Chinese Event Coreference Resolution: Understanding the State of the Art

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

Abstract

Given the relatively small amount of work on event coreference resolution, our understanding of the task is arguably fairly limited. This makes it difficult to determine how to improve an event coreference resolver. We seek to gain a better understanding of the state of the art in event coreference resolution by performing the first publicly available analysis of a Chinese event coreference resolver.

1 Introduction

Event coreference resolution is the task of determining which event mentions in a text refer to the same real-world event. Compared to entity coreference (the task of determining which entity mentions in a text refer to the same real-world entity), there is much less work on event coreference.

Given the lesser amount of work on event coreference, our understanding of the task is arguably fairly limited. Specifically, while it is not surprising that the performance of an event coreference resolver depends heavily on the quality of the output of its preprocessing components, it is not clear *to what extent* the noise inherent in the output of each preprocessing component is limiting the performance of a resolver. Note that this issue is more serious for event coreference than for entity coreference: since an event coreference resolver lies towards the end of the information extraction pipeline, it has to rely on the noisy output produced by more upstream components than its entity counterpart. The lack of understanding of this issue makes it difficult to identify the components that need most attention. Moreover, when analyzing the errors made by a resolver, it is not clear which types of errors can be fixed by improving the resolution algorithm and which ones can be fixed by improving the preprocessing components. This makes it difficult to precisely determine how to improve the resolution algorithm. Taken together, these two issues make it difficult to understand how an event coreference resolver should be improved.

Our goal in this paper is to better understand the state of the art in event coreference and provide directions for further work on this task by addressing the aforementioned issues, which are summarized by the following two questions. First, to what extent is the noise inherent in the output of *each* of its upstream components limiting its performance? Second, what are the major types of errors that are attributable to (and therefore can be fixed by improving) the resolution algorithm?

We address these two questions by presenting a systematic analysis of a state-of-the-art Chinese event coreference resolver. Our decision to focus on Chinese can be attributed in part to the lack of publicly available results on Chinese event coreference resolution. In particular, to our knowledge, almost all recent work on event coreference has reported results for English (e.g., Humphreys et al. (1997), Chen et al. (2009), Bejan and Harabagiu (2010), Chen et al. (2011), Lee et al. (2012)).[1] Hence, the results in the paper can serve as a baseline against which future work on Chinese event coreference can be compared.

It is worth mentioning that similar questions were faced by entity coreference researchers, who have reached a point where it is crucial to answer these questions in order to make further progress. For that reason, a number of recent research papers have focused on these questions via analyzing the inner workings of entity coreference resolvers (e.g., Stoyanov et al. (2009), Recasens and Hovy (2010), Chen and Ng (2012a)). On the other hand, no attempts have been made to address these questions in the context of event coreference.

[1] A notable exception is Zinmeister et al.'s (2012) work, which reports results for both German and English.

International Joint Conference on Natural Language Processing, pages 822–828,
Nagoya, Japan, 14-18 October 2013.

2 ACE Event Coreference

As mentioned in the introduction, event coreference is the task of determining which event mentions in a text refer to the same real-world event. The ACE 2005 event coreference task, which is the version of the event coreference task we focus on, requires that an event coreference resolver performs coreference only on event mentions belonging to one of the ACE event types.

More specifically, an ACE event mention has a *type* and a *subtype*. In ACE 2005, eight types are defined, which are further subcategorized into 33 subtypes. Not surprisingly, two event mentions that have different subtypes cannot be coreferent.

To better understand the ACE 2005 coreference task, consider the sentence in Figure 1, which is taken from the ACE 2005 corpus. This example contains three event mentions: 砍 (stabbed), 伤 (injured) and 行凶 (criminal). 砍 and 行凶 have type CONFLICT and subtype ATTACK, whereas 伤 has type LIFE and subtype INJURE. In this example, 砍 and 行凶 are coreferent because they refer to the same real-world event.

(张家荣)(昨天傍晚) 在 (路上) 骑自行车运动时遭到了 (两名歹徒) 持 (刀)[砍][伤]。(歹徒) 的 [行凶] 动机可能和 (张家荣) 的问证有关联。
(Zhang Jiarong) was cycling on (the road) (yesterday evening) and was [injured] when (two men) [stabbed] (him) with (a knife). (The thugs)' [criminal] motivation may have something to do with (Zhang Jiarong)'s testimony in a criminal case.

Figure 1: An example. Event mentions are bracketed and entity mentions are parenthesized.

