Collaborative Partitioning for Coreference Resolution

Olga Uryupina[◊] and **Alessandro Moschitti**
[◊]DISI, University of Trento 38123 Povo (TN), Italy
Qatar Computing Research Institute, HBKU, 34110, Doha, Qatar
{uryupina,amoschitti}@gmail.com

Abstract

This paper presents a collaborative partitioning algorithm—a novel ensemble-based approach to coreference resolution. Starting from the all-singleton partition, we search for a solution close to the ensemble's outputs in terms of a task-specific similarity measure. Our approach assumes a loose integration of individual components of the ensemble and can therefore combine arbitrary coreference resolvers, regardless of their models. Our experiments on the CoNLL dataset show that collaborative partitioning yields results superior to those attained by the individual components, for ensembles of both strong and weak systems. Moreover, by applying the collaborative partitioning algorithm on top of three state-of-the-art resolvers, we obtain the second-best coreference performance reported so far in the literature (MELA v08 score of 64.47).

1 Introduction

Coreference resolution has been one of the key areas of NLP for several decades. Major modeling breakthroughs have been achieved, not surprisingly, following three successful shared tasks, such as MUC (Hirschman and Chinchor, 1997), ACE (Doddington et al., 2004) and, most recently, CoNLL (Pradhan et al., 2011; Pradhan et al., 2012). As of today, several high-performing systems are available publicly and, in addition, novel algorithms are being proposed regularly, even if without any code release. Our study aims at making a good use of these resources through a novel ensemble resolution method.

Coreference is a heterogeneous task that requires a combination of accurate and robust processing for relatively easy cases (e.g., name-matching) with very complex modeling of difficult cases (e.g., nominal anaphora or some types of pronouns). The general feeling in the community is that we are currently approaching the upper bound for the easy cases and our next step should involve more complex resolution. If true, this means that most state-of-the-art systems should produce very similar outputs: correctly resolving easy anaphora and failing on less trivial examples. Table 1 scores the outputs of the three best systems from the CoNLL-2012 shared task against each other. As it can be seen, the three systems are rather different, each of them being only slightly closer to each other than to the gold key.[1] This suggests that a meta-algorithm could merge their outputs in an intelligent way, combining the correct decisions of individual systems to arrive at a superior partition.

Although several coreference resolution toolkits exist for over a decade, to our knowledge, there have been no attempts at trying to merge their outputs. The very few ensemble methods reported in the literature focus on combining several resolution strategies within the same system. Following the success of the CoNLL shared task (Pradhan et al., 2011; Pradhan et al., 2012), however, multiple complex approaches have been investigated, with very different underlying models. This means that a re-implementation of all these algorithms within a single system requires a considerable engineering effort. In the present study, we combine the final outputs of the individual systems, without making any assumptions on their specifications. This means that our approach is completely modular, allowing to combine third-party software as black boxes.

The present study aims at finding a partition

[1] Across all the systems, the two most different submissions are zhekova vs. li (34.10 MELA) and the two closest ones are chunyang vs. shou (95.85 MELA).

Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), pages 47–57,
Vancouver, Canada, August 3 - August 4, 2017. ©2017 Association for Computational Linguistics

	key	fernandes	martschat	bjorkelund
fernandes	60.64	100	66.74	67.07
martschat	57.67		100	64.22
bjorkelund	57.41			100

Table 1: Scoring top CoNLL-2012 systems against each other, MELA v08.

combining the outputs of individual coreference resolvers in a collaborative way. To this end, we search the space of possible partitions, starting from the all-singleton solution and incrementally growing coreference entities, with the objective of getting a partition similar to the individual outputs. As a measure of similarity, we rely on task-specific metrics, such as, for example, MUC or MELA scores. To our knowledge, this is the first ensemble-based approach to coreference, operating directly on the partition level. While traditional ensemble techniques, such as boosting or co-training, have been successfully used for coreference resolution before, they are applicable to classification tasks and can only be used on lower levels (e.g., for classifying mention pairs). Combining partitions directly is a non-trivial problem that requires an extra modeling effort.

The rest of the paper is organized as follows. In the next section, we discuss the previous ensemble-based approaches to coreference resolution. Section 3 presents our collaborative partitioning algorithm. In Section 4, we evaluate our approach on the English portion of the OntoNotes dataset. Section 5 summarizes our contributions and highlights directions for future research.

2 Related Work

Only very few studies have so far investigated possibilities of using multiple resolvers for coreference. The first group of approaches aim at parameter optimization for choosing the best overall partition from the components' outputs. This line of research is motivated by the fact that in most approaches to coreference, the underlying classifier does not take into account the task metric, such as, for example, MUC or MELA scores. For instance, in the classical mention-pair model (Soon et al., 2001), the classifier is trained to distinguish between coreferent and non-coreferent pairs. The output of this classifier is then processed heuristically to create coreference partitions. There is therefore no guarantee that the classifier optimized on pairs would lead to the best-scoring partition. One way to overcome this issue involves training a collection of models and then picking the globally best one on the development data (Munson et al., 2005; Saha et al., 2011). Another possible solution is to learn a ranker that would pick the best model on a per-document basis, using partition-specific features (Ng, 2005). While these approaches can integrate arbitrary systems, they only allow to pick the best output partition, thus, only considering a single solution at a given time. Our algorithm, on the contrary, builds a new partition in a collaborative way, manipulating entities produced by individual components.

