Mention Detection: Heuristics for the OntoNotes annotations

Jonathan K. Kummerfeld, Mohit Bansal, David Burkett and **Dan Klein**
Computer Science Division
University of California at Berkeley
{jkk,mbansal,dburkett,klein}@cs.berkeley.edu

Abstract

Our submission was a reduced version of the system described in Haghighi and Klein (2010), with extensions to improve mention detection to suit the OntoNotes annotation scheme. Including exact matching mention detection in this shared task added a new and challenging dimension to the problem, particularly for our system, which previously used a very permissive detection method. We improved this aspect of the system by adding filters based on the annotation scheme for OntoNotes and analysis of system behavior on the development set. These changes led to improvements in coreference F-score of 10.06, 5.71, 6.78, 6.63 and 3.09 on the MUC, B^3, Ceaf-e, Ceaf-m and Blanc, metrics, respectively, and a final task score of 47.10.

1 Introduction

Coreference resolution is concerned with identifying *mentions* of entities in text and determining which mentions are referring to the same entity. Previously the focus in the field has been on the latter task. Typically, mentions were considered correct if their span was within the true span of a gold mention, and contained the head word. This task (Pradhan et al., 2011) has set a harder challenge by only considering exact matches to be correct.

Our system uses an unsupervised approach based on a generative model. Unlike previous work, we did not use the Bllip or Wikipedia data described in Haghighi and Klein (2010). This was necessary for the system to be eligible for the closed task.

The system detects mentions by finding the maximal projection of every noun and pronoun. For the OntoNotes corpus this approach posed several problems. First, the annotation scheme explicitly rejects noun phrases in certain constructions. And second, it includes coreference for events as well as things. In preliminary experiments on the development set, we found that spurious mentions were our primary source of error. Using an oracle to exclude all spurious mentions at evaluation time yielded improvements ranging from five to thirty percent across the various metrics used in this task. Thus, we decided to focus our efforts on methods for detecting and filtering spurious mentions.

To improve mention detection, we filtered mentions both before and after coreference resolution. Filters prior to coreference resolution were constructed based on the annotation scheme and particular cases that should never be mentions (e.g. single word spans with the EX tag). Filters after coreference resolution were constructed based on analysis of common errors on the development set.

These changes led to considerable improvement in mention detection precision. The heuristics used in post-resolution filtering had a significant negative impact on recall, but this cost was out-weighed by the improvements in precision. Overall, the use of these filters led to a significant improvement in F_1 across all the coreference resolution evaluation metrics considered in the task.

2 Core System

We use a generative approach that is mainly unsupervised, as described in detail in Haghighi and

Klein (2010), and briefly below.

2.1 Model

The system uses all three of the standard abstractions in coreference resolution; mentions, entities and types. A mention is a span in the text, the entity is the actual object or event the mention refers to, and each type is a group of entities. For example, "the Mountain View based search giant" is a mention that refers to the entity Google, which is of type organization.

At each level we define a set of properties (e.g. proper-head). For mentions, these properties are linked directly to words from the span. For entities, each property corresponds to a list of words, instances of which are seen in specific mentions of that entity. At the type level, we assign a pair of multinomials to each property. The first of these multinomials is a distribution over words, reflecting their occurrence for this property for entities of this type. The second is a distribution over non-negative integers, representing the length of word lists for this property in entities of this type.

The only form of supervision used in the system is at the type level. The set of types is defined and lists of prototype words for each property of each type are provided. We also include a small number of extra types with no prototype words, for entities that do not fit well in any of the specified types.

These abstractions are used to form a generative model with three components; a semantic module, a discourse module and a mention module. In addition to the properties and corresponding parameters described above, the model is specified by a multinomial prior over types (ϕ), log-linear parameters over discourse choices (π), and a small number of hyperparameters (λ).

Entities are generated by the semantic module by drawing a type t according to ϕ, and then using that type's multinomials to populate word lists for each property.

The assignment of entities to mentions is handled by the discourse module. Affinities between mentions are defined by a log-linear model with parameters π for a range of standard features.

