Data-driven Multilingual Coreference Resolution using Resolver Stacking

Anders Björkelund and **Richárd Farkas**
Institute for Natural Language Processing
University of Stuttgart
{anders,farkas}@ims.uni-stuttgart.de

Abstract

This paper describes our contribution to the CoNLL 2012 Shared Task.[1] We present a novel decoding algorithm for coreference resolution which is combined with a standard pair-wise coreference resolver in a stacking approach. The stacked decoders are evaluated on the three languages of the Shared Task. We obtain an official overall score of 58.25 which is the second highest in the Shared Task.

1 Introduction

In this paper we present our contribution to the CoNLL 2012 Shared Task (Pradhan et al., 2012). We follow the standard architecture where mentions are extracted in the first step, then they are clustered using a pair-wise classifier (see e.g., (Ng, 2010)). For English, the set of extracted mentions is filtered by removing non-referential occurrences of certain pronouns. Our coreference resolver at its core relies on a pair-wise classifier. To overcome the problems associated with the isolated pair-wise decisions, we devised a novel decoding algorithm which compares a mention to partially built clusters. For our Shared Task contribution we combined this algorithm with conventional pair-wise decoding algorithms in a stacking approach.

In the Shared Task evaluation, our system received an overall official score of 58.25, which is the second highest among the sixteen participants.[2]

2 Mention Extraction

Since all mentions are not annotated in Shared Task data, but only mentions that take part in coreference chains, training a general-purpose anaphoricity classifier is non-trivial. We thus implemented a high-recall, low-precision mention extraction module that allows the coreference resolver to see most of the possible mentions, but has to learn to sort out the non-referential mentions.

The mention extraction module relies mainly on the syntactic parse tree, but also on named entities (which were only provided for English in the predicted versions of the Shared Task data).

Since the part-of-speech tags vary a bit across the languages, so do our extraction rules: For Arabic, we extract all NP's, and all terminals with part-of-speech tags PRP and PRP$; for Chinese, we extract all NP's, and all terminals with part-of-speech tags PN and NR; for English, we extract all NP's, all terminals with part-of-speech tags PRP and PRP$, and all named entities.

Early experiments indicated that the English coreference resolver frequently makes mistakes related to non-referential instances of the pronouns *it* (often referred to as expletive or pleonastic in the literature), *we*, and *you* (generic mentions, which are not annotated according to the OntoNotes annotation guidelines). To address this issue, we developed a **referential/non-referential** mention classifier in order to identify these mentions. The classifier acts as a filter after the mention extraction module and removes clear cases of non-referential mentions.

Our basic assumption was that when these pro-

[1] The system is available for download on http://www.ims.uni-stuttgart.de/~anders/

[2] The overall score is the average of MUC, B^3, and CEAFE, averaged over all three languages

Proceedings of the Joint Conference on EMNLP and CoNLL: Shared Task, pages 49–55,
Jeju Island, Korea, July 13, 2012. ©2012 Association for Computational Linguistics

	$th = 0.5$			$th = 0.95$			
	Precision	Recall	F_1	Precision	Recall	F_1	# occurrences
it	75.41	61.92	68	86.78	38.65	53.48	10,307
we	65.93	41.61	51.02	75.41	24.20	36.64	5,323
you	79.10	74.26	76.60	88.36	51.59	65.15	11,297
Average	75.73	63.05	68.81	86.17	41.04	55.60	26,927

Table 1: Performance of the non-referential classifier used for English. Precision, recall, and F-measure are broken down by pronoun (top three rows), and the micro-average over all three (bottom row). The left side uses a probability threshold of 0.5, and the right one a threshold of 0.95. The last column denotes the number of occurrences of the corresponding token. All numbers are computed on the development set.

nouns do not participate in any coreference chain, they are examples of non-referential mentions. Based on this assumption, we extracted referential and non-referential examples from the training set and trained binary MaxEnt classifiers using the Mallet toolkit (McCallum, 2002).

Since the mentions filtered by these classifiers are permanently removed, they are never presented as potential mentions to the coreference resolver. Hence, we aim for a classifier that yields few false positives (i.e., mentions classified as non-referential although they were not). False negatives, on the other hand, may be passed on to the resolver, which, ideally, does not assign them to a cluster. The precision/recall tradeoff can easily be controlled by adjusting the threshold of the posterior probability of these classifiers, requiring a very high probability that a mention is non-referential. Preliminary experiments indicated that a threshold of 0.95 worked best when the coreference resolver was trained and evaluated on these filtered mentions.