The second research line involves training ensembles of classifiers within the same model, using bagging, boosting or co-training (Vemulapalli et al., 2009; Ng and Cardie, 2003b; Ng and Cardie, 2003a; Kouchnir, 2004). Building upon these studies, Rahman and Ng (2011) combine different coreference algorithms in an ensemble-based approach. For each mention in the document, they run several models (mention-pair, mention-ranking, entity-ranking) and heuristically merge their outputs. All these approaches, however, assume a very tight integration of individual components into the ensemble. Thus, they all assume the same set of mentions to be classified.[2] Moreover, most algorithms can only make ensembles of rather similar components, for example, varying feature sets or parameters within the same model of coreference. While Rahman and Ng (2011) allow for a combination of different models, they do it via model-specific rules, assuming the same set of mentions and a left-to-right per-mention resolution strategy—so a completely different novel model cannot be integrated. Finally, most ensembles use some internal information from their individual components, e.g., the confidence scores for mention-pairs. In practice, these considerations mean that all the individual systems should be re-implemented in a single framework before they can be combined in an ensemble. Our study, on the contrary, makes no assumptions about individual components. We combine their outputs at the partition level, without any requirements on their internal structure. Thus, the individual systems can rely on different mention detection approaches. They can have arbitrary models. We do not use any system-internal information, which

[2] The CoNLL systems differ considerably with respect to their underlying mentions, thus, the mention detection F-score between two systems varies from 50.11 (`xinxin` vs. `li`) to 99.07 (`chunyang` vs. `shou`).

allows us to use individual components as black boxes. Most importantly, our approach can be run without any modification on top of any resolver, present or future, thus benefiting from other studies on coreference and advancing the state of the art performance.

The machine learning community offers several algorithms combining multiple solutions for tasks going beyond simple classification or regression. The work of Strehl and Ghosh (2003) is of particular relevance for our problem. Thus, Strehl and Ghosh (2003) introduce the task of *ensemble* or *consensus clustering*, where the combiner aims at creating a meta-clustering on top of several individual solutions, without accessing their internal representations, e.g., features. The formulation of Strehl and Ghosh (2003) is identical to ours. However, there are several important differences. Thus, Strehl and Ghosh (2003) focus on the clustering problem, in particular, for large sets of data points. They show that the optimal solution to the consensus clustering problem is not computationally feasible and investigate several very greedy approaches.

Although coreference is formally a partitioning problem, the setting is rather different from a typical clustering scenario. Thus, individual mentions and mention properties are very important for coreference and should carefully be assessed one by one. The resolution clues are very heterogeneous and different elements (mentions) of clusters (entities) can be rather dissimilar in a strict sense. This is why, for example, clustering evaluation measures are not reliable for coreference—and, indeed, task-specific metrics have been put forward. While algorithms of (Strehl and Ghosh, 2003) constitute the state of the art in the ensemble clustering in general, we propose a coreference-specific approach. More specifically, (i) while Strehl and Ghosh (2003) rely on task-agnostic measures of similarity between partitions (mutual information), approximating the search for its maximum with various heuristics, we explicitly integrate coreference metrics, such as MUC and MELA and (ii) since our partitions are much smaller than typical clustering outputs, we can afford a less greedy agglomerative search strategy, again, motivated by the specifics of the final task. In our future work, we plan to evaluate our approach against the general-purpose algorithms proposed in (Strehl and Ghosh, 2003).

Algorithm 1 Collaborative Partitioning

```
Require: $P = \{p^1..p^n\}$: list of partitions generated by
    $n$ systems; each partition $p^i$ is a set of entities $p^i = \{e^i_1..e^i_{k_n}\}$, each entity is a set of mentions $m$
Require: $coreference\_score$: an external metric, e.g.
    MUC or MELA
begin
create a list of all the mentions $M = \{m_1..m_k\}$
init the all-singleton partition $p = \{e_1..e_k\}, e_i = \{m_i\}$
while $\|p\| > 1$ do
    $current\_similarity = vote(p, P)$
    $max = 0$
    for all $e_a, e_b \in p$ do
        $p' = p \cup \{e_a \cup e_b\} \setminus \{e_a\} \setminus \{e_b\}$
        $cand\_similarity = vote(p', P)$
        if $cand\_similarity > max$ then
            $max = cand\_similarity, maxp = p'$
    if $max < current\_similarity$ then
        $break$;
    $p = maxp$
end
function VOTE(p,$P$)
    $sim = 0$
    for all $p^i \in P$ do
        $sim+ = coreference\_score(p, p^i)$
    return $sim$
```

3 Collaborative Partitioning

This section first describes our collaborative partitioning algorithms, summarized in Algorithm 1 and then addresses technical details essential for running it in a practical scenario. The main idea behind collaborative partitioning is rather straightforward: we aim at finding a partition that is similar to all the outputs produced by the individual components of the ensemble. To implement this strategy, we have to specify two aspects: (a) the procedure to effectively search the space of possible partitions generating outputs to be tested and (b) the way to measure similarity between a candidate partition and a component's output. In both cases, we propose task-specific solutions.

