A State-of-the-Art Mention-Pair Model for Coreference Resolution

Olga Uryupina[1] and **Alessandro Moschitti**[2,1]
[1]Department of Information Engineering and Computer Science, University of Trento,
[2]Qatar Computing Research Institute
uryupina@gmail.com, amoschitti@gmail.com

Abstract

Most recent studies on coreference resolution advocate accurate yet relatively complex models, relying on, for example, entity-mention or graph-based representations. As it has been convincingly demonstrated at the recent CoNLL 2012 shared task, such algorithms considerably outperform popular basic approaches, in particular mention-pair models. This study advocates a novel approach that keeps the simplicity of a mention-pair framework, while showing state-of-the-art results. Apart from being very efficient and straightforward to implement, our model facilitates experimental work on the pairwise classifier, in particular on feature engineering. The proposed model achieves the performance level of up to 61.82% (MELA F, v4 scorer) on the CoNLL test data, on par with complex state-of-the-art systems.

1 Introduction

The mention-pair model, as proposed by Soon et al. (2001) has been used for over a decade now. It combines a simple classifier trained to discriminate between coreferent and not-coreferent pairs of mentions ("links") with fast heuristic procedures for merging the classifier's decisions at the decoding stage. Several decoding heuristics have been advocated in the literature, the most commonly used ones including first-link (Soon et al., 2001) and best-link (Ng and Cardie, 2002).

Most state-of-the-art algorithms for coreference resolution, on the contrary, rely on complex modeling, ranging from entity-ranking to structural perceptron and other graph-based approaches (for an overview of state-of-the-art coreference resolvers, see (Ng, 2010; Pradhan et al., 2012)). Such algorithms show a clearly superior performance: thus, at the CoNLL-2012 shared task, the best-performing (Soon et al., 2001)-style system loses around 8% to the winning algorithm.

However, more traditional mention-pair approaches still have some important advantages. Thus, a mention-pair model is easy to implement and allows for fast prototyping. It relies on a simple binary classifier making it very fast to train compared to state-of-the-art models that are based on complex structural representations (Fernandes et al., 2012; Björkelund and Kuhn, 2014). This efficiency at the training step allows for straightforward automatic parameter optimization. Most importantly, mention-pair models can be useful for understanding low-level system behavior, and, in particular, for feature engineering. This can in turn help improve more complex models, since many of them rely on mention-pairs as their basic building blocks.

In this paper, we advocate a new easy-first mention-pair algorithm (EFMP): while it is based solely on pairs of mentions and does not attempt any global inference, it benefits from the decision propagation strategy to create a coreference partition. Augmented with the sieve-style prefiltering, the system achieves a performance level comparable to the state of the art.

The contribution of this paper is two-fold. First, we propose a novel decoding approach that combines predictions of the mention-pair classifier based on its confidence score, taking into account—in contrast to the previous studies, e.g. (Ng and Cardie, 2002; Stoyanov and Eisner, 2012; Björkelund and Kuhn, 2014)—both positive and negative links. We thus propose a procedure for propagating positive and negative links to create the final coreference partition: we start from the most confident among all the classifier's decisions and iteratively construct coreference partitions by merging coreference chains (positive links) or blacklisting future merges (negative links). This decoding strategy is slower than the commonly used best-link model, but considerably faster than ILP-based decoding (Finkel and Manning, 2008; Denis and Baldridge, 2009).

Second, we show that our approach, being very fast and easy to implement, can be used for a variety of low-level experiments on coreference resolution, in particular, for studies on feature engineering or selection. Thus, we augment our system with two feature combination techniques, Jaccard Item Mining (Segond and Borgelt, 2011) and Entropy Guided Feature Inductions (Fernandes et al., 2012). While the latter has been used for coreference resolution before, Jaccard Item Mining (JIM), to our knowledge, has never been applied to any NLP task. The JIM algorithm has been developed within the data mining community and aims at finding combinations that tend to occur in a particular set of unlabeled transactions. In this paper, we introduce a post-filtering technique to re-score JIM output w.r.t. the class labels (±coreferent). We show empirically that JIM is more suitable for coreference: it provides smaller and more meaningful feature combinations leading to a better performance level.

The combination of our decoding approach with the JIM feature induction technique allows us to achieve a performance level of 61.82% on the CoNLL-2012 test data, just 1.5% percent below the (much more complex) winning system and above all the other submissions (cf. Table 4).