3 Six Upstream Components

Our event coreference resolver adopts a fairly standard ACE event coreference system architecture, relying on six components. As we will see, the first four components have a direct influence on event coreference, meaning that their output will be used to create features for use by the event coreference model. On the other hand, the last two components only have an indirect influence on event coreference through other components.

Component 1: Event mention boundary identification and subtyping. This component (1) provides the event mentions for event coreference resolution, and (2) labels each event mention with its event subtype. Since two event mentions with different subtypes cannot be coreferent, subtypes can be used to create useful features for event coreference. To implement this component, we use our Chinese event extraction system (Chen and Ng, 2012c), which jointly learns these tasks.

Component 2: Event mention attribute value computation. This component takes as input a set of event mentions (provided by Component 1) and computes for each mention its attributes, including its POLARITY, MODALITY, GENERICITY and TENSE. Since two event mentions that differ with respect to any of these attributes cannot be coreferent, they can be used to create useful features for event coreference. Following Chen et al. (2009), we train a classifier to compute the value of each attribute of each event mention (see Chen et al. for details on the implementation of these classifiers).

Component 3: Event argument and role classification. This component takes as input a set of event mentions (provided by Component 1) and a set of candidate arguments (provided by Component 5). For each event mention em, it (1) identifies those candidate arguments that are the true arguments of em (e.g., the participants, time, and place of em), and then (2) assigns a *role* (e.g., VICTIM, PLACE, TIME-WITHIN) to each of its true arguments. Since two events involving different times, places, or participants cannot be coreferent, the arguments and their roles can be used to create useful features for event coreference. To implement this component, we use our Chinese event extraction system (Chen and Ng, 2012c), which jointly learns these two tasks.

Component 4: Entity coreference resolution. This component takes as input a set of entity mentions (provided by Component 5) and creates a coreference partition in which each cluster contains all and only those entity mentions that refer to the same real-world entity. Since two event mentions having coreferent arguments are likely to be coreferent, the output of this component can be used to create useful features for event coreference. To create a coreference partition from a set of entity mentions, we employ our Chinese entity coreference resolver (Chen and Ng, 2012b).

Component 5: Entity mention boundary identification. This component provides the candidate arguments and the entity mentions needed by the aforementioned components, so it only has an indirect influence on event coreference. Since candidate arguments can be entity mentions, time ex-

pressions, and value expressions[2], we train one CRF (using CRF++[3]) to extract each of these three types of candidate arguments.

Component 6: Entity typing and subtyping. This component takes a set of entity mentions (provided by Component 5) and determines the semantic type and subtype of each entity mention. Knowing the semantic type and subtype of an argument is helpful for classifying the role of event arguments. For example, we can assign the role VICTIM only to those arguments with entity type PERSON. To determine semantic type and subtype, we train two SVM multiclass classifiers using SVM^{multiclass} (Tsochantaridis et al., 2004).

4 Chinese Event Coreference Resolver

Underlying our learning-based Chinese event coreference resolver is a mention-pair model (Soon et al., 2001) trained using the SVM^{light} package (Joachims, 1999). Training instances are created as follows. For each anaphoric event mention em, we create one positive instance between em and its closest antecedent. To create negative instances, we pair em with each of its preceding event mentions that is not coreferent with it.

Each instance is represented using 32 features, which are modeled after a state-of-the-art English event coreference resolver (Chen and Ji, 2009; Chen et al., 2009) (see the Appendix for a detailed description of these features). After training, the resulting mention-pair model is used in combination with a closest-first single-link clustering algorithm to impose a coreference partition on the event mentions in a test text (Soon et al., 2001).

5 Empirical Analysis

Next, we address our first question: to what extent is the noise inherent in the output of *each* of the upstream components limiting a resolver's performance? To answer this question, we start with a resolver where all of its upstream components are assumed to be oracle components, and then replace each of them with its real (i.e., imperfect) version one after the other, as described below.

5.1 Experimental Setup

For evaluation, we report five-fold cross-validation results over the 633 Chinese documents in the ACE 2005 training corpus. Results, expressed in terms of recall (R), precision (P), and F-measure (F), are obtained by applying three commonly-used coreference scoring programs, MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and $CEAF_e$ (a.k.a. ϕ_4-CEAF) (Luo, 2005), to the coreference partitions produced by our resolver after singleton event mentions are removed. Stanford's Chinese NLP and Speech Processing tool[4] is used for word segmentation, syntactic parsing, and dependency parsing.