Thus, we start with the all-singleton partition, where each mention makes its own entity and then try to incrementally grow our entities. At each iteration, we try to merge two clusters, comparing the similarity to the components' outputs before and after the candidate merge. If a candidate merge leads to the highest voting score, we execute this merge and proceed to the next iteration. If no candidate merges improve the similarity score for more than a predefined termination threshold, the algorithm stops. Several things should be noted. First, when trying to build a new partition, we only allow for merging: we never go back and split already constructed entities. This

<table>
<tr><td colspan="4">President Clinton has told a memorial service for the victims of the deadly bomb attack on the USS Cole that justice will prevail . Mr. Clinton promised the gathering at the Norfolk Naval station Wednesday that those who carried out the deadly attack that killed 17 sailors will be found . To those who attacked them , we say you will not find a safe harbor , we will find you and justice will prevail . Meanwhile , in Yemen President Ali Abdul Salay said important evidence had been uncovered in the investigation . President Salay was quoted as saying two people responsible for the blast were killed in a suicide mission and that the attack had been planned for a long time . His comments were not immediately confirmed by US officials who are leading the investigation with Yemen 's help .</td></tr>
<tr><th>fernandes</th><th>martschat</th><th>bjorkelund</th><th>ensemble</th></tr>
<tr><td>[1,2]: President Clinton
[25,26] Mr. Clinton</td><td>[1,2]: President Clinton
[25,26] Mr. Clinton</td><td>[1,2]: President Clinton
[25,26] Mr. Clinton</td><td>[1,2] President Clinton
[25,26] Mr. Clinton</td></tr>
<tr><td>[12,15]: the deadly bomb attack
[115,116]: the attack</td><td>[115,116]: the attack
[41,47]: the deadly attack .. sailors</td><td>[12,19]: the deadly .. Cole
[115,116]: the attack
[41,47]: the deadly attack .. sailors</td><td>[12,15]: the deadly bomb attack
[115,116]: the attack
[41,47]: the deadly attack .. sailors</td></tr>
<tr><td rowspan="2">[56,56]: them
[37,47]: those who carried .. sailors</td><td>[46,47]: 17 sailors
[56,56] them</td><td>[46,47]: 17 sailors
[56,56] them</td><td>[46,47]: 17 sailors
[56,56] them</td></tr>
<tr><td rowspan="2">[37,47]: those who carried .. sailors
[53,56]: those who attacked them</td><td></td><td rowspan="2">[37,47]: those who carried .. sailors
[53,56]: those who attacked them
[60,60]: you
[71,71]: you</td></tr>
<tr><td>[53,56]: those who attacked them
[60,60]: you
[71,71]: you</td><td>[60,60]: you
[71,71]: you</td></tr>
<tr><td>[58,58]: we
[68,68]: we</td><td>[58,58]: we
[68,68]: we</td><td>[58,58]: we
[68,68]: we</td><td>[58,58]: we
[68,68]: we</td></tr>
<tr><td>[80,80]: Yemen
[140,141]: Yemen 's</td><td>[80,80]: Yemen
[140,141]: Yemen 's</td><td>[80,80]: Yemen
[140,141]: Yemen 's</td><td>[80,80]: Yemen
[140,141]: Yemen 's</td></tr>
<tr><td>[81,84]: President Ali Abdul Salay
[95,96]: President Salay
[125,125]: His</td><td>[81,84]: President Ali Abdul Salay
[95,96]: President Salay
[125,125]: His</td><td>[95,96]: President Salay
[125,125]: His</td><td>[81,84]: President Ali Abdul Salay
[95,96]: President Salay
[125,125]: His</td></tr>
<tr><td>[92,93]: the investigation
[137,138]: the investigation</td><td></td><td>[92,93]: the investigation
[137,142]: the inv. with Yemen 's help</td><td>[92,93]: the investigation
[137,138]: the investigation</td></tr>
</table>

Table 2: Collaborative partitioning on a sample OntoNotes document: 3 top systems and their ensemble, using MELA similarity. Each row corresponds to a mention, each (multi-row) cell corresponds to an entity created by a specific system. Bracketed numbers indicate word ids.

decision is motivated by the cost of a single operation: while there is only one way to merge two entities, there are exponentially many ways to split an entity in two, making the latter operation much more computationally expensive. Second, unlike most approaches to coreference, we do not process the text in the left-to-right order. Instead, we consider the whole set of mentions from the initial iteration, doing first the merges supported by the majority of the components in the ensemble.

