A Machine Learning-Based Coreference Detection System For OntoNotes

Yaqin Yang
Computer Science Department
Brandeis University
Waltham, Massachusetts, USA
yaqin@brandeis.edu

Nianwen Xue
Computer Science Department
Brandeis University
Waltham, Massachusetts, USA
xuen@brandeis.edu

Peter Anick
Computer Science Department
Brandeis University
Waltham, Massachusetts, USA
peter_anick@yahoo.com

Abstract

In this paper, we describe the algorithms and experimental results of Brandeis University in the participation of the CoNLL Task 2011 closed track. We report the features used in our system, and describe a novel cluster-based chaining algorithm to improve performance of coreference identification. We evaluate the system using the OntoNotes data set and describe our results.

1 Introduction

This paper describes the algorithms designed and experiments finished in the participation of the CoNLL Task 2011. The goal of the Task is to design efficient algorithms for detecting entity candidates and identifying coreferences. Coreference identification is an important technical problem. Its importance in NLP applications has been observed in previous work, such as that of Raghunathan et al., Pradhan et al., Bergsma et al., Haghighi et al., and Ng et al.. While most of the existing work has evaluated their systems using the ACE data set, in this work we present our experimental results based on the OntoNotes data set used in the CoNLL 2011 Shared Task. We detail a number of linguistic features that are used during the experiments, and highlight their contribution in improving coreference identification performance over the OntoNotes data set. We also describe a cluster-based approach to multi-entity chaining. Finally, we report experimental results and summarize our work.

2 Data Preparation

We divide the CoNLL Task into three steps. First, we detect entities from both the training data and the development data. Second, we group related entities into entity-pairs. Finally, we use the generated entity-pairs in the machine learning-based classifier to identify coreferences. In this section, we describe how we extract the entities and group them into pairs.

2.1 Generating Entity Candidates

We use the syntactic parse tree to extract four types of entities, including noun phrase, pronoun, premodifier and verb (Pradhan et al., 2007). This method achieves 94.0% (Recall) of detection accuracy for gold standard trees in the development data. When using the automatic parses, not surprisingly, the detection accuracy becomes lower, with a performance drop of 5.3% (Recall) compared with that of using the gold standard trees. Nevertheless, this method can still cover 88.7% of all entities existing in the development data, thus we used it in our algorithm.

2.2 Generating Entity-Pairs From Individual Entities

In the annotated training documents, an entity has been marked in a coreference chain that includes all coreferential entities. In our algorithm, we only detect the closest antecedent for each entity, instead of all coreferences, of each entity. Specifically, we define each training and testing instance as a pair of entities. During the training process, for each entity encountered by the system, we create a *positive* instance by pairing an entity with its closest antecedent (Soon et al., 2001). In addition, a set of *negative* instances are also created by pairing the entity with any preceding entities that exist between its closest antecedent and the entity itself (note that the antecedent must be a coreference of the current entity, whereas preceding entities may not be coreferential). For example, in the entity sequence “A, B, C, D, E”, let us assume that “A” is the closest antecedent of “D”. Then, for entity “D”, “A-D” is considered a positive instance, whereas “B-D” and “C-D” are two negative instances.

To generate testing data, every entity-pair within the same sentence is considered to form positive or negative instances, which are then used to form testing data. Since occasionally the distance between an entity and its closest antecedent can be far apart, we handle considerably distant coreferences by consid-

Proceedings of the 15th Conference on Computational Natural Language Learning: Shared Task, pages 117–121,
Portland, Oregon, 23-24 June 2011.

ering each entity-pair that exists within the adjacent N sentences. During our experiments, we observed that the distance between an entity and its closest antecedent could be as far as 23 sentences. Therefore, in the classification process, we empirically set N as 23.

