imperfect or perfect pruning is applied to system mentions extracted from system parses, we see that coreference results with perfect pruning surpass those with imperfect pruning by an Avg F-score of 6.8–7.2%. Not

[2]Two points concerning NE annotations deserve mention. First, when "no NE" is used, all the coreference rules that depend on NE information are removed from the sieves. We consider the "no NE" option because the use of NEs is not permitted in the official closed track evaluation in the shared task. Second, system NEs are not provided by the shared task organizers. To obtain system NEs for the development and test sets, we employ a CRF-based NE recognizer trained on the gold NEs in the training set using 18 lexical, semantic, and gazetteer-based features.

Mention Type	No Pruning			Imperfect Pruning			Perfect Pruning		
	R	P	F	R	P	F	R	P	F
System mentions from system parses	86.1	33.0	47.7	83.5	43.7	57.4	86.1	100	92.5
System mentions from gold parses	98.7	36.6	53.4	96.3	48.9	64.8	98.7	100	99.3
Gold mentions	100	100	100	---	---	---	---	---	---

Table 3: Mention detection results.

surprisingly, improvements stem primarily from increases in precision according to MUC and B^3, since these pruned mentions will not be (erroneously) resolved.[3]

A related question is: how well does our pruning method perform at the mention detection level? To answer this question, we show in Table 3 the results of mention detection when different mention extraction methods are combined with different mention pruning methods. From row 1, we can see that our (imperfect) pruning method leaves a lot of room for improvement: currently it only yields a precision of 43.7%. However, row 1 also shows that our pruning method is somewhat useful: pruning increases precision by more than 10% with only a 3% drop in recall.

Conclusion 1: Improving the mention pruning algorithm can substantially improve coreference performance.

Comparing the first and third blocks of results, we see that coreference results obtained using mentions from gold parse trees surpass those obtained using mentions from system parse trees by an Avg F-score of 6.3−7.3%. Additional insights can be gained by considering the mention detection results in Table 3. From row 2, we can see that without pruning, we manage to recall 98.7% of the mentions from gold parse trees using our simple mention extraction heuristic. We believe that the high recall can be attributed to the fact that the manual coreference annotations in OntoNotes were performed on top of gold parse trees.

Conclusion 2: Improving the recall of mention detection can substantially improve coreference performance.

Note, however, that conclusion 2 does not necessarily hold true for other languages: Pradhan et al. (2012) found in their *gold mention boundaries* experiments that merely increasing the recall of mention detection does not lead to improvements in coreference performance for English and Arabic.[4] We hypothesize that this can be attributed to the failure of the resolution algorithm to find the correct antecedent in these languages despite the increase in the number of correct mentions. Additional experiments are needed to precisely determine the reason, however.

Comparing the third and fourth blocks of results, we see that coreference results obtained using gold mentions surpass those obtained using system mentions from gold parse trees by an Avg F-score of 9.0−9.4%. This improvement is accompanied by a simultaneous rise in recall and precision for MUC and B^3. This should not be surprising: precision increases because erroneous mentions are pruned and will not be resolved, and recall increases because mentions are more likely to be correctly resolved due to the reduction in the number of candidate antecedents.

Comparing the first and fourth blocks of results, the improvement is even more dramatic: coreference results obtained using gold mentions surpass those obtained using system mentions from system parse trees by an Avg F-score of 15.5−16.5%. An interesting question is: how much of this difference can be attributed to the *recall* versus the *precision* of our mention detector? We can answer this question by comparing the third and fourth blocks of results in Table 2 again. Row 2 of Table 3 says that 96.3% of the gold mentions are used when producing the third block of results

[3]Note that results obtained from $CEAF_e$ do not show the same trend as those from MUC and B^3 owing to the somewhat counterintuitive definitions of $CEAF_e$ recall and precision, but space limitations preclude further discussion.