To compute the voting score, we first define the similarity between two partitions, based on coreference metrics, as implemented in the CoNLL scorer (Pradhan et al., 2014): we score our generated partitions against the outputs of the ensemble components. This way we ensure that the final partition is related to the individual outputs in the way that is relevant for the task. There are multiple ways to derive the voting score from existing metrics. The parameters to consider here are: the specific measure to be used (e.g., MUC vs. CEAF vs. MELA), the granularity (e.g., whether to measure the increase/decrease of the specific metric as a continuous or binary value) and the way to combine measures from the different ensemble components in a single score (e.g., weighted vs. unweighted voting). In Section 3.1 below, we discuss several practical considerations for making this choice.

Note that our approach does not make any assumptions about mention detection for individual components: to initialize the run, we simply lump together all the mentions. This, however, leads to performance drops if several individual systems suggest different boundaries for the same mention: the final solution will then keep all the variants merging them into the same entity. To avoid this issue, we implement a post-processing cleanup step: if the final solution contains entities with nested mentions, we keep the most popular variants (or the shorter one for the same popularity). This post-processing helps us avoid any complex merging machinery at the mention level.

Table 2 shows a sample OntoNotes document with outputs of the three top systems and the partition created by the collaborative ensemble. Some entities (e.g. *Yemen*) are easy for all the systems. Some entities (*attack*; *investigation*; *Salay*) are recovered fully only by two systems, probably for the lack of required features. Note that although each system misses some coreference relations, altogether they resolve all the three entities, leading to a considerable improvement in the collaborative partition. Finally, the two entities for *attackers* and *sailors*, central to the document, are represented with pronominal mentions that are hard to resolve. Not surprisingly, the systems make several spurious decisions w.r.t. these entities. The collaborative partitioning algorithm, however, manages to filter out erroneous assignments and produce the correct partition.

3.1 Performance Issues

The collaborative partitioning algorithm starts from the all-singleton solution and tries to incrementally merge entities. Each candidate merge is evaluated with the coreference scorer. This means that, in the worst case, the system requires $0(n^3)$ scorer runs, where n is the total number of mentions: it does n merges and for each merge i, it searches for a pair among $n - i + 1$ entities that maximizes the overall similarity score, requiring $\frac{(n-i+1)*(n-i)}{2}$ scorer runs. This can become prohibitively slow, making the approach not practical. Below we discuss three solutions to speed up the algorithm.

First, the voting function can be simplified. Thus, instead of using continuous similarity values (i.e., how much a candidate merge brings the solution closer to the components' output via increasing or decreasing the specific coreference metric), we can rely on binary indicators: the component up-/down- votes a merge if the metric's value increases/decreases. To compute the final score, we use unweighted voting (or, alternatively, weighted voting with very simple integer weights). This way, the final score can only take a small number of values and, for each merge, we can stop the search once the highest possible score is observed, instead of assessing all the $\frac{(n-i+1)*(n-i)}{2}$ possible pairs. This trick does not affect the worst-case complexity, but can help a lot on the average. Moreover, a simple voting function is necessary for the second speed-up adjustment.

Each merge only involves two entities. Thus, at the merge iteration i, the system observes $n-i$ entities it has already seen before and one new entity generated at the merge iteration $i-1$. To speed up the processing, we can therefore store voting values for merge attempts and reuse them at each iteration. With this adjustments, the algorithm needs only to evaluate candidate merges with the newly constructed entity and therefore each iteration requires a linear number of scoring runs, leading to $O(n^2)$ runs overall. Two considerations should be taken into account. Suppose we evaluate a merge attempt for two entities, e_1 and e_2, at the iteration i and store the value for the voting function. If we then attempt to merge the same two entities at the iteration i', the coreference scoring functions will be different, since they assess the whole partition. This means that this speed-up trick only works if the ensemble voting function is very simple and is not affected by slight changes in the individual coreference scores. The second consideration is more troublesome. Hashing of voting results only works if the underlying coreference scoring function respects certain monotonicity properties: suppose a (candidate) merge of two entities, e_1 and e_2 at iteration i improves the coreference score with respect to a component's output; the same merge should improve the coreference score also at any later iteration i'. Intuitively speaking, this means that two entities should or shouldn't be merged, according to a specific coreference metric, regardless of the rest of the partition. While link- or mention-based metrics respect this property, the CEAF scores evaluate partitions as a whole and therefore are not monotonic.

Finally, some coreference metrics, such as B^3 and, most importantly, MUC are very fast to compute. The CEAF scores, on the contrary, require a computationally expensive partition alignment procedure. A considerable speed-up can be achieved by opting for a faster scorer. In the experimental section, we evaluate the algorithm's performance with different scoring metrics.

3.2 Algorithm adjustments for the CoNLL/OntoNotes setting

Following the state of the art, we evaluate our approach within the CoNLL framework (Pradhan et al., 2012): we use the OntoNotes dataset (Hovy et al., 2006) and rely on the official release (v8) of the scorer (Pradhan et al., 2014). Several important adjustments should be made to our algorithm to account for peculiarities of this set-up. In particular, (a) the OntoNotes guidelines do not provide annotations for singleton entities and (b) the official shared task score (MELA) relies strongly on B^3 and CEAF metrics. These two properties in combination lead to a number of counter-intuitive effects. We refer the reader to a recent study by Moosavi and Strube (2016) for an extensive discussion of problematic issues with the CoNLL scoring strategy.