3 Machine Learning-Based Classification

After labeling entity pairs, we formalize the coreference identification problem as a binary classification problem. We derive a number of linguistic features based on each entity-pair, i and j, where i is the potential antecedent and j the anaphor in the pair (Soon et al., 2001). Generally, we select a set of features that have been proved to be useful for the coreference classification tasks in previous work, including gender, number, distance between the antecedent and the anaphor, and WordNet (WordNet, 2010). In addition, we design additional features that could be obtained from the OntoNotes data, such as the speaker or author information that is mainly available in Broadcast Conversation and Web Log data (Pradhan et al., 2007). Moreover, we extract apposition and copular structures and used them as features. The features we used in the system are detailed below.

- **Independent feature:** 1) if a noun phrase is definite; 2) if a noun phrase is demonstrative; 3) gender information of each entity; 4) number information of each entity; 5) the entity type of a noun phrase; 6) if an entity is a subject; 7) if an entity is an object; 8) if an noun phrase is a coordination, the number of entities it has; 9) if a pronoun is preceded by a preposition; 10) if a pronoun is "you" or "me"; 11) if a pronoun is "you" and it is followed by the word "know".
- **Name entity feature:** 1) i-j-same-entity-type-etype=True, if i and j have the same entity type; 2) i-j-same-etype-subphrase=True, if i and j have the same entity type and one is the subphrase of the other.
- **Syntactic feature:** 1) i-j-both-subject=True, if i and j are both subjects; 2) if i and j are in the same sentence, record the syntactic path between i and j, e.g. i-j-syn-path=PRP$\wedge$NP!PRP; 3) i-j-same-sent-diff-clause=True, if i and j are in the same sentence but in different clauses.
- **Gender and number feature:** 1) i-j-same-gender=True/False, by comparing if i and j have the same gender; 2) i-j-same-num=True/False, by comparing if i and j have the same number; 3) i-j-same-num-modifier=True/False, by comparing if i and j have the same number modifier, e.g. "two countries" and "they both" have the same number modifier; 4) i-j-same-family=True/False, we designed seven different families for pronouns, e.g. "it", "its" and "itself" are in one family while "he", "him", "his" and "himself" are in another one.
- **Distance feature:** 1) i-j-sent-dist, if the sentence distance between i and j is smaller than three, use their sentence distance as a feature; 2) i-j-sent-dist=medium/far: if the sentence distance is larger than or equal to three, set the value of i-j-sent-dist to "medium", otherwise set it to "far" combined with the part-of-speech of the head word in j.
- **String and head word match feature:** 1) i-j-same-string=True, if i and j have the same string; 2) i-j-same-string-prp=True, if i and j are the same string and they are both pronouns; 3) i-j-sub-string=True, if one is the sub string of the other, and neither is a pronoun; 4) i-j-same-head=True, if i and j have the same head word; 5) i-j-prefix-head=True, if the head word of i or j is the prefix of the head word of the other; 6) i-j-loose-head, the same as i-j-prefix-head, but comparing only the first four letters of the head word.
- **Apposition and copular feature:** for each noun phrase, if it has an apposition or is followed by a copular verb, then the apposition or the subject complement is used as an attribute of that noun phrase. We also built up a dictionary where the key is the noun phrase and the value is its apposition or the subject's complement to define features. 1) i-appo-j-same-head=True, if i's apposition and j have the same head word; 2) i-j-appo-same-head=True, if j's apposition has the same head word as i; we define the similar head match features for the noun phrase and its complement; Also, if an i or j is a key in the defined dictionary, we get the head word of the corresponding value for that key and compare it to the head word of the other entity.
- **Alias feature:** i-j-alias=True, if one entity is a proper noun, then we extract the first letter of each word in the other entity. (The extraction process skips the first word if it's a determiner and also skips the last one if it is a possessive case). If the proper noun is the same as the first-letter string, it is the alias of the other entity.
- **Wordnet feature:** for each entity, we used Wordnet to generate all synsets for its head word, and for each synset, we get all hypernyms and hyponyms. 1) if i is a hypernym of j, then i-hyper-j=True; 2) if i is a hyponym of j, then i-hypo-j=True.
- **Speaker information features:** In a conversation, a speaker usually uses "I" to refer to himself/herself, and most likely uses "you" to refer to the next speaker. Since speaker or author name information is given in Broadcast Conversation and Web Log data, we use such information to design features that represent relations between pronouns and