[4]We repeated the experiments in Table 2 on the mentions with gold boundaries and obtained results that are identical to those of the system mentions obtained from system parses.

in Table 2. Hence, the difference between the third and fourth blocks of results in Table 2 can be attributed mostly to the *precision* of our mention detector. Returning to our question with this assumption, we can attribute approximately 58% of the difference (i.e., 9.0−9.4 of 15.5−16.5) to the precision of the mention detector and the remaining 42% to its recall.[5]

Conclusion 3: Improving both the recall and precision of mention detection can yield substantially better coreference performance than improving one of them.

Next, we examine how coreference performance varies when different types of parse trees and NEs are used for feature computations.[6] Regardless of which of the four types of mentions are used, we can see from Table 2 that (1) replacing gold parses with system parses and/or replacing gold NEs with system NEs for feature computations has little impact on coreference performance; (2) Avg F-score drops by 1.0−1.5% when NE information is not used. A closer examination of the results reveals that (1) NE information is particularly useful in establishing a mention and its abbreviated form; and (2) the insignificant difference between the results using gold NEs and those using system NEs can be attributed to the fact that our NE recognizer achieves reasonably good F-scores (80%) for PERSON and GPE, the NE classes that our resolver relies on.[7]

Conclusion 4: Improving syntactic parsing and NE recognition for the sake of feature computation is unlikely to improve coreference performance.

4.2 The Coreference Algorithm

Our next set of experiments aims to address three questions concerning our rule-based and learning-based coreference algorithm. All experiments in this subsection are performed with system mentions extracted from system parse trees, with features computed over system parses and NEs.

First, how important is the ordering of the sieves? Raghunathan et al. (2010) implicitly suggest that ordering matters by noting that the sieves should be arranged in decreasing order of precision, although they never show how important this particular ordering is to coreference performance. To answer this question, we randomly order the sieves in our resolver and measure the performance of the resulting resolver on the test set. Results averaged over five random orderings are shown in row 2 of Table 4. For convenience, the results of our unperturbed resolver are shown in row 1. As we can see, Avg F-score decreases significantly by 1.2% when the sieves are ordered randomly.

Conclusion 5: The ordering of the sieves in our resolver is important.

Second, how important is the learning component of our resolver? To answer this question, we remove all components of our resolver that are learning-based, including (1) the String Pair sieve; (2) the last condition of the non-coreference rule; and (3) learning-based mention pruning. Test results of the resulting resolver are shown in row 3 of Table 4. In comparison to row 1, Avg F-score drops significantly by 0.6%.

Conclusion 6: The learning component plays a significant role in our hybrid approach.

Finally, how much performance gain is provided by each sieve? To answer this question, we start with only the first sieve and then add the sieves incrementally to our resolver. The Avg F-score obtained after adding each sieve is shown in Table 5. As we can see, Chinese Head Match and Exact String Match contribute the most to performance, followed by Discourse Processing.

[5] Note that this estimation is very rough: it does not take into account the fact that it tends to be harder to get from, say, 80% F-score to 90% F-score than it is to get from 70% F-score to 80% F-score.

[6] Note that we distinguish between using parse trees for feature computations versus using them for mention extraction. Hence, we can compute features from system parses while extracting mentions from gold parses.

[7] All statistical significance test results in this paper are obtained using the paired t-test, with $p < 0.05$.

System Variation	MUC			B^3			$CEAF_e$			Avg
	R	P	F	R	P	F	R	P	F	F
Our resolver	62.5	67.1	64.7	71.2	78.4	74.6	53.6	49.1	51.3	63.5
With randomly ordered sieves	60.8	65.5	63.1	69.9	77.2	73.3	52.8	48.2	50.4	62.3
Without learning	61.6	66.7	64.1	70.4	78.2	74.1	53.2	48.3	50.6	62.9

Table 4: Results on perturbing the components of our resolver.

Sieve	CHM	DP	ESM	PC	SHMA	SHMB	SHMC	PHM	Pro	SP
Avg F	32.6	39.0	56.8	58.2	58.9	59.7	59.7	59.8	63.3	63.5

Table 5: Avg F-scores of our resolver as sieves are incrementally inserted.