The following adjustments have been made to run the algorithm in the CoNLL setting. First, each mention has been duplicated to mitigate the mention identification effect (Moosavi and Strube, 2016): we expand each document by several lines and fill them with dummy mentions. This prevents the system from making spurious merges at the initial iterations as a result of problematic CEAF values.

Second, we employ several clean-up strategies to post-process the final partition. Thus, we remove mentions recognized by a single system only, unless they are considered coreferent with exactly one popular (recognized by multiple systems) mention. This rather inelegant solution could be replaced with a simple requirement that each mention should be recognized by several systems if the singletons were not removed from the evaluation.

4 Experiments

In this section, we evaluate empirically the performance of the collaborative partitioning approach for a variety of ensembles. In particular, we investigate ensembles of different size and composition with respect to the components' quality and assess different coreference scoring metrics as criteria for partition similarity.

4.1 Experimental setup

In our experiments, we rely on the English portion of the CoNLL-2012 dataset (Pradhan et al., 2012). We use the outputs of the CoNLL submissions on the test data, made available publicly by the organizers.

To speed up the system, we use the techniques discussed in Section 3.1 above. In particular, we rely on a very simple unweighted voting scheme: each component contributes equally to the final score. The per-component score for a candidate merge between e_1 and e_2 is computed as follows: if either e_1 or e_2 are not represented in a component's output, it abstains from voting ($score = 0$). Otherwise, the component upvotes candidate merges if the underlying coreference score increases ($score = 2$) and downvotes, if it decreases ($score = -1$). The preference for positive votes (2 vs. 1) is motivated by the fact, that most state-of-the-art models explicitly model coreference, but not non-coreference: if two entities are annotated as non-coreferent by the system, it can be due to several factors, such as the lack of relevant features or algorithm peculiarities that limit the search space. The positive information in the systems' output is therefore more reliable than the negative one. The specific threshold ($2 : 1$) has been chosen arbitrary without any tuning. Finally, the termination threshold has been set to 0.

4.2 Choosing the scoring metric

In our first experiment, we evaluate different ways of defining similarity between partitions. Recall that each merge is evaluated based on whether it makes the constructed partition closer to the outputs of individual components. The similarity between two partitions is assessed with a task-specific measure. Multiple metrics have been proposed to evaluate coreference resolvers, we refer the reader to (Luo and Pradhan, 2016) for a detailed description and to (Moosavi and Strube, 2016) for a discussion of their problematic properties. In the present experiment, we assess three commonly accepted metrics, MUC, B^3 and CEAFE as well as their average, MELA, used for the official ranking of the CoNLL shared task.

Table 3 summarizes the results achieved by ensembles of the top-3 CoNLL systems. The upper half of the table presents individual components, re-evaluated with the v8 scorer. The lower part presents the performance achieved by four different ensembles, varying the underlying similarity measure used for growing up the partitions. For each performance metric, we highlight the best approach with boldface.

This experiment suggests several findings. First, the collaborative partitioning clearly brings a considerable improvement: depending on the underlying similarity score, the ensemble performs up to 3.5 percentage points better than the best individual components. Moreover, all the four created ensembles yield scores comparable to the very best state-of-the-art systems.

Second, all the four ensembles outperform individual components according to all the evaluation metrics. This means that the overall improvement (MELA) reflects a real quality increase and not just some fortunate re-shuffling of the individual scores to be averaged.

Third, the best overall improvement is achieved with the voting function based on the MELA similarity. The much faster MUC-based method performs 1.5 percentage points worse. This is an ambiguous result: on the one hand, a difference of 1.5% on the CoNLL dataset is non-negligible. On the other hand, even the MUC-based method outperforms each individual component.

4.3 Ensembles of top vs. bottom CoNLL systems

The performance of different systems submitted to CoNLL varies considerably, from 36.11 to 60.64 (MELA score, v08). In this experiment, we try to combine different types of systems. We split all the CoNLL systems into "tiers" of 3 submissions,

components	MUC F	CEAFE F	B^3 F	MELA
CoNLL system outputs				
fernandes	70.51	53.86	57.58	60.64
martschat	66.97	51.46	54.62	57.67
bjorkelund	67.58	50.21	54.47	57.41
Per-tier ensembles (3 systems per ensemble), score>0				
fernandes, martschat,bjorkelund; MUC similarity	**72.45**	55.71	59.87	62.67
fernandes, martschat,bjorkelund; CEAFE similarity	71.73	58.04	61.00	63.58
fernandes, martschat,bjorkelund; B^3 similarity	71.75	58.31	61.08	63.70
fernandes, martschat,bjorkelund; MELA similarity	71.96	**58.95**	**61.35**	**64.08**

Table 3: Collaborative partitioning with the 3 top CoNLL-2012 systems, using different coreference metrics when assessing candidate merges. Boldface indicates the best performing system for each score.