Finally, the mention module generates the actual words in the span. Words are drawn for each property from the lists for the relevant entity, with a hyper-parameter for interpolation between a uniform distribution over the words for the entity and the underlying distribution for the type. This allows the model to capture the fact that some properties use words that are very specific to the entity (e.g. proper names) while others are not at all specific (e.g. pronouns).

2.2 Learning and Inference

The learning procedure finds parameters that are likely under the model's posterior distribution. This is achieved with a variational approximation that factors over the parameters of the model. Each set of parameters is optimized in turn, while the rest are held fixed. The specific update methods vary for each set of parameters; for details see Section 4 of Haghighi and Klein (2010).

3 Mention detection extensions

The system described in Haghighi and Klein (2010) includes every NP span as a mention. When run on the OntoNotes data this leads to a large number of spurious mentions, even when ignoring singletons.

One challenge when working with the OntoNotes data is that singleton mentions are not annotated. This makes it difficult to untangle errors in coreference resolution and errors in mention detection. A mention produced by the system might not be in the gold set for one of two reasons; either because it is a spurious mention, or because it is not co-referent. Without manually annotating the singletons in the data, these two cases cannot be easily separated.

3.1 Baseline mention detection

The standard approach used in the system to detect mentions is to consider each word and its maximal projection, accepting it only if the span is an NP or the word is a pronoun. This approach will introduce spurious mentions if the parser makes a mistake, or if the NP is not considered a mention in the OntoNotes corpus. In this work, we considered the provided parses and parses produced by the Berkeley parser (Petrov et al., 2006) trained on the provided training data. We added a set of filters based on the annotation scheme described by Pradhan et al. (2007). Some filters are applied before coreference resolution and others afterward, as described below.

Data Set	Filters	P	R	F
Dev	None	37.59	76.93	50.50
	Pre	39.49	76.83	52.17
	Post	59.05	68.08	63.24
	All	58.69	67.98	63.00
Test	All	56.97	69.77	62.72

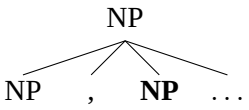
Table 1: Mention detection performance with various subsets of the filters.

3.2 Before Coreference Resolution

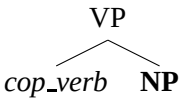
The pre-resolution filters were based on three reliable features of spurious mentions:

- Appositive constructions
- Attributes signaled by copular verbs
- Single word mentions with a POS tag in the set: EX, IN, WRB, WP

To detect appositive constructions we searched for the following pattern:



And to detect attributes signaled by copular structures we searched for this pattern:



where we used the fairly conservative set of copular verbs: {`is, are, was, 'm`}. In both cases, any mention whose maximal NP projection appeared as the bold node in a subtree matching the pattern was excluded.

In all three cases, errors from the parser (or POS tagger) may lead to the deletion of valid mentions. However, we found the impact of this was small and was outweighed by the number of spurious mentions removed.

3.3 After Coreference Resolution

To construct the post-coreference filters we analyzed system output on the development set, and tuned based on MUC and B^3 performance. The final set of filters used were:

Filters	MUC	B^3	Ceaf-e	Blanc
None	25.24	45.89	50.32	59.12
Pre	27.06	47.71	50.15	60.17
Post	42.08	62.53	43.88	66.54
All	42.03	62.42	43.56	66.60

Table 2: Precision for coreference resolution on the dev set.

Filters	MUC	B^3	Ceaf-e	Blanc
None	50.54	78.54	26.17	62.77
Pre	51.20	77.73	27.23	62.97
Post	45.93	64.72	39.84	61.20
All	46.21	64.96	39.24	61.28

Table 3: Recall for coreference resolution on the dev set.

- Filter if the head word is in a gazetteer, which we constructed based on behavior on the development set (head words found using the Collins (1999) rules)
- Filter if the POS tag is one of WDT, NNS, RB, JJ, ADJP
- Filter if the mention is a specific case of `you` or `it` that is more often generic (`you know`, `you can`, `it is`)
- Filter if the mention is any cardinal other than a year

A few other more specific filters were also included (e.g. `'s` when tagged as PRP) and one type of exception (if all words are capitalized, the mention is kept).