5.2 Results and Analysis

As mentioned above, we start with an event coreference resolver that assumes that all the six upstream components are error-free (see row 1 of Table 1 for its performance), and then replace each oracle component with its *system* (i.e., machine-learned) counterpart one after the other (rows 2--7 of Table 1). Therefore, the results in the last row of Table 1 correspond to the performance of an *end-to-end* event coreference resolver that relies solely on system components. Below, we discuss the impact of each component on coreference performance. Note that these components can be considered in a different order than what we show in this section. Here, we show one ordering in which the parent(s) of a component are considered *after* the component itself.

Component 2 (Event mention attribute value computation). First, we replace the oracle event mention attribute predictors with their system counterparts. Since there are four event mention attributes, namely, POLARITY, MODALITY, GENERICITY and TENSE, we trained four classifiers to predict the attribute values of an event mention. Our results suggest that each of these four classifiers is only marginally better than a simple majority baseline.[5] We then used the values predicted by these classifiers to compute features for the *test* instances for the event coreference resolver. Note that the features for the *training* instances for the resolver are computed based on gold rather than system event attribute values.

[2] See the ACE 2005 task definition (http://www.itl.nist.gov/iad/mig/tests/ace/2005/doc/ace05-evalplan.v2a.pdf).

[3] http://crfpp.googlecode.com/svn/trunk/doc/index.html

[4] http://nlp.stanford.edu/projects/chinese-nlp.shtml

[5] Chen et al. (2009) trained classifiers to predict the attribute values of English event mentions. While their POLARITY, MODALITY, and GENERICITY classifiers perform only slightly better than a majority baseline, their TENSE classifier has reasonably good performance. This is perhaps not surprising: tense classification for English verbs is easier than for Chinese verbs since Chinese verb forms do not change according to tense.

	MUC			B^3			$CEAF_e$			Avg
	R	P	F	R	P	F	R	P	F	F
1. All oracle	80.4	70.0	74.8	88.4	79.7	83.8	57.3	66.8	61.7	73.4
2. + System event mention attribute values										
2a. system event mention attributes on test only	59.9	60.8	60.3	76.8	78.5	77.6	53.3	52.5	52.9	63.6
2b. no event mention attribute features	72.8	63.4	67.8	84.2	76.6	80.2	52.1	60.0	55.8	67.9
2c. system event mention attributes on train & test	72.5	64.5	68.3	83.8	77.4	80.5	53.1	59.9	56.3	68.3
3. + System event arguments and roles	71.2	61.2	65.8	83.9	74.9	79.1	49.9	58.0	53.6	66.2
4. + System entity coreference chains	61.6	58.5	60.0	79.0	75.7	77.3	49.1	51.5	50.3	62.5
5. + System entity types & subtypes	62.2	57.9	60.0	79.4	75.2	77.2	49.0	52.3	50.6	62.6
6. + System entity mention boundaries	63.3	57.4	60.2	80.2	74.4	77.2	48.2	52.8	50.4	62.6
7. + System entity mention boundaries & subtypes	37.4	36.7	37.1	72.8	71.1	71.9	40.6	41.1	40.8	49.9

Table 1: Results when oracle components are replaced with system components one after the other.

Given the poor performance of these attribute predictors, we hypothesize that coreference performance will drop considerably when the oracle attribute predictors are replaced with their system counterparts. Results of this experiment, shown in row 2a, are consistent with this hypothesis: in comparison to row 1, the Avg F-score (unweighted average of the MUC, B^3, and $CEAF_e$ F-scores)[6] drops significantly by nearly 10%.[7] A natural question is: do these results represent an unnecessary amplification of the impact of event mention attributes on event coreference performance? To answer this question, we consider two alternative ways of employing the system event mention attribute values for event coreference resolution. One way is to simply discard all features created from these attributes when training and testing the event coreference resolver. Results of this experiment are shown in row 2b. Somewhat interestingly, we can see by comparing rows 2a and 2b that discarding these features does yield a less considerable drop in performance than employing them. Another way is to employ system attribute values to generate features for both the training and test instances for the event coreference resolver. Results of this experiment are shown in row 2c. In comparison to row 2b, we see that there is a slight, but insignificant, improvement in coreference performance. Consequently, we assume in the rest of our experiments that we employ system event mention attribute values on both the training set and the test set (i.e., the configuration in row 2c), since it seems to more accurately reflect the impact of event mention attributes on event coreference performance.