components	MUC F	CEAFE F	B^3 F	MELA
CoNLL system outputs				
tier1: fernandes, martschat,bjorkelund	70.51 66.97 67.58	53.86 51.46 50.21	57.58 54.62 54.47	60.65 57.68 57.42
tier2: chang,chen,chunyang	**66.38** 63.71 63.82	48.94 48.10 47.58	52.99 51.76 51.21	56.10 54.52 54.20
tier3: stamborg,yuan,xu	64.26 62.55 66.18	46.60 45.99 41.25	51.66 50.11 50.30	54.17 52.88 52.57
tier4: shou,uryupina,songyang	62.91 60.89 59.83	46.66 42.93 42.36	49.44 46.24 45.90	53.00 50.02 49.36
tier5: zhekova,xinxin,li	53.52 48.27 50.84	32.16 31.90 25.21	35.66 35.73 32.29	40.44 38.63 36.11
Per-tier ensembles (3 systems per ensemble)				
tier1: fernandes, martschat,bjorkelund	**71.96**	**58.95**	**61.35**	**64.08**
tier2: chang,chen,chunyang	66.35	**53.54**	**56.11**	**58.66**
tier3: stamborg,yuan,xu	**68.60**	**52.98**	**57.89**	**59.22**
tier4: shou,uryupina,songyang	**66.75**	**51.25**	**55.10**	**57.70**
tier5: zhekova,xinxin,li	**56.18**	**34.67**	**41.51**	**44.12**

Table 4: Ensembles of 3 classifiers for different tiers, using MELA for merging. Boldface indicates the best performing system for each tier.

components	tier MUC (R)	tier MUC (P)	tier MUC (F)	tier MELA
CoNLL system outputs				
tier1: fernandes,martschat,bjorkelund	65.83 65.21 65.23	**75.91** 68.83 70.10	70.51 66.97 67.58	60.64 57.67 57.41
tier2: chang,chen,chunyang	64.77 63.47 64.08	**68.06** 63.96 63.57	66.38 63.71 63.82	**56.10** 54.51 54.20
tier3: stamborg,yuan,xu	65.41 62.08 59.11	63.15 63.02 **75.18**	64.26 62.55 66.18	54.17 52.87 52.57
tier4: shou,uryupina,songyang	63.45 61.00 55.29	62.38 60.78 65.19	62.91 60.89 59.83	**53.00** 50.01 49.35
tier5: zhekova,xinxin,li	54.28 55.48 39.12	52.79 42.72 **72.57**	53.52 48.27 50.84	40.44 38.62 36.11
Per-tier ensembles (3 systems per ensemble)				
tier1: fernandes,martschat,bjorkelund	**69.60**	75.55	**72.45**	**62.67**
tier2: chang,chen,chunyang	**69.26**	64.61	**66.85**	54.63
tier3: stamborg,yuan,xu	**67.48**	69.12	**68.29**	**54.26**
tier4: shou,uryupina,songyang	**69.26**	**66.07**	**67.63**	52.23
tier5: zhekova,xinxin,li	**57.07**	61.77	**59.33**	**40.85**

Table 5: Ensembles of 3 classifiers for different tiers, using MUC for merging. Boldface indicates the best performing system for each tier.

based on their ranking. We do not use the system scores; however, we rely on the ranking computed on the same dataset.[3]

Tables 4 and 5 report the performance figures for ensembles composed of systems from each tier. The former uses MELA as a similarity measure, the latter—MUC. In both tables, the upper half reports performance figures for individual components (each cell in the upper half contains three values for the performance of the three systems of each tier). The lower half reports performance figures for collaborative partitioning with the components from each tier. The best-performing system for each performance metric is shown in boldface: for example, the MUC-based ensemble of the three tier1 systems outperforms its individual components in MUC Recall, MUC F and MELA (Table 5, lower half, first row), with the scores of 69.6%, 72.45% and 62.67% respectively; the best MUC Precision for tier1 (75.91%) is, however, achieved by an individual component, the system `fernandes` (Table 5, upper half, first row).

As these two tables suggest, collaborative partitioning yields improvement over individual components, for both stronger and weaker tiers. This suggests that collaborative partitioning can be used on top of any systems: unlike many other ensemble techniques, it does not suffer from the error propagation problem when operating on ensembles of weaker components.

The final partition depends on the similarity measure used by the collaborative algorithm. Thus, the MELA measure, being an average of

[3] This is a rather unfortunate set-up, but there are no means to roughly evaluate CoNLL systems without using the test data. We assume, however, that an external evaluation, if possible, would be able to differentiate top against bottom submissions.

components	MUC R	MUC P	MUC F	MELA
best individual component (`fernandes`)				
fernandes	65.83	75.91	70.51	60.65
ensembles, default termination threshold ($= 0$)				
tier1	69.60	75.55	72.45	62.67
tier1+2	74.85	61.73	67.66	52.38
tier1+2+3	75.78	56.43	64.69	43.97
tier1+2+3+4	74.85	53.34	62.29	39.58
tier1+2+3+4+5 (all)	74.08	48.02	58.27	33.10
ensembles, optimal termination threshold				
tier1	69.60	75.55	72.45	62.67
tier1+2	70.53	75.93	73.13	53.60
tier1+2+3	71.54	75.24	73.35	44.41
tier1+2+3+4	68.50	77.12	72.55	49.06
tier1+2+3+4+5 (all)	65.36	80.24	72.04	45.78

Table 6: Ensembles of different sizes, using MUC for merging.