	MUC			B^3			$CEAF_e$			Avg
	R	P	F	R	P	F	R	P	F	F
Our resolver	62.5	67.1	64.7	71.2	78.4	74.6	53.6	49.1	51.3	63.5
MP (atomic features)	56.7	54.2	55.4	71.2	70.0	70.6	41.9	44.0	42.9	56.3
MP (atomic features + non-coreference)	56.2	55.3	55.7	70.6	71.1	70.8	42.7	43.5	43.1	56.5
MP (rules as features)	55.1	62.8	58.7	66.4	78.4	71.9	52.3	45.2	48.5	59.7
MP (rules as features + non-coreference)	54.8	63.9	59.0	66.0	79.4	72.1	53.2	44.8	48.6	59.9

Table 6: Comparison of our resolver with the mention-pair model.

4.3 Comparison with the Mention-Pair Model

Our final set of experiments aims to compare our resolver with the MP model. To implement the MP model, we use SVM^{light} (Joachims, 1999) for classifier training with the instances created using Soon et al.'s (2001) method. We then apply the resulting classifier in combination with Soon et al.'s closest-first clustering algorithm to create a coreference partition for each test document.

For a fairer comparison, both models are given the same set of mentions (i.e., system mentions extracted from system parse trees). To ensure that they are given the same set of features, we experiment with two methods of creating features for the MP model from the coreference rules used by our resolver. In the first method, we create one binary feature from each rule used in the sieves, setting its value to 1 if and only if the corresponding rule was used to establish a coreference link between the two mentions in our resolver. In the second method, we create one binary feature from each distinct *rule condition*[8]; employing these *atomic* features will enable us to determine whether the difference in performance between our resolver and the MP model can be attributed to the way the SVM combines features. Recall that our resolver also employs lexical features and the non-coreference rule. To ensure fairness, we incorporate lexical features into the MP model's feature set by creating one binary feature from each head, head pair and string pair found in the training data. In addition, we employ the non-coreference rule as a hard constraint for the closest-first clustering algorithm: the clustering algorithm cannot posit two mentions as coreferent if they satisfy the non-coreference rule, even if the classifier posits them as coreferent.

Given two feature generation methods and a choice of whether the non-coreference constraint is applied in the clustering process, we have four experiments with the MP model. Their results are shown in rows 2−5 of Table 6, and the results of our resolver are shown in row 1 for convenience. As we can see, our resolver is significantly better than the MP models that use rules as features, which in turn are significantly better than those that use atomic features. However, the use of the non-coreference constraint has an insignificant impact on the performance of the MP model.

Conclusion 7: The SVM used to train the MP model is unable to combine features as well as a human.

[8]Recall that each rule is of the form $A_1 \wedge A_2 \wedge \cdots A_n$, where each A_i is a condition that needs to be satisfied in order for the rule to posit two mentions as coreferent. If A_i appears in multiple rules, only one binary feature will be created.

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 Grants IIS-1147644 and IIS-1219142.

References

Chen, C. and Ng, V. (2012). Combining the best of two worlds: A hybrid approach to multilingual coreference resolution. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning: Shared Task*, pages 56−63.

Haghighi, A. and Klein, D. (2009). Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1152−1161.

Joachims, T. (1999). Making large-scale SVM learning practical. In Scholkopf, B. and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*, pages 44−56. MIT Press.

Lee, H., Peirsman, Y., Chang, A., Chambers, N., Surdeanu, M., and Jurafsky, D. (2011). Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28−34.

Pradhan, S., Moschitti, A., Xue, N., Uryupina, O., and Zhang, Y. (2012). CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning: Shared Task*, pages 1−40.

Raghunathan, K., Lee, H., Rangarajan, S., Chambers, N., Surdeanu, M., Jurafsky, D., and Manning, C. (2010). A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492−501.

Rahman, A. and Ng, V. (2011a). Coreference resolution with world knowledge. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 814−824.

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

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

Stoyanov, V., Gilbert, N., Cardie, C., and Riloff, E. (2009). Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 656−664.