We also found that the target pronouns should be handled separately, i.e., instead of training one single classifier we trained independent classifiers for each of the target pronouns. The individual performance of the classifiers, as well as the micro-average over all three pronouns are shown in Table 1, both using the default probability threshold of 0.5, and the higher 0.95. In the final, fine-tuned English coreference system, we found that the use of the classifiers with the higher threshold improved in all coreference metrics, and gave an increase of about 0.5 in the official CoNLL score.

The feature set used by the classifiers describes the (in-sentence) context of the pronoun. It consists of the uni-, bi-, and trigrams of word forms and POS tags in a window of ± 5; the position inside the sentence; the preceding and following verb and adjective; the distance to the following named entity; the genre of the document; and whether the mention is between quotes. For English, we additionally extended this general feature set by re-implementing the features of Boyd et al. (2005).

We investigated similar classifiers for Arabic and Chinese as well. We selected targets based on the frequency statistics of tokens being referential and non-referential on the training set and used the general feature set described above. However, these classifiers did not contribute to the more complex coreference system, hence the non-referential classifiers are included only in the English system.

3 Training Instance generation

To generate training instances for the pair-wise classifier, we employed the approach described by Soon et al. (2001). In this approach, for every extracted anaphoric mention m_j, we create a positive training instance with its closest preceding antecedent m_i: $P = \{(m_i, m_j)\}$. Negative examples are constructed by considering all the pairs of m_j and the (non-coreferent) mentions m_k between m_i and m_j: $N = \{(m_k, m_j) | i < k < j\}$. We extract the training examples on the version of the training set that uses predicted information, and restrict the mentions considered to the ones extracted by our mention extraction module. Using these training examples, we train a linear logistic regression classifier using the LibLinear package (Fan et al., 2008).

To create training examples for the English classifier, which uses the non-referential classifier for pronouns, we made a 10-fold cross-annotation on the training set with this classifier. I.e., the documents were partitioned into 10 sets $D_1, D_2, ..., D_{10}$, and when extracting training examples for documents in D_p, the non-referential classifier trained on $D_p^t = \bigcup_{i \neq p} D_i$ was applied.

4 Decoding

We implemented several decoding algorithms for our resolver. The two most common decoding algorithms often found in literature are the so-called *BestFirst* (henceforth BF) and *ClosestFirst* (CF) algorithms (Ng, 2010). Both work in a similar manner and consider mentions linearly ordered as they occur in the document. They proceed left-to-right and for every mention m_j, they consider all pairs (m_i, m_j), where m_i precedes m_j, and queries the classifier whether they are coreferent or not. The main difference between the two algorithms is that the CF algorithm selects the *closest* preceding mention deemed coreferent with m_j by the classifier, while the BF algorithm selects the *most probable* preceding mention. Most probable is determined by some sort of confidence measure of how likely two mentions are to corefer according to the classifier. For both algorithms, the threshold can also be tuned separately, e.g., requiring a probability larger than a certain threshold th_{coref} in order to establish a link between two mentions. Since the logistic classifiers we use directly model a probability distribution, we simply use the posterior probability of the $coref$ class as our confidence score.

Following Björkelund and Nugues (2011) we also implemented a decoder that works differently depending on whether m_j is a pronoun or not. Specifically, for pronouns, the CF algorithm is used, otherwise the BF algorithm is used. In the remainder, we shall refer to this decoder as *PronounsClosestFirst*, or simply PCF.

4.1 Disallowing transitive nesting

A specific kind of mistake we frequently saw in our output is that two clearly disreferent nested mentions are put in the same cluster. Although nestedness can be used as a feature for the classifier, and this appeared to improve performance, two nested mentions can still be put into the same cluster because they are both classified as coreferent with a different, preceding mention. The end result is that the two nested mentions are inadvertently clustered through transitivity.

For example, consider the two occurrences of the phrase *her mother* in (1) below. The spans in the example are labeled alphabetically according to their linear order in the document.[3] Before the resolver considers the last mention d, it has already successfully placed (a, c) in the same cluster. The first pair involving d is (c, d), which is correctly classified as disreferent (here, the feature set informs the classifier that (c, d) are nested). However, the pair (a, d) is easily classified as coreferent since the head noun of a agrees in gender and number with d (and they are not nested).