	MUC	CEAFE	B^3	MELA
competitive upper bound, tier1	71.53	56.46	59.74	62.57
competitive upper bound, tier1+2	71.53	56.46	59.74	62.57
competitive upper bound , all	**72.12**	57.55	60.53	63.39
collaborative, tier1	71.96	**58.95**	**61.35**	**64.08**

Table 7: Competitive vs. collaborative partitioning, using MELA for selection (competitive) or merging (collaborative).

MUC, B^3 and CEAF, leads to more balanced final partitions, improving on each individual score. MUC-based ensembles, on the contrary, improve on MUC (through a drastic increase in MUC recall without much precision loss), but do not guarantee any increase in B^3 or CEAF, leading to mixed results on MELA.

4.4 Ensembles of different size

In this experiment, we consider ensembles of different sizes, starting from tier1 and adding less performing components. Table 6 reports the results for ensembles of different size, using MUC for measuring the similarity while growing partitions. The upper half presents the results with the default termination parameter. As it shows, the inclusion of more lower-quality systems leads to better MUC recall values at the cost of the sharp deterioration in precision and the overall scores.

The lower half shows the results obtained with the optimal value of the termination parameter. In a practical scenario, this parameter can be tuned on the development data. Here, the best MUC results ($F = 73.35$) are achieved with the top nine systems. However, this MUC improvement comes at a high cost in B^3 and CEAF, leading to low MELA values even with the optimal parametrization.

4.5 Collaborative vs. Competitive Partitioning

One of the key advantages of the collaborative partitioning algorithm is its loose coupling approach

components	MUC	CEAFE	B^3	MELA
berkeleycoref	69.13	54.30	57.40	60.27
ims-hotcoref	70.25	55.44	58.03	61.23
LSPE	**72.34**	57.40	60.36	63.36
ensemble	71.98	**60.01**	**61.44**	**64.47**

Table 8: Collaborative partitioning for state-of-the-art systems, using MELA for merging. Boldface indicates the best result for each score.

with respect to individual components. This allows for straightforward integration of any coreference resolver at the moment of its release. The only other approach with the same property has been advocated by Ng (2005), where a ranker is learned to select the best partition from the individual outputs. We refer to this algorithm as *competitive partitioning*, since individual components compete with each other for each document instead of collaborating to build a new improved partition.

The competitive partitioning algorithm has a natural upper bound: by using an oracle to always select the best-performing component for each individual document, we can get the highest performance level possibly attainable with this model. Table 7 shows these upper bounds for the first 3, 6 and 15 (all) CoNLL systems. Note that these numbers are obtained with an oracle—the results with a real ranker will, obviously, be lower. The last row of the table shows, for comparison, the tier1 performance for the collaborative partitioning algorithm.

First, it is clear that competitive partitioning on top of CoNLL systems is hardly promising: even in the oracle setting, the performance improves by only 2-3 percentage points. This is due to the fact that CoNLL has a clear winner, the system `fernandes`, yielding the best solution for more than half of the documents and never losing too much for the remaining half.

Second, collaborative partitioning, on the contrary, seems more beneficial, yielding the results superior to the upper bound of the competitive partitioning algorithm. This is due to the fact that the collaborative approach makes a better use of individual components, combining their entities to arrive at a better new solution.

4.6 Ensembles of post-CoNLL systems

In our last experiment, we depart from the CoNLL outputs to run the collaborative partitioning algorithm on top of the state-of-the-art coreference resolvers. In particular, we combine three very different high-performing systems, berkeleycoref (Durrett et al., 2013), ims-hotcoref (Björkelund and Kuhn, 2014) and lspe (Haponchyk and Moschitti, 2017b; Haponchyk and Moschitti, 2017a). The former relies on an entity-level modeling, whereas the latter two use different structural learning approaches to coreference. All these systems represent state-of-the-art research in the field. Note that we do not include the very latest deep learning based approaches (Wiseman et al., 2016; Clark and Manning, 2016) to allow for a fair comparison: since, as we have seen in the experiments above, the collaborative partitioning algorithm consistently improves over individual ensemble components, integrating the very best systems would be a trivial but not very informative way of advancing the state of the art.

Table 8 shows the performance level of each of these systems on the English portion of the CoNLL-2012 dataset, individually and of the collaborative ensemble. The best performing system according to each metric is shown in bold. The numbers were obtained by running the v08 scorer on the outputs provided by the developers (berkeleycoref, lspe) or created using the official distribution and the provided pre-trained model (ims-hotcoref). No adjustments have been made to the collaborative partitioning algorithm.

Similarly to the experimental findings presented in the previous sections, the collaborative partitioning algorithm outperforms the best individual components. Most importantly, it yields the second-best results reported in the literature, outperforming the system of Wiseman et al. (2016) by 0.26 percentage points.