4 Other modifications

The parses in the OntoNotes data include the addition of structure within noun phrases. Our system was not designed to handle the NML tag, so we removed such nodes, reverting to the standard flattened NP structures found in the Penn Treebank.

We also trained the Berkeley parser on the provided training data, and used it to label the development and test sets.[1] We found that performance was

[1] In a small number of cases, the Berkeley parser failed, and we used the provided parse tree instead.

Filters	MUC	B^3	Ceaf-e	Ceaf-m	Blanc
None	33.67	57.93	34.43	42.72	60.60
Pre	35.40	59.13	35.29	43.72	61.38
Post	43.92	63.61	41.76	49.74	63.26
All	44.02	63.66	41.29	49.46	63.34

Table 4: F_1 scores for coreference resolution on the dev set.

Metric	R	P	F_1
MUC	46.39	39.56	42.70
B^3	63.60	57.30	60.29
Ceaf-m	45.35	45.35	45.35
Ceaf-e	35.05	42.26	38.32
Blanc	58.74	61.58	59.91

Table 5: Complete results on the test set

slightly improved by the use of these parses instead of the provided parses.

5 Results

Since our focus when extending our system for this task was on mention detection, we present results with variations in the sets of mention filters used. In particular, we have included results for our baseline system (None), when only the filters before coreference resolution are used (Pre), when only the filters after coreference resolution are used (Post), and when all filters are used (All).

The main approach behind the pre-coreference filters was to consider the parse to catch cases that are almost never mentions. In particular, these filters target cases that are explicitly excluded by the annotation scheme. As Table 1 shows, this led to a 1.90% increase in mention detection precision and 0.13% decrease in recall, which is probably a result of parse errors.

For the post-coreference filters, the approach was quite different. Each filter was introduced based on analysis of the errors in the mention sets produced by our system on the development set. Most of the filters constructed in this way catch some true mentions as well as spurious mentions, leading to significant improvements in precision at the cost of recall. Specifically an increase of 21.46% in precision and decrease of 8.85% in recall, but an overall increase of 12.74% in F_1-score.

As Tables 2 and 3 show, these changes in mention detection performance generally lead to improvements in precision at the expense of recall, with the exception of Ceaf-e where the trends are reversed. However, as shown in Table 4, there is an overall improvement in F_1 in all cases.

In general the change from only post-coreference filters to all filters is slightly negative. The final system used all of the filters because the process used to create the post-coreference filters was more susceptible to over-fitting, and the pre-coreference filters provided such an unambiguously positive contribution to mention detection.

6 Conclusion

We modified the coreference system of Haghighi and Klein (2010) to improve mention detection performance. We focused on tuning using the MUC and B^3 metrics, but found considerable improvements across all metrics.

One important difference between the system described here and previous work was the data available. Unlike Haghighi and Klein (2010), no extra data from Wikipedia or Bllip was used, a restriction that was necessary to be eligible for the closed part of the task.

By implementing heuristics based on the annotation scheme for the OntoNotes data set and our own analysis of system behavior on the development set we were able to achieve the results shown in Table 5, giving a final task score of 47.10.

7 Acknowledgments

We would like to thank the anonymous reviewers for their helpful suggestions. This research is supported by the Office of Naval Research under MURI Grant No. N000140911081, and a General Sir John Monash Fellowship.

References

Michael John Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, Philadelphia, PA, USA. AAI9926110.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Proceed-*

ings of NAACL, pages 385–393, Los Angeles, California, June. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Sameer S. Pradhan, Lance Ramshaw, Ralph Weischedel, Jessica MacBride, and Linnea Micciulla. 2007. Unrestricted coreference: Identifying entities and events in ontonotes. In *Proceedings of the International Conference on Semantic Computing*, pages 446–453, Washington, DC, USA. IEEE Computer Society.

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.