A different problem is related to named entities in possessive constructions. Consider (2), where our mention extractor extracted e, because it was an NP, and f, because it was tagged as a GPE by the named entity recognizer. Again, the pair (e, f) is correctly classified as disreferent, but both e and f are likely to be classified as coreferent with preceding mentions of *Taiwan*, since our string matching feature ignores possessive markers.

(1) ... she seemed to have such a good relationship with [[her]$_b$ mother]$_a$. Like [[her]$_d$ mother]$_c$ treated her like a human being ...

(2) [[Taiwan]$_f$'s]$_e$

To circumvent this problem, we let the decoders build the clusters incrementally as they work their way through a document and disallow this type of transitive nesting. For instance, when the decoder is trying to find an antecedent for d in (1), a and c have already been clustered together, and when the pair (c, d) is classified as disreferent, the decoder is constrained to skip over other members of c's cluster as it moves backwards in the document. This modification gave an increase of about 0.6 in the CoNLL score for English, and about 0.4 for Arabic and Chinese, and we used this constraint whenever we use the above-mentioned decoders.

4.2 A Cluster-Mention Decoding Algorithm

The pair-wise classifier architecture has, justifiably, received much criticism as it makes decisions based on single pairs of mentions only. We therefore de-

[3] We impose a total order on the mentions by sorting them by starting point. For multiple mentions with the same starting point, the longer is considered to precede the shorter.

vised a decoding algorithm that has a better perspective on entire clusters.

The algorithm works by incrementally merging clusters as mentions are processed. Initially, every mention forms its own cluster. When the next mention m_j is processed, it is compared to all the preceding mentions, $M = \{m_i | i < j\}$. The score of linking m_j with m_i is defined according to:

$$score(m_i, m_j) = (\prod_{m_c \in C} P(coref | (m_c, m_j)))^{1/|C|}$$

where $P(coref|(m_i, m_j))$ is the posterior probability that m_i and m_j are coreferent according to the pair-wise classifier, and C denotes the cluster that m_i belongs to.

After considering all preceding mentions, the cluster of m_j is merged with the cluster of the mention with which it had the highest score, assuming this score is higher than a given threshold th_{coref}. Otherwise it remains in its own cluster.

The task of the $score$ function is to capture cluster-level information. When m_j is compared to a mention m_i, the score is computed as the geometric mean of the product of the probabilities of linking m_j to *all* mentions in the cluster that m_i belongs to. Also note that for two preceding mentions m_{i_1} and m_{i_2} that already belong to the same cluster, $score(m_{i_1}, m_j) = score(m_{i_2}, m_j)$. I.e., the score is the same when m_j is compared to all mentions belonging to the same cluster. Since this algorithm works by maximizing the average probability for linking a mention, we dub this algorithm *AverageMaxProb*, or AMP for short.

It should also be noted that other definitions of the cluster score function $score$ are conceivable.[4] However, initial experiments with other cluster score functions performed worse than the definition above, and time prevented us from exploring this conclusively.

Contrary to the pair-wise decoding algorithms where pair-wise decisions are made in isolation, the order in which mentions are processed make a difference to the AMP decoder. It is generally accepted that named entities are more informative and easier to resolve than common noun phrases and pronouns. To leverage this, we follow Sapena et al. (2010) who reorder mentions based on mention type. Specifically, we first process proper noun phrases, then common noun phrases, and finally pronouns. This implies that common noun phrases have to have a reasonable agreement not only with preceding proper noun phrases of a cluster, but *all* proper noun phrases in a document (where reasonable means that the geometric average of all posterior probabilities stay reasonably high). Similarly, pronouns are forced agree reasonably with all proper and common nouns phrases in a given cluster, and not only the preceding ones. Early experiments showed an increase in performance using reordering, and we consequently used reordering for all languages in the experiments.

5 Features

An advantage of the pair-wise model and of the linear classifiers we use is that they can easily accommodate very large feature spaces, while still remaining reasonably fast. We exploited this by building a large number of parametrized feature templates, that allowed us to experiment easily and quickly with different feature sets. Additionally, since our classifiers are linear, we also evaluated a large number of feature conjunctions, which proved to be crucial to gain reasonable performance.