5 Conclusion

This paper presents collaborative partitioning—a novel ensemble-based approach to coreference resolution. Starting from the all-singleton solution, we search the space of all partitions, aiming at finding the solution close to the components' partitions according to a coreference-specific metric. Our algorithm assumes a loose coupling of individual components within the ensemble, allowing for a straightforward integration of any third party coreference resolution system.

Our evaluation experiments on the CoNLL dataset show that the collaborative partitioning method improves upon individual components, both for high and low performing ensembles. This performance improvement is consistent across all the metrics. Moreover, when combining three state-of-the-art systems, the collaborative ensemble achieves the second-best results reported in the literature so far (MELA score of 64.47).

In the future, we plan to concentrate on improving the voting scheme for the ensemble. Currently, the model relies on a very simplistic unweighted voting strategy. This choice is motivated by practical considerations: a more complex scheme would not make possible the necessary system speed up techniques. The unweighted voting, however, is problematic for ensembles that (a) contain components of very different quality or (b) contain some extremely similar components. This issue has been investigated within the ensemble classification framework, where several approaches have been put forward to construct large ensembles that ensure diversity of their components, e.g., through splitting training data and/or feature sets. In our scenario, however, we can not rely on such techniques, since we build ensembles of few existing high-quality systems, each of them being an outcome of a considerable research and engineering effort. We plan to overcome these issues, investigating different versions of heterogeneous voting.

Another direction of our future work involves an extensive comparison of our approach with ensemble clustering algorithms proposed within the machine learning and data mining community, in particular, by Strehl and Ghosh (2003). Thus, we plan to (i) evaluate our model against these general-purpose techniques in terms of both accuracy and efficiency and (ii) investigate possibilities of adapting the existing ensemble clustering algorithms to explicitly incorporate task-specific metrics.

Finally, we plan to extend our approach to other NLP tasks, investigating collaborative ensembles for other problems with complex outputs, going beyond simple classification-based ensemble techniques.

Acknowledgments

This work has been partially supported by the EC project CogNet, 671625 (H2020-ICT-2014-2, Research and Innovation action).

References

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 47–57, Baltimore, Maryland, June. Association for Computational Linguistics.

Kevin Clark and Christopher D. Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653, Berlin, Germany, August. Association for Computational Linguistics.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassell, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program–tasks, data, and evaluation. In *Proceedings of the Language Resources and Evaluation Conference*.

Greg Durrett, David Hall, and Dan Klein. 2013. Decentralized entity-level modeling for coreference resolution. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 114–124, Sofia, Bulgaria, August. Association for Computational Linguistics.

Iryna Haponchyk and Alessandro Moschitti. 2017a. Dont understand a measure? Learn it: Structured prediction for coreference resolution optimizing its measures. In *Proceedings of the 55th Annual Conference of the Association for Computational Linguistics (ACL)*, Vancouver, Canada, July. Association for Computational Linguistics.

Iryna Haponchyk and Alessandro Moschitti. 2017b. A practical perspective on latent structured prediction for coreference resolution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 143–149, Valencia, Spain, April. Association for Computational Linguistics.

Lynette Hirschman and Nancy Chinchor. 1997. MUC-7 coreference task definition. In *Message Understanding Conference Proceedings*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

Beata Kouchnir. 2004. A machine learning approach to german pronoun resolution. In *Proceedings of the ACL 2004 Workshop on Student Research*, ACLstudent '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiaoqiang Luo and Sameer Pradhan. 2016. Evaluation metrics. In *Anaphora Resolution*, pages 141–163. Springer Berlin Heidelberg.

Nafise Sadat Moosavi and Michael Strube. 2016. Which coreference evaluation metric do you trust? a proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 632–642.

Art Munson, Claire Cardie, and Rich Caruana. 2005. Optimizing to arbitrary NLP metrics using ensemble selection. In *Proceedings of HLT/EMNLP*, pages 539–546.

Vincent Ng and Claire Cardie. 2003a. Bootstrapping coreference classifiers with multiple machine learning algorithms. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 113–120. Association for Computational Linguistics.

Vincent Ng and Claire Cardie. 2003b. Weakly supervised natural language learning without redundant views. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics - Volume 1*, pages 173–180. Association for Computational Linguistics.

Vincent Ng. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 157–164.

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.

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'12)*, Jeju, Korea.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard H Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *ACL (2)*, pages 30–35.

Altaf Rahman and Vincent Ng. 2011. Ensemble-based coreference resolution. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1994–1889.

Sriparna Saha, Asif Ekbal, Olga Uryupina, and Massimo Poesio. 2011. Single and multi-objective optimization for feature selection in anaphora resolution. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP'11)*.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistic*, 27(4):521–544.

Alexander Strehl and Joydeep Ghosh. 2003. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, March.

Smita Vemulapalli, Xiaoqiang Luo, John F. Pitrelli, and Imed Zitouni. 2009. Classifier combination techniques applied to coreference resolution. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*, SRWS '09, pages 1–6, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 994–1004. The Association for Computational Linguistics.