speakers. 1) i-PRP1-j-PRP2-same-speaker=True, if both i and j are pronouns, and they have the same speaker; 2) i-I-j-I-same-speaker=True, if both i and j are "I", and they have the same speaker; 3) i-I-j-you-same-speaker=True, if i is "I" and j is "you", and they have the same speaker; 4) if i is "I", j is "you" and the speaker of j is right after that of i, then we have feature i-I-j-you&itarget=jspeaker; 5) if i is "you", j is "I" and the speaker of j is right after that of i, then we have feature i-you-j-I-itarget=jspeaker; 6) if both i and j are "you", and they followed by the same speaker, we consider "you" as a general term, and this information is used as a negative feature.

- **Other feature:** i-j-both-prp=True, if both i and j are pronouns.

4 Chaining by Using Clusters

After the classifier detects coreferential entities, coreference detection systems usually need to chain multiple coreferential entity-pairs together, forming a coreference chain. A conventional approach is to chain all entities in multiple coreferential entity-pairs if they share the same entities. For example, if "A-B", "B-C", and "C-D" are coreferential entity-pairs, then A, B, C, and D would be chained together, forming a coreference chain "A-B-C-D".

One significant disadvantage of this approach is that it is likely to put different coreference chains together in the case of erroneous classifications. For example, suppose in the previous case, "B-C" is actually a wrong coreference detection, then the coreference chain created above will cause A and D to be mistakenly linked together. This error can propagate as coreference chains become larger.

To mitigate this issue, we design a cluster-based chaining approach. This approach is based on the observation that some linguistic rules are capable of detecting coreferential entities with high detection precision. This allows us to leverage these rules to *double-check* the coreference identifications, and reject chaining entities that are incompatible with rule-based results.

To be specific, we design two lightweight yet efficient rules to cluster entities.

- **Rule One**. For the first noun phrase (NP) encountered by the system, if 1) this NP has a name entity on its head word position or 2) it has a name entity inside and the span of this entity includes the head word position, a cluster is created for this NP. The name entity of this NP is also recorded. For each following NP with a name entity on its head word position, if there is a cluster that has the same name entity, this NP is considered as a coreference to other NPs in that cluster, and is put into that cluster. If the system cannot find such a cluster, a new cluster is created for the current NP.
- **Rule Two**. In Broadcast Conversation or Web Log data, a speaker or author would most likely use "I" to refer to himself/herself. Therefore, we used it as the other rule to cluster all "I" pronouns and the same speaker information together.

Given the labeled entity pairs, we then link them in different coreference chains by using the cluster information. As the Maximum Entropy classifier not only labels each entity-pair but also returns a confidence score of that label, we sort all positive pairs using their possibilities. For each positive entity-pair in the sorted list, if the two entities are in different clusters, we consider this to be a conflict, and withdraw this positive entity-pair; if one entity belongs to one cluster whereas the other does not belong to any cluster, the two entities will be both included in that cluster. This process is repeated until no more entities can be included in a cluster. Finally, we chain the rest of entity pairs together.

5 Results and Discussion

To evaluate the features and the chaining approach described in this paper, we design experiments described as follows. Since there are five different data types in the provided OntoNotes coreference data set, we create five different classifiers to process each of the data types. We used the features described in Section 3 to train the classifiers, and did the experiments using a Maximum Entropy classifier trained with the Mallet package (McCallum, 2002). We use the gold-standard data in the training set to train the five classifiers and test the classifiers on both gold and automatically-parsed data in the development data set. The MUC metric provided by the Task is used to evaluate the results.