Due to space restrictions we can not list the complete set of features used in this paper but mention briefly what type of features we used. Most of them are taken from previous work on coreference resolution (Soon et al., 2001; Luo and Zitouni, 2005; Sapena et al., 2010; Björkelund and Nugues, 2011). For a complete list of features the reader can refer to the download of the resolver, which includes the feature sets and parameters used for every language.

One set of feature templates we use is based on surface forms and part-of-speech tags of the first and last, previous and following, and head tokens of the spans that make up mentions. Another set of templates are based on the syntax trees, including both subcategorization frames as well as paths in the syntax tree. To extract head words of mentions, we used the head percolation rules of Choi and Palmer (2010) for Arabic and English, and those of Zhang

[4] In the extreme case, one could take the maximum of the link probabilities over the mentions that belong to the cluster C, in which case the algorithm collapses into the BF algorithm.

and Clark (2011) for Chinese.

While Chinese and English display no or relatively small variety in morphological inflection, Arabic has a very complex morphology. This means that Arabic suffers from greater data sparseness with respect to lexical features. This is exaggerated by the fact that the Arabic training set is considerably smaller than the Chinese and English ones. Hence, we used the lemmas and unvocalised Buckwalter forms that were provided in the Arabic dataset.

We also tried to extract number and gender information based on affixes of Arabic surface forms. These features did, however, not help much. We did however see a considerable increase in performance when we added features that correspond to the Shortest Edit Script (Myers, 1986) between surface forms and unvocalised Buckwalter forms, respectively. We believe that edit scripts are better at capturing the differences in gender and number signaled by certain morphemes than our hand-crafted rules.

6 Resolver Stacking

In Table 2 we present a comparison of the BF, PCF, and AMP resolvers. We omit the results of the CF decoder, since it always did worse and the corresponding numbers would not add more to the picture. The table shows F-measures of mention detection (MD), the MUC metric, the B^3 metric, and the entity-based CEAF metric. The CoNLL score, which is computed as the arithmetic mean of MUC, B^3, and CEAFE, is shown in the last row.

Comparing the AMP decoder to the pair-wise decoders, we find that it generally – i.e., with respect to the CoNLL average – performs worse though it always obtains higher scores with the CEAFE metric. When we looked at the precision and recall for mention detection, we also found that the AMP decoder suffers from lower recall, but higher precision. This led us to conclude that this decoder is more conservative in terms of clustering mentions, and builds smaller, but more consistent clusters. We could also verify this when we computed average cluster sizes on the output of the different decoders.

In order to combine the strengths of the AMP decoder and the pair-wise decoders we employed *stacking*, i.e., we feed the output of one resolver as input to a second. The second resolver is informed about the decision of the first one by introducing an additional feature that encodes the decision of the first resolver. This feature can take five values, depending on how the first resolver treated the two mentions in question: NEITHER, when none of the mentions were placed in a cluster; IONLY, when only the first (antecedent) mention was placed in a cluster; JONLY, when only the second (anaphor) mention was placed in a cluster; COREF, when both mentions were placed in the same cluster; and DISREF, when both mentions were clustered, but in different clusters.

Arabic	BF	PCF	AMP	Stacked
MD	58.63	58.49	58.21	60.51
MUC	45.8	45.4	43.2	46.66
B^3	66.65	66.56	66.39	66.3
CEAFE	41.52	41.58	43.1	42.57
CoNLL	51.32	51.18	50.9	51.84
Chinese	BF	PCF	AMP	Stacked
MD	67.22	67.19	66.79	67.61
MUC	59.58	59.43	57.23	59.84
B^3	72.9	72.82	72.7	73.35
CEAFE	46.99	46.98	48.25	47.7
CoNLL	59.82	59.74	59.39	60.30
English	BF	PCF	AMP	Stacked
MD	74.33	74.42	73.75	74.96
MUC	66.76	66.93	62.74	67.12
B^3	70.96	71.11	68.05	71.18
CEAFE	45.46	45.83	46.49	46.84
CoNLL	61.06	61.29	59.09	61.71

Table 2: Performance of different decoders on the development set for each language. The configuration of the Stacked systems is described in detail in Section 7.

In addition to the stacking feature, the second resolver uses the exact same feature set as the first resolver. To generate the information for the stack feature for training, we made a 10-fold cross-annotation on the training set, in the same way that we cross-annotated the non-referential classifier for English.

In early stacking experiments, we experimented with several combinations of the different decoders. We found that stacking different pair-wise decoders did not give any improvement. We believe the reason for this is that these decoders are too similar and hence can not really benefit from each other. However, when we used the AMP decoder as the first step, and a pair-wise decoder as the second, we saw an increase in performance, particularly with respect to the CEAFE metric.

