ETS: An Error Tolerable System for Coreference Resolution

Hao Xiong , Linfeng Song , Fandong Meng , Yang Liu , Qun Liu and Yajuan Lü
Key Lab. of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
{xionghao,songlinfeng,mengfandong,yliu,liuqun,lvyajuan}@ict.ac.cn

Abstract

This paper presents our error tolerable system for coreference resolution in CoNLL-2011(Pradhan et al., 2011) shared task (closed track). Different from most previous reported work, we detect mention candidates based on packed forest instead of single parse tree, and we use beam search algorithm based on the Bell Tree to create entities. Experimental results show that our methods achieve promising results on the development set.

1 Introduction

Over last decades, there has been increasing interest on coreference resolution within NLP community. The task of coreference resolution is to identify expressions in a text that refer to the same discourse entity. This year, CoNLL[1] holds a shared task aiming to model unrestricted coreference in OntoNotes.[2] The OntoNotes project has created a large-scale, accurate corpus for general anaphoric coreference that covers entities and events not limited to noun phrases or a limited set of entity types. And Pradhan et al. (2007) have ever used this corpus for similar unrestricted coreference task.

Our approach to this year's task could be divided into two steps: mention identification and creation of entities. The first stage is conducted on the analysis of parse trees produced by input data. The official data have provided gold and automatic parse trees for each sentences in training and development set. However, according to statistics, almost 3% mentions have no corresponding constituents in automatic parse trees. Since only automatic parse trees will be provided in the final test set, the effect of parsing errors are inevitable. To alleviate this issue, based on given automatic parse trees, we modify a state-of-the-art parser (Charniak and Johnson, 2005) to generate packed forest, and determine mention candidates among all constituents from both given parse tree and packed forest. The packed forest is a compact representation of all parse trees for a given sentence. Readers can refer to (Mi et al., 2008) for detailed definitions.

Once the mentions are identified, the left step is to group mentions referring to same object into similar entity. This problem can be viewed as binary classification problem of determining whether each mention pairs corefer. We use a Maximum Entropy classifier to predict the possibility that two mentions refer to the similar entity. And mainly following the work of Luo et al. (2004), we use a beam search algorithm based on Bell Tree to obtain the global optimal classification.

As this is the first time we participate competition of coreference resolution, we mainly concentrate on developing fault tolerant capability of our system while omitting feature engineering and other helpful technologies.

2 Mention Detection

The first step of the coreference resolution tries to recognize occurrences of mentions in documents. Note that we recognize mention boundaries only on development and test set while generating training

[1] http://conll.bbn.com/

[2] http://www.bbn.com/ontonotes/

Proceedings of the 15th Conference on Computational Natural Language Learning: Shared Task, pages 76–80,
Portland, Oregon, 23-24 June 2011. ©2011 Association for Computational Linguistics

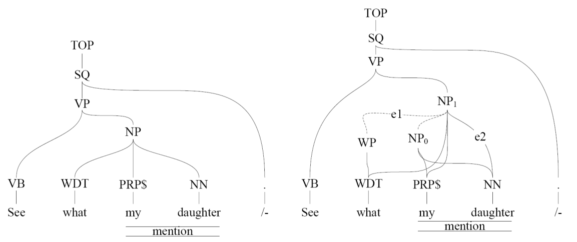

Figure 1: Left side is parse tree extracted from development set, and right side is a forest. "*my daughter*" is a mention in this discourse, however it has no corresponding constituent in parse tree, but it has a corresponding constituent NP_0 in forest.

instances using gold boundaries provided by official data.

The first stage of our system consists of following three successive steps:

- Extracting constituents annotated with *NP*, *NNP*, *PRP*, *PRP$* and *VBD* **POS tags** from single parse tree.
- Extracting constituents with the same tags as the last step from packed forest.
- Extracting Named Entity recognized by given data.

It is worth mentioning that above three steps will produce duplicated mentions, we hence collect all mentions into a list and discard duplicated candidates. The contribution of using packed forest is that it extends the searching space of mention candidates. Figure 1 presents an example to explain the advantage of employing packed forest to enhance the mention detection process. The left side of Figure 1 is the automatic parse tree extracted from development set, in which mention "*my daughter*" has no corresponding constituent in its parse tree. Under normal strategy, such mention will not be recognized and be absent in the clustering stage. However, we find that mention has its constituent NP_0 in packed forest. According to statistics, when using packed forest, only 0.5% mentions could not be recognized while the traditional method is 3%, that means the theoretical upper bound of our system reaches 99% compared to baseline's 97%.