Conclusion 1: Improving the four event attribute classifiers could significantly improve event coreference.

Component 3 (Event argument and role classification). Next, we replace the oracle event argument and role classification component with its system counterpart. Results on the test set indicate that when gold event mentions and subtype, gold entity type and subtype, and gold entity mention boundaries are used, the F-scores of system argument classification and role classification are 76.9% and 68.2% respectively. Replacing the oracle component with this system counterpart, we see that the Avg coreference performance drops slightly, though still significantly, by 2.1%.

Conclusion 2: Event argument classification and role classification have a small, but significant, impact on event coreference performance.

Component 4 (Entity coreference). Next, we replace oracle entity coreference with system entity coreference. As noted before, event coreference directly depends on entity coreference. Our system entity coreference resolver achieves a MUC F-score of 78.0% when gold entity mentions are used. Comparing rows 3 and 4, we see that replacing oracle entity coreference with system entity coreference incurs nearly a 4% drop in coreference performance according to Avg F-score. These results suggest that employing a better entity coreference resolver can improve event coreference. Joint learning of event and entity coreference may help to improve both tasks.

Conclusion 3: Improving entity coreference could significantly improve event coreference.

Component 6 (Entity typing and subtyping). Next, we replace oracle entity typing and subtyping with its system counterpart. Recall that this component has only an indirect impact on event coreference but a direct impact on event argument and role classification, since the entity type and subtype of a mention are used to create features for training the event argument and role classifier. Our

[6] For ease of exposition, we follow the CoNLL 2011 and 2012 shared tasks and use Avg F when discussing results.

[7] All statistical significance test results reported in this paper are conducted using the paired t-test ($p < 0.05$).

system entity type and subtype classifiers achieve F-scores of 90.1% and 81.6%, respectively, when gold entity mentions are used. Using system rather than gold entity types and subtypes, the F-scores of the event argument classifier and the event role classifier drop by 2.8% and 4.3% respectively, but comparing rows 4 and 5, coreference performance does not drop.

Conclusion 4: Improving entity type and subtype classification is unlikely to improve event coreference.

Component 5 (Entity mention boundary detection). Next, we replace gold entity mention boundary detection with its system counterpart. The performance of our system mention boundary detection component is reasonably good: it achieves an F-score of 84.7%. Comparing rows 5 and 6, we see that replacing gold mention boundary detection with its system counterpart does not alter event coreference performance.

Conclusion 5: Improving entity mention boundary detection may not improve event coreference.

Component 1 (Event mention boundary identification and subtyping). Finally, we replace the oracle event mention boundary identification and subtyping component with its system counterpart. Our learned event mention boundary identifier achieves an F-score of 65.1%, while our event subtype classifier achieves an F-score of 61.30%. Comparing rows 6 and 7, we see that replacing oracle with system event mention boundary identification and subtyping causes Avg F-score to drop by 12.7%, even though the event extraction system we employ for event mention boundary identification and subtype classification is a state-of-the-art system. Furthermore, MUC recall decreases more abruptly than MUC precision, which suggests that the low recall of event mention boundary identification severely harms system performance.

Conclusion 6: Event mention boundary identification and subtyping is the upstream component that has the largest impact on event coreference. There is a lot of room for improving this component, especially its recall.

We conclude this section with two noteworthy points. First, the *cumulative* study we conducted in this section is just one possible way to examine the extent to which the noise inherent in the output of each of the upstream components limits a resolver's performance. Another way is to conduct an *ablation* study: we start with a resolver where all of its upstream components are assumed to be oracle components, and then replace exactly one oracle upstream component with its real (i.e., imperfect) version in each ablation experiment. Since the conclusions that can be drawn from the ablation study and the cumulative study are similar, we will omit the description of the ablation experiments and their results for the sake of brevity.

Second, although we discuss the results in this section in terms of Avg F, it turns out that in these experiments Avg F exhibits the same performance trends as those of MUC, B^3, and $CEAF_e$. In particular, the significance test results that we obtained using Avg F remain unchanged when Avg F is replaced with any of the three scoring metrics.

6 Error Analysis

Next, we address our second question: what are the major types of errors that are attributable to (and therefore can be fixed by improving) the resolution algorithm? To answer this question, we perform a qualitative error analysis on the output produced by the resolver where all six upstream components are gold. This ensures that all the errors are attributable to the resolution algorithm.