2 Related work

An improvement over the original mention-pair model (Soon et al., 2001) has been proposed by Ng and Cardie (2002). Their "best-link" algorithm picks the most confident antecedent for each anaphor. Unlike Ng and Cardie (2002), we do not process the input text from left to right incrementally, instead, we assess the confidence of all the proposed links at the same time ("easy-first") and keep track of negative assignments.

Our work has been motivated by more complex algorithms using the easy-first strategy, most importantly, by Stoyanov and Eisner (2012), Nicolae and Nicolae (2006) and Björkelund and Farkas (2012). There are two important differences between these studies and the Easy-First Mention-Pair model (EFMP): (i) EFMP does not evaluate links between entities or clusters, always operating on mention pairs instead; (ii) EFMP integrates both positive and negative assignments in its hierarchy of easy-to-hard decisions.

Being conceptually very simple, our algorithm allows for a straightforward integration of other techniques proposed in the literature, in particular, sieve-style prefiltering (Lee et al., 2011) and feature induction. Several recent studies have attempted exhaustive analysis of features and their impact on the overall performance (Recasens and Hovy, 2009; Uryupina, 2006; Bengtson and Roth, 2008; Durrett and Klein, 2013). We refer the reader to (Ng, 2010) for an overview of different features. Kobdani et al. (2010) create a framework that facilitates the engineering process for complex features. This approach, however, still relies on the human expertise for creating meaningful combinations. Versley et al. (2008) use kernel-based similarity as an implicit feature induction technique.

The only study we are aware of that investigates an explicit feature combination technique has been conducted by Fernandes et al. (2012). Their algorithm for Entropy-based feature induction (EFI), shows substantial improvement on the OntoNotes dataset. In the present work, we propose an alternative to EFI, based on the recent advances in Data Mining. We believe that Fernandes et al. (2012) have opened a very important research direction with their feature induction approach. We want therefore to evaluate EFI in a simpler and more straightforward mention-pair model—and compare it to our approach.

3 Easy-First Mention-Pair Model

In what follows, we describe our Easy-First Mention-Pair (EFMP) approach and then propose a solution for combining our model with manually engineered filters, inspired by Stanford "sieves" (Lee et al., 2011). EFMP is a decoding algorithm.

Algorithm 1 Easy-first decoding (EFMP)

```
Require: $L = \{< ana_l, ante_l, label_l, confidence_l >\}$: list
    of classified mention pairs
 1: sort $L$ according to $confidence$
 2: for all $l \in L$ do
 3:     /* don't override prev. decisions */
 4:     if $chain(ana_l) == chain(ante_l)$ then
 5:         continue
 6:     if $unlinked(chain(ana_l), chain(ante_l))$ then
 7:         continue
 8:     /* update chains and unlink info */
 9:     if $label_l$==not-coreferent then
10:         $unlink(chain(ana_l), chain(ante_l))$
11:     if $label_l$==coreferent then
12:         UpdateUnlinkInfo($\{chain(ana_l), chain(ante_l)$)
13:         MergeChains($\{chain(ana_l), chain(ante_l)$)
14: function UPDATEUNLINKINFO($chain_1, chain_2$)
15:     for all $c$ such as $unlinked(c, chain_2)$ do
16:         $unlink(c, chain_1)$
17: function MERGECHAINS($chain_1, chain_2$)
18:     for all $m \in chain_2$ do
19:         $chain(m) = chain_1$
```

At the encoding step, we generate mention-pairs in a straightforward exhaustive way: each candidate anaphor is paired with all the preceding candidate antecedents. Following the state-of-the-art, we filter out mention pairs using the same sieve-style approach at both the encoding and the decoding steps (cf. Section 3.2 below).

3.1 Plain EFMP

Our EFMP approach addresses the clustering step of a coreference resolution process: as its input, it assumes a set of mention pairs for a given document, labeled as positive (two mentions corefer) or negative (two mentions do not corefer) by an external classifier. We also assume the classifier to output the confidence of its decisions.

The key idea behind EFMP is the processing of all the decisions, both positive and negative ones, in a specific order, according to the classifier's confidence. We start by sorting all the mention pairs by the confidence of the assigned label. We instantiate our clustering assigning each mention to its own cluster ("all singletons"), however, we do not prohibit potential links between any of them. Our EFMP module processes all the (sorted) mention pairs one-by-one, at each step performing one of the following operations, whenever possible:

- **link**: merge two clusters (includes propagating unlink information, cf. below)
- **unlink**: mark two given clusters to prohibit potential merge at any future step

These operations, however, are only performed if the system has no information about the possibility of (un)linking the two mentions at the given step.