Since the requirement of this year's task is to model unrestricted coreference, intuitively, we should not constraint in recognizing only noun phrases but also adjective phrase, verb and so on. However, we find that most mentions appeared in corpus are noun phrases, and our experimental results indicate that considering constituents annotated with above proposed POS tags achieve the best performance.

3 Determining Coreference

This stage is to determine which mentions belong to the same entity. We train a Maximum Entropy classifier (Le, 2004) to decide whether two mentions are coreferent. We use the method proposed by Soon, et al.'s to generate the training instances, where a positive instance is formed between current mention M_j and its closest preceding antecedent M_i, and a negative instance is created by paring M_j with each of the intervening mentions, M_{i+1}, M_{i+2},...,M_{j-1}.

We use the following features to train our classifier.

Features in Soon et al.'s work (Soon et al., 2001)

Lexical features

IS_PREFIX: whether the string of one mention is prefix of the other;

IS_SUFFIX: whether the string of one mention is suffix of the other;

ACRONYM: whether one mention is the acronym of the other;

Distance features

SENT_DIST: distance between the sentences containing the two mentions;

MEN_DIST: number of mentions between two mentions;

Grammatical features

IJ_PRONOUN: whether both mentions are pronoun;

I_NESTED: whether mention i is nested in another mention;

J_NESTED: whether mention j is nested in another mention;

Syntax features

HEAD: whether the heads of two mentions have the same string;

HEAD_POS: whether the heads of two mentions have the same POS;

HEA_POS_PAIRS: pairs of POS of the two mentions' heads;

Semantic features

WNDIST: distance between two mentions in WordNet;

I_ARG0: whether mention i has the semantic role of Arg0;

J_ARG0: whether mention j has the semantic role of Arg0;

IJ_ARGS: whether two mentions have the semantic roles for similar predicate;

In the submitted results, we use the L-BFGS parameter estimation algorithm with gaussian prior smoothing (Chen and Rosenfeld, 1999). We set the gaussian prior to 2 and train the model in 100 iterations.

3.1 Creation of Entities

This stage aims to create the mentions detected in the first stage into entities, according to the prediction of classifier. One simple method is to use a greedy algorithm, by comparing each mention to its previous mentions and refer to the one that has the highest probability. In principle, this algorithm is too greedy and sometimes results in unreasonable partition (Ng, 2010). To address this problem, we follow the literature (Luo et al., 2004) and propose to use beam search to find global optimal partition.

Intuitively, creation of entities can be casted as partition problem. And the number of partitions equals the Bell Number (Bell, 1934), which has a "closed" formula $B(n) = \frac{1}{e}\sum_{k=0}^{\infty}\frac{k^n}{k!}$. Clearly, this number is very huge when n is large, enumeration of all partitions is impossible, so we instead designing a beam search algorithm to find the best partition.

Formally, the task is to optimize the following objective,

$$\hat{y} = \arg\max_{\phi \in P} \sum_{e \in \phi} Prob(e) \tag{1}$$

where P is all partitions, $Prob(e)$ is the cost of entity e. And we can use the following formula to calculate the $Prob(e)$,

$$\begin{aligned} Prob(e) = & \sum_{i \in e, j \in e} pos(m_i, m_j) \\ & + \sum_{i \in e, j \notin e} neg(m_i, m_j) \end{aligned} \tag{2}$$

where $pos(m_i, m_j)$ is the score predicted by classifier that the possibility two mentions m_i and m_j group into one entity, and $neg(m_i, m_j)$ is the score that two mentions are not coreferent.

Theoretically, we can design a dynamic algorithm to obtain the best partition schema. Providing there are four mentions from A to D, and we have obtained the partitions of A, B and C. To incorporate D, we should consider assigning D to each entity of every partition, and generate the partitions of four mentions. For detailed explanation, the partitions of three mentions are [A][B][C], [AB][C], [A][BC] and [ABC], when considering the forth mention D, we generate the following partitions:

- [A][B][C][D], [AD][B][C], [A][BD][C], [A][B][CD]
- [AB][C][D], [ABD][C],[AB][CD]
- [A][BC][D], [AD][BC], [A][BCD]
- [ABC][D], [ABCD]

The score of partition [AD][B][C] can be calculated by $score([A][B][C]) + pos(A, D) + neg(B, D) + neg(C, D)$. Since we can computer pos and neg score between any two mentions in advance, this problem can be efficiently solved by dynamic algorithm. However, in practice, enumerating the whole partitions is intractable, we instead exploiting a beam with size k to store the top k partitions of current mention size, according to the score the partition obtain. Due to the scope limitation, we omit the detailed algorithm, readers can refer to Luo et al. (2004) for detailed description, since our approach is almost similar to theirs.