7 Feature and Parameter Tuning

For every language we tuned decoder parameters and feature sets individually. The feature sets were tuned semi-automatically by evaluating the addition of a new feature template (or template conjunction) to a baseline set. Ideally, we would add feature templates to the baseline set incrementally one at a time, following a cross-validation on the training set. However, to reduce computational effort and time consumption, we resorted to doing only one or two folds out of a 4-fold cross-validation, and adding the two to three most contributing templates in every iteration to the baseline set. The feature sets were optimized to maximize the official CoNLL score using the standard BF decoder.

For the final submission we tuned the thresholds for each decoder, and the choice of pair-wise decoder to use as the second decoder for each language. Modifying the threshold of the AMP decoder gave very small differences in overall score and we kept the threshold for this decoder at 0.5. However, when we increased the probability threshold for the second resolver, we found that performance increased across all languages.

The choice of decoder for the second resolver, and the probability threshold for this, was determined by a 4-fold cross-validation on the training set. For our final submission, as well as in the column *Stacked* in Table 2, we used the following combinations: For Arabic, the threshold was set to 0.60, and the PCF decoder was used; for Chinese, the threshold was set to 0.65, and the BF decoder was used; for English, the threshold was set to 0.65, and the PCF decoder was used.

8 Official Results

The final scores of our system are presented in Table 3. The table also includes the results on the supplementary tracks: gold mention boundaries (GB), when the perfect boundaries of mentions were given; and gold mentions (GM), when only the mentions in the gold standard were given (with gold boundaries). For all three settings we used the same model, which was trained on the concatenation of the training and the development sets.

Arabic	PM	GB	GM
MD	60.55	60.61	76.43
MUC	47.82	47.90	60.81
B^3	68.54	68.61	67.29
CEAFE	44.3	44	49.32
CoNLL	53.55	53.50	59.14
Chinese	PM	GB	GM
MD	66.37	71.02	83.47
MUC	58.61	63.56	76.85
B^3	73.10	74.52	76.30
CEAFE	48.19	50.20	56.61
CoNLL	59.97	62.76	69.92
English	PM	GB	GM
MD	75.38	75.3	86.16
MUC	67.58	67.29	78.70
B^3	70.26	69.70	72.67
CEAFE	45.87	45.27	53.23
CoNLL	61.24	60.75	68.20

Table 3: Performance on the shared task test set. Using predicted mentions (PM; i.e., the official evaluation), gold mentions boundaries (GB), and gold mentions (GM).

Compared to the results on the development set (cf. Table 2), we see a slight drop for Chinese and English, but a fairly big increase for Arabic. Given that Chinese and English have the biggest training sets, we speculate that the increase in Arabic might stem from the increased lexical coverage provided by training on both the training and the development sets.

9 Conclusion

We have presented a novel cluster-based coreference resolution algorithm. This algorithm was combined with conventional pair-wise resolution algorithms in a stacking approach. We applied our system to all three languages in the Shared Task, and obtained an official overall final score of 58.25 which was the second highest in the Shared Task.

Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) via the SFB 732 “Incremental Specification in Context”, projects D4 (PI Helmut Schmid) and D8 (PI Jonas Kuhn).

References

Anders Björkelund and Pierre Nugues. 2011. Exploring lexicalized features for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 45–50, June.

Adriane Boyd, Whitney Gegg-Harrison, and Donna Byron. 2005. Identifying non-referential it: A machine learning approach incorporating linguistically motivated patterns. In *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, pages 40–47, June.

Jinho D. Choi and Martha Palmer. 2010. Robust Constituent-to-Dependency Conversion for English. In *Proceedings of 9th Treebanks and Linguistic Theories Workshop (TLT)*, pages 55–66.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Xiaoqiang Luo and Imed Zitouni. 2005. Multi-lingual coreference resolution with syntactic features. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 660–667, October.

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

Eugene W. Myers. 1986. An O(ND) difference algorithm and its variations. *Algorithmica*, 1:251–266.

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, July.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning (CoNLL 2012)*.

Emili Sapena, Lluís Padró, and Jordi Turmo. 2010. A global relaxation labeling approach to coreference resolution. In *Coling 2010: Posters*, pages 1086–1094, August.

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.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.