Let us illustrate the approach with the following example:

(1) [Alice]$_1$ is showing [Zoe]$_2$ [her]$_3$ papers on coreference.

In this snippet, we collect three mentions ("Alice" (M_1), "Zoe" (M_2) and "her" (M_3)), forming three mention pairs.[1] A state-of-the-art pairwise coreference classifier would confidently label $< Alice, Zoe >$ as negative; and less confidently—$< Alice, her >$ and $< Zoe, her >$ as positive. The score for $< Alice, her >$ would be slightly higher: "Alice" is a subject and a first mention in the sentence. The EFMP module starts from three clusters: $C_1 = \{M_1\}$, $C_2 = \{M_2\}$, $C_3 = \{M_3\}$. At the first step, it registers the information, that C_1 and C_2 should never be merged ("unlink"). At the second step, it links M_1 and M_3, merging C_1 and C_3, thus producing a partition with two clusters: $C'_1 = \{M_1, M_3\}$, $C_2 = \{M_2\}$. It also propagates the previously collected unlinking information, registering the fact that C'_1 and C_2 should never be merged. At the next step, it tries to link M_2 and M_3. This, however, doesn't work, since the system has already collected some more reliable evidence that the corresponding clusters shouldn't be merged. As there are no more mention-pairs left, the system stops and outputs the last partition (C'_1, C_2).

3.2 EFMP with Sieves

The plain EFMP approach, as described above, assumes that all the mention pairs are labeled by the pairwise classifier. This has several potential issues. First, it requires sorting all the links: when mention-pairs are generated exhaustively, it amounts to the running time of $O(n^2 \log(n))$, where n is the total number of mentions. Second, the pairwise classifier has to be trained on a very biased dataset, containing

[1]We omit other mentions ("her papers" and "coreference") to simplify the presentation.

too many negative or irrelevant examples; this might decrease the performance and also requires substantial tuning of learning parameters.

To alleviate the problem, we pre-filter mention pairs aggressively, heuristically eliminating links that are either definitely positive (for example, pairs of same named entities), definitely negative (pairs with incompatible gender values) or "uninformative". The latter are pairs that we cannot realistically expect to be analyzed by our system, due to the limits of our feature representation. For example, a pair of two noun phrases sharing no common tokens and appearing far apart from each other might be either positive or negative, with the particular decision depending on a lot of factors, starting from the semantic compatibility of the two mentions ("car" and "relativity" can hardly be coreferent), but also including discourse-related factors and some suitably represented knowledge of other entities ("car" and "Ferrari" in different parts of a document talking about Formula 1 may refer to different entities). We believe that forcing our pairwise classifier to learn a labeling for such "uninformative" examples without providing adequate features might lead to inferior performance.

Our pre-filtering approach was inspired by the Stanford sieves algorithm (Lee et al., 2011), where several high-precision rules are applied in a specific order to filter out candidates. This approach has since then been used in several systems, most successfully by Fernandes et al. (2012) to filter out training data for coreference resolution classifiers. The idea of distinguishing between "informative" and "uninformative" instances has been implicitly adopted by many systems, restricting their search to a specific window. This approach is very common for pronominal anaphora, but it's also used by several general-purpose coreference resolvers (Fernandes et al., 2012; Stoyanov and Eisner, 2012).

All these rule-based decisions can be integrated into EFMP in a straightforward way. Thus, the uninformative pairs are simply excluded from the further processing. They do not produce training material and they are not processed by EFMP at the test time (consequently, mentions from such a pair may end up in the same cluster, as well as in two different ones, depending on other (un)links established by the system). This allows for a substantial reduction of the pairs to be processed at the decoding step (for example, in our setting described in Section 5.1 below, around 90% of all the pairs are eliminated as "uninformative"). The pairs, deemed positive or negative by the rule-based pre-filtering, do not contribute to the training data. At the test step, they are considered to be very confidently positive/negative instances, outscoring any test pairs, originating from the classifier output. Such pairs do not contribute to speeding up the EFMP part, however, they help improve the quality of our pairwise classifier, decreasing the bias towards negative instances in the data.