4 Experiments

4.1 Data Preparation

The shared task provided data includes information of lemma, POS, parse tree, word sense, predicate arguments, named entity and so on. In addition to those information, we use a modified in house parser to generate packed forest for each sentence in development set, and prune the packed forest with threshold p=3 (Huang, 2008). Since the OntoNotes involves multiple genre data, we merge all files and

	Mention	**MUC**	**BCUBED**	**CEAFM**	**CEAFE**	**BLANC**
baseline	58.97%	44.17%	63.24%	45.08%	37.13%	62.44%
baseline_gold	59.18%	44.48%	63.46%	45.37%	37.47%	62.36%
sys_forest	59.07%	44.4%	63.39%	45.29%	37.41%	62.41%
sys_btree	59.44%	44.66%	63.77%	45.62%	37.82%	62.47%
sys_forest_btree	59.71%	44.97%	63.95%	45.91%	37.96%	62.52%

Table 1: Experimental results on development set (F score).

	Mention	**MUC**	**BCUBED**	**CEAFM**	**CEAFE**	**BLANC**
sys1	54.5%	39.15%	63.91%	45.32%	37.16%	63.18%
sys2	53.06%	35.55%	59.68%	38.24%	32.03%	50.13%

Table 2: Experimental results on development set with different training division (F score).

take it as our training corpus. We use the supplied score toolkit [3] to compute MUC, BCUBED, CEAFM, CEAFE and BLANC metrics.

4.2 Experimental Results

We first implement a baseline system (*baseline*) that use single parse tree for mention detection and greedy algorithm for creation of entities. We also run the baseline system using gold parse tree, namely *baseline_gold*. To investigate the contribution of packed forest, we design a reinforced system, namely *sys_forest*. And another system, named as *sys_btree*, is used to see the contribution of beam search with beam size k=10. Lastly, we combine two technologies and obtain system *sys_forest_btree*.

Table 1 shows the experimental results on development data. We find that the system using beam search achieve promising improvement over baseline. The reason for that has been discussed in last section. We also find that compared to *baseline*, *sys_forest* and *baseline_gold* both achieve improvement in term of some metrics. And we are glad to find that using forest, the performance of our system is approaching the system based on gold parse tree. But even using the gold parse tree, the improvement is slight. [4] One reason is that we used some lexical and grammar features which are dominant during prediction, and another explanation is that packed forest enlarges the size of mentions but brings difficulty to resolve them.

To investigate the effect of different genres to develop set, we also perform following compared experiments:

- *sys1:* all training corpus + WSJ development corpus
- *sys2:* WSJ training corpus + WSJ development corpus

Table 2 indicates that knowledge from other genres can help coreference resolution. Perhaps the reason is the same as last experiments, where syntax diversity affects the task not very seriously.

5 Conclusion

In this paper, we describe our system for CoNLL-2011 shared task. We propose to use packed forest and beam search to improve the performance of coreference resolution. Multiple experiments prove that such improvements do help the task.

6 Acknowledgement

The authors were supported by National Natural Science Foundation of China, Contracts 90920004. We would like to thank the anonymous reviewers for suggestions, and SHUGUANG COMPUTING PLATFORM for supporting experimental platform.

[3] http://conll.bbn.com/download/scorer.v4.tar.gz

[4] Since under task requirement, singleton mentions are filtered out, it is hard to recognize the contribution of packed forest to mention detection, while we may incorrectly resolve some mentions into singletons that affects the score of mention detection.

References

E.T. Bell. 1934. Exponential numbers. *The American Mathematical Monthly*, 41(7):411–419.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Stanley F. Chen and Ronald Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. Technical report, CMU-CS-99-108.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio, June.

Z. Le. 2004. Maximum entropy modeling toolkit for Python and C++.

X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 135–es. Association for Computational Linguistics.

H. Mi, L. Huang, and Q. Liu. 2008. Forestbased translation. In *Proceedings of ACL-08: HLT*, pages 192–199. Citeseer.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1396–1411, Uppsala, Sweden, July. Association for Computational Linguistics.

Sameer Pradhan, Lance Ramshaw, Ralph Weischedel, Jessica MacBride, and Linnea Micciulla. 2007. Unrestricted Coreference: Identifying Entities and Events in OntoNotes. In *in Proceedings of the IEEE International Conference on Semantic Computing (ICSC)*, September 17-19.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL 2011)*, Portland, Oregon, June.

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.