6.1 Three Major Types of Precision Errors

Lack of event timestamping. Only those events that occur at the same time can be coreferent. We use the TENSE event attribute as an feature to enforce TENSE consistency, essentially employing TENSE as a very rough approximation of the timestamp of an event. Perhaps not surprisingly, many events having the same tense are not coreferent. For example, consider the following pair of sentences, where the *triggers* (i.e., the words/phrases corresponding to the event mentions) are enclosed in brackets.

去年三月间杨光南在上海首度 [被捕]

In last March, Yang Guangnan was [arrested] in Shanghai for the first time

杨光南在上海再度 [被捕]

Yang Guangnan was [arrested] again in Shanghai

It is fairly easy to see that the first *arrest* occurred before the second one and therefore the two event mentions should not be coreferent. However, our resolver incorrectly posits them as coreferent, since they have the same PERSON and PLACE arguments (i.e., 杨光南 (Yang Guangnan) and 上海 (Shanghai)) and the same tense (i.e., *past*). If we could assign a timestamp to each event, our resolver might fix this type of precision error.

Incompatible triggers. As coreference between arguments is a strong indicator that the corresponding event mentions are coreferent, our resolver tends to link two events mentions with coreferent arguments together even when the triggers are not semantically compatible. The following pair of sentences illustrates this type of error.

萨姆·努乔马 2 8 日乘专机抵达平壤开始对朝鲜进行正式友好 [访问]

On the 28th, Sam Nujoma arrived in Pyongyang by plane for an official goodwill [visit] to the DPRK

纳米比亚总统萨姆·努乔马 2 8 日乘专机 [抵达] 平壤

Namibian President Sam Nujoma [arrived] in Pyongyang by plane on the 28th

As we can see, the triggers are 访问 (visit) and 抵达 (arrived). Since both their ARTIFACT arguments (i.e., 萨姆·努乔马 (Sam Nujoma)) and their VEHICLE arguments (i.e., 专机 (plane)) are coreferent, our resolver wrongly posits the two event mentions as coreferent, although the corresponding triggers, 访问 and 抵达, are semantically incompatible. If we had access to a semantic dictionary from which we can derive the fine-grained semantic type of these triggers, our resolver might fix this type of error.

Incompatible important arguments. Our resolver has the tendency to posit two *largely compatible* event mentions as coreferent, i.e., they have only a small number of incompatible arguments but many features that suggest that they are coreferent (e.g, same trigger word, some coreferent arguments). Consider the example below.

代表团 [访问] 了瑞典

The delegation [visited] Sweden

[访问] 丹麦期间，中国基督教代表团举行记者招待会

During their [visit] in Denmark, the Chinese Christian delegation held a press conference

Note that the two event mentions have identical triggers 访问 (visited) and ARTIFACT arguments 代表团 (The delegation). Given these positive evidences, our resolver posits the event mentions as coreferent despite the fact that their DESTINATION arguments are incompatible: one is 瑞典 (Sweden) and the other is 丹麦 (Denmark). To fix this kind of error, we may employ human knowledge to mark each argument role of each event subtype as *important* or *unimportant*, and enforce the constraint that two event mentions cannot be coreferent if their arguments in an *important* argument role are incompatible. In our example, we would mark both ARTIFACT and DESTINATION as important argument roles for the MEETING event subtype (i.e., the subtype for *visited*), meaning that we will disallow the two event mentions in the example to be coreferent unless both the ARTIFACT and DESTINATION arguments are compatible.

6.2 Two Major Types of Recall Errors

Coreferent mentions with synonymous triggers. Our resolver fails to link some event mentions that have synonymous but lexically different trigger words. Consider the following example.

犹太人针对阿拉伯人的 [暴力]

Jewish [violence] against the Arabs

[冲突] 双方

Two parties of [conflict]

While the event mentions corresponding to the two synonymous triggers, 暴力 (violence) and 冲突 (conflict), are coreferent, our resolver fails to identify them as coreferent because the triggers are lexically different. We could fix this type of error if we had access to a semantic dictionary that can suggest that *violence* and *conflict* have similar meaning.

Coreferent mentions with compatible arguments. Some arguments are not coreferent but compatible. Consider the following sentences.