4 Techniques for generating feature combinations

Most state-of-the-art coreference resolution systems combine complex modeling with rich feature sets. While early data-driven approaches were essentially knowledge-poor (for example, the famous system of Soon et al. (2001) is based on 12 shallow features), modern algorithms rely on dozens of carefully engineered features, encoding various clues relevant for the task: from different measures of surface similarity, to morphological, syntactic, semantic and discourse properties, and world knowledge.

This study focuses on the automatic feature engineering task. We start from atomic features that are already encoded in a state-of-the-art toolkit (BART) and use a data mining technique, Jaccard item mining, to boost the system performance through automatic induction of complex features.

The features used by most coreference resolution systems are very heterogeneous. Some of them (for example, different measures of salience) encode insights from the linguistic theory, whereas others are purely data-driven (tokens). Some features are direct indicators for or against coreference (string matching vs. contra-indexing constraints), whereas others are supposed to provide more general information (individual properties of mentions). Some features are very frequent (mention types), whereas others are relatively rare (apposition). Finally, some features encode basic properties ("anaphor is a pronoun"), whereas others are combinations of such properties ("both anaphor and antecedent are pronouns and they have the same surface form"). Therefore, we specifically aim at designing an algorithm that is able to overcome these idiosyncrasies and provide meaningful combinations for such a heterogeneous set of atomic features.

4.1 Jaccard Item Mining

In this study, we adapt the Jaccard Item Mining (JIM) technique (Segond and Borgelt, 2011) to the coreference resolution task. Below we describe the JIM algorithm and our adjustment of JIM to the task of selecting meaningful features.

The Data Mining community has invested substantial efforts into Frequent Item Mining algorithms: techniques for finding frequent combinations of "items" in a "transaction database". We assume that a database is a set of item sets with some items often appearing together. Several approaches have been proposed to solve the task of enumerating all such frequently occurring combinations in a fast and efficient way, the most popular ones being Eclat and FP-growth. A typical application would be, for example, the task of finding similarities in shopping lists for different customers. We refer the reader to (Borgelt, 2012) for an overview of relevant approaches.

Frequent Item Mining algorithms output all the combinations with a frequency ("support") higher than a predefined threshold. If the original items are very heterogeneous, the output might get very noisy: for example, if there are some very frequent items, they will pollute most combinations and the interesting item sets will be difficult to find. To overcome this problem, Segond and Borgelt (2011) propose to use the Jaccard index as a measure of the item set quality. For a given set of items I, the Jaccard index $J_T(I)$ is defined as a ratio of the set's support over the number of transactions containing at least one item from the set:

$$J_T(I) = \frac{|\cap_{i \in I} K_T(\{i\})|}{|\cup_{i \in I} K_T(\{i\})|},$$

where $K_T(\{i\})$ is a set of transactions, containing the item i.

It is straightforward to see that $J_T(I)$ is an anti-monotone function. Therefore, standard frequent item mining algorithms can be easily adapted to cover the Jaccard index. In our experiments, we use a publicly available JIM implementation.[2]

[2]http://www.borgelt.net/jim.html

We recast the feature induction problem as a frequent item mining task. We start from atomic features (string, nominal and binary) and convert them into binary features that represent our items. Since our feature set is very heterogeneous, some items are very rare or, conversely, very frequent. We filter out all the items with the support below or above predefined thresholds. The frequent item mining approaches assume that all the transactions are equal. In our case, however, transactions correspond to training instances—and they come with the class labels. Since we are interested in the feature combinations that help distinguish between positive and negative examples, we perform two JIM runs, splitting the training data into the positive and negative parts. For each part, we induce all the combinations with the high Jaccard index ($J^+_T(I)$ and $J^-_T(I)$). After this step, we have two lists of items, each corresponding to feature combinations showing a good association strength for positive and negative examples respectively.

Both lists are then reranked, dividing the positive index over the negative one and vice versa:

$$score^+(I) = \frac{J^+_T(I)}{J^-_T(I)}, score^-(I) = \frac{J^-_T(I)}{J^+_T(I)}.$$

This reranking step helps us to filter out feature combinations that are either redundant or not indicative of coreference. For example, our atomic features already contain some combinations (e.g., `NEStringMatch` is a combination of `MentionType_Coarse` and `StringED`). Without the reranking step, we are getting numerous combinations reflecting peculiarities in the feature design. As the final JIM output, we take all the sets I with the scores exceeding some predefined thresholds ($score^+(I) > thr^+$ or $score^-(I) > thr^-$). Note that our $score$ measures are not monotone and cannot therefore be used in a fast Eclat-style algorithm (Borgelt, 2012) to directly provide $score$-optimal combinations.