5.1 Performance without Clustering

First, we evaluate the system by turning the clustering technique off during the process of creating coreference chains. For entity detection, we observe that for all five data types, i.e. Broadcast (BC), Broad news (BN), Newswire (NW), Magazine (MZ), and Web blog (WB), the NW and WB data types achieve relatively lower F1-scores, whereas the BC, BN, and MZ data types achieve higher per-

	BC	BN	NW	MZ	WB
			Without Clustering		
Gold	57.40 (64.92/51.44)	59.45 (63.53/55.86)	52.01 (59.71/46.07)	55.59 (62.90/49.80)	49.53 (61.16/41.62)
Auto	54.00 (61.28/48.26)	55.40 (59.05/52.17)	48.44 (55.32/43.09)	52.21 (59.78/46.33)	47.02 (58.33/39.39)
			With Clustering		
Gold	57.44 (64.12/52.03)	56.56 (58.10/55.09)	51.37 (56.64/46.99)	54.26 (60.07/49.47)	49.00 (60.09/41.36)
Auto	54.19 (60.82/48.87)	52.69 (54.07/51.37)	48.01 (52.74/44.05)	50.82 (56.76/46.01)	46.86 (57.49/39.55)

Table 1: Performance comparison of coreference identification between using and without using the clustering technique in chaining. Note that the results are listed in sequence of F1-scores (Recalls/Precisions). The results shown are based on MUC.

formance. Due to limited space, the performance table of entity detection is not included in this paper.

For coreference identification, as shown in Table 1, we observe pretty similar performance gaps among different data types. The NW and WB data types achieve the lowest F1-scores (i.e. 52.01% and 49.53% for gold standard data, and 48.44% and 47.02% for automatically-parsed data) among all the five data types. This can be explained by seeing that the entity detection performance of these two data types are also relatively low. The other three types achieves more than 55% and 52% F1-scores for gold and auto data, respectively.

These experiments that are done without using clustering techniques tend to indicate that the performance of entity detection has a positive correlation with that of coreference identification. Therefore, in the other set of experiments, we enable the clustering technique to improve coreference identification performance by increasing entity detection accuracy.

Metric	Recall	Precision	F1
MUC	59.94	45.38	51.65
BCUBED	72.07	53.65	61.51
CEAF (M)	45.67	45.67	45.67
CEAF (E)	29.43	42.54	34.79
BLANC	70.86	60.55	63.37

Table 2: Official results of our system in the CoNLL Task 2011. Official score is 49.32. ((MUC + BCUBED + CEAF (E))/3)

5.2 Performance with Clustering

After enabling the clustering technique, we observe an improvement in entity detection performance. This improvement occurs mainly in the cases of the NW and WB data types, which show low entity detection performance when not using the clustering technique. To be specific, the performance of the NW type on both the gold standard and automatic data improves by about 0.5%, and the performance of the WB type on the automatic data improves about 0.1%. In addition, the performance of the BC type on both the gold standard and automatic data also increases about 0.2% to 0.6%.

Although the clustering technique succeeds in improving entity detection performance for multiple data types, there is no obvious improvement gained with respect to coreference identification. This is quite incompatible with our observation in the experiments that do not utilize the clustering technique. Currently, we attribute this issue to the low accuracy rates of the clustering operation. For example, "H. D. Ye." and "Ye" can be estimated correctly to be coreferential by the Maxtent classifier, but the clustering algorithm puts them into different clusters since "H. D. Ye." is a PERSON type name entity while "Ye" is a ORG type name entity. Therefore, the system erroneously considers them to be a conflict and rejects them. We plan to investigate this issue further in our future work.

The official results of our system in the CoNLL Task 2011 are summarized in Table 2.

6 Conclusion

In this paper, we described the algorithm design and experimental results of Brandeis University in the CoNLL Task 2011. We show that several linguistic features perform well in the OntoNotes data set.

References

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Sameer S. Pradhan, Lance Ramshaw, Ralph Weischedel, Jessica MacBride, and Linnea Micciulla. 2007. Unrestricted Coreference: Identifying Entities and Events in OntoNotes. *International Conference on Semantic Computing (ICSC 2007)*, pages 446–453, September.

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

WordNet. 2010. Princeton University "About WordNet." WordNet. Princeton University. 2010. http://wordnet.princeton.edu.