南斯拉夫国家元首第一次对波黑进行这样的 [访问]

Yugoslavia's head of state [visited] Bosnia-Herzegovina for the first time

科什图尼察 [访问] 波黑首都萨拉热窝

Kostunica [visited] Sarajevo, the capital of Bosnia-Herzegovina

Here, the triggers are 访问 (visited). Our resolver does not posit these two event mentions as coreferent since their arguments are not coreferent. However, if we had the world knowledge to recognize 波黑 (Bosnia and Herzegovina) and 萨拉热窝 (Sarajevo) are compatible arguments, then our resolver might be able to discover the coreference link between the two event mentions.

7 Conclusion

We conducted the first empirical analysis of an ACE-style Chinese event coreference system, focusing on the questions of (1) the extent to which event coreference performance is affected by errors made by upstream components in the information extraction pipeline, and (2) the types of errors made by the resolution algorithm. We hope our analysis will help direct future research.

Acknowledgments

We thank the four reviewers for their insightful comments. This work was supported in part by NSF Grants IIS-1147644 and IIS-1219142.

Appendix: Event Coreference Features

The 32 features used by our event coreference resolver can be divided into five groups. Group i ($1 \leq i \leq 4$) contains the features computed based on Component i's output, and Group 5 contains the remaining features. For convenience, we use em_2 to refer to an event mention to be resolved and em_1 to refer to one of its candidate antecedents.

Group 1 (4): Whether em_1 and em_2 agree w.r.t. event type; whether they agree w.r.t. event subtype; the concatenation of their event types; the concatenation of their event subtypes.

Group 2 (8): The four event mention attributes of em_2; whether em_1 and em_2 are compatible w.r.t. each of the event mention attributes.

Group 3 (4): The roles and number of the arguments that only appear in em_1; the roles and number of the arguments that only appear in em_2.

Group 4 (6): The roles and number of arguments between em_1 and em_2 that have the same role and are also in the same entity coreference chain; the roles and number of arguments between em_1 and em_2 that have same role but are in different coreference chains; the roles and number of arguments between em_1 and em_2 that have different roles but are in the same coreference chain.

Group 5 (10): The sentence distance between em_1 and em_2; the number of event mentions intervening them; the number of words between them; whether they have the same trigger word; whether they are in a coordinating structure; whether they have same basic verb; whether they agree in number if they are nouns; whether they have incompatible modifiers if they are nouns; the concatenation of the part-of-speech tags of their heads; the concatenation of their trigger words.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proc. of the LREC Workshop on Linguistic Coreference*, page 563--566.

Cosmin Bejan and Sanda Harabagiu. 2010. Unsupervised event coreference resolution with rich linguistic features. In *Proc. of the ACL*, pages 1412--1422.

Zheng Chen and Heng Ji. 2009. Graph-based event coreference resolution. In *Proc. of TextGraphs-4*, pages 54--57.

Chen Chen and Vincent Ng. 2012a. Chinese noun phrase coreference resolution: Insights into the state of the art. In *Proc. of COLING 2012: Posters Volume*, pages 185--194.

Chen Chen and Vincent Ng. 2012b. Combining the best of two worlds: A hybrid approach to multilingual coreference resolution. In *Proc. of EMNLP-CoNLL: Shared Task*, pages 56--63.

Chen Chen and Vincent Ng. 2012c. Joint modeling for Chinese event extraction with rich linguistic features. In *Proc. of COLING*, pages 529--544.

Zheng Chen, Heng Ji, and Robert Haralick. 2009. A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In *Proc. of the RANLP Workshop on Events in Emerging Text Types*, pages 17--22.

Bin Chen, Jian Su, Sinno Jialin Pan, and Chew Lim Tan. 2011. A unified event coreference resolution by integrating multiple resolvers. In *Proc. of IJCNLP*, pages 102--110.

Kevin Humphreys, Robert Gaizauskas, and Saliha Azzam. 1997. Event coreference for information extraction. In *Proc. of the ACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, pages 75--81.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*, pages 44--56. MIT Press.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proc. of EMNLP-CoNLL*, pages 489--500.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proc. of HLT/EMNLP*, pages 25--32.

Marta Recasens and Eduard Hovy. 2010. Coreference resolution across corpora: Languages, coding schemes, and preprocessing information. In *Proc. of the ACL*, pages 1423--1432.

Wee Meng Soon, 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.

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

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. of ICML*, pages 104--112.

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

Heike Zinmeister, Stefanie Dipper, and Melanie Seiss. 2012. Abstract pronominal anaphors and label nouns in German and English: Selected case studies and quantitative investigations. *Translation: Corpora, Computation, Cognition*, 2(1):47--80.