To better align our approach with the Entropy Guided Feature Induction framework presented below, we convert our item sets back to the sets of atomic features, abstracting away from the particular values used for binarization.

The JIM-based feature induction algorithm relies on several parameters: the feature filtering thresholds (for removing too rare or too common features), the minimal Jaccard index for the JIM algorithm and the thresholds thr^+ and thr^- for selecting good item sets after the reranking step. We fit these parameters on the development set.

4.2 Entropy Guided Feature Induction

Fernandes et al. (2012) have proposed using Entropy Guided Feature Induction (EFI) for coreference resolution and have shown that it significantly improves the performance of their system. Below we provide a brief overview of the EFI approach, referring the reader to the original paper for further details.

The system works at two stages, using two different machine learning techniques. At the *first stage*, the EFI algorithm relies on a decision tree, generated from the training data, to obtain meaningful feature combinations. In particular, the algorithm extracts all the paths leading from the root of the induced tree to any node. Each node in a tree corresponds to a specific value assigned to some atomic feature and therefore each path corresponds to a conjunction of atomic features with assigned values. Fernandes et al. (2012) abstract over the values, thus, converting each path to a conjunction of atomic features. These conjunctions, or combinations, are then used to generate numerous binary features to be used by a linear classifier at the *second stage*.

Since the induced tree might get very large, the EFI algorithm might lead to conjunctions of too many atomic features, generating, in turn, too many binary features. To address the issue, Fernandes et al. (2012) prune their tree at the depth 5. In our implementation, we follow the algorithm of Fernandes et al. (2012) with no adjustments or alterations.

5 Experiments

Our first group of experiments assesses the quality of the baseline setting, with no feature combination techniques. We compare against the CoNLL submission of the BART group to make sure that our (Soon et al., 2001)-style mention-pair baseline shows an acceptable performance. We then evaluate the EFMP approach to confirm that it provides much higher performance figures and is on par with the state of the art. In our second experiment, we use EFMP to assess the impact of the feature combination techniques on the performance of a coreference resolution system.

5.1 Experimental Setup

We evaluate our approach on the English portion of the CoNLL-2012 dataset (Pradhan et al., 2012). To asses the system's performance, we use the official scorer, provided by the CoNLL organizers. However, the version used at the competition time (v4) was later found to contain errors and replaced with another implementation (v7). This procedure resulted in a performance drop for all the systems, but didn't affect their ranking. To facilitate comparison against previous and future studies, we report both v4 and v7 MELA scores. All the experiments are performed on automatically extracted mentions and use no gold information.

For our study, we use the publicly available BART toolkit (Uryupina et al., 2012). We have made several adjustments, starting from the configuration, suggested in the BART distribution for the OntoNotes/CoNLL data. Thus, we have modified the mention detection module, improving the treatment of coordinations and eliminating numeric named entities (PERCENT, MONEY etc). We have replaced the original `split` architecture with a single-classifier approach to be able to estimate the impact of our feature combination techniques in a more principled way. We have also replaced Decision Trees (Weka J48) with the LibLinear SVM package, to get a classifier outputting reliable confidence values, as needed by EFMP. We have considerably expanded the feature set, mainly reimplementing features from the winning system of CoNLL-2012 (Fernandes et al., 2012). Altogether, we have around 170 individual features (string, nominal or binary values), corresponding to around 20k features after the binarization step. The full list of our feature templates can be found at `http://bart-coref.eu/papers/sem15-suppl.pdf`.

Finally, we have augmented BART with a rule-based prefiltering module, motivated by Stanford Sieves (Lee et al., 2011), the winning approach of the CoNLL-2011 shared task. Our sieve-style prefiltering algorithm splits all the training instances into confidently positive, confidently negative, irrelevant and relevant. To implement the prefiltering

	development	test	
	v4	v4	v7
BART CoNLL-2012 submission			
	-	56.12	50.02
simplified reimplemented BART submission			
WEKA (j48)	56.02	55.84	49.83
SVM (Liblinear)	48.31	47.29	39.17
-*-, all features			
WEKA (j48)	56.53	55.98	49.84
SVM (Liblinear)	49.83	48.11	40.04
best-link, all features			
SVM (Liblinear)	59.71	59.03	55.21
EFMP, features from BART submission			
SVM (Liblinear)	57.40	56.16	50.65
EFMP, all features			
SVM (Liblinear)	60.02	59.12	55.38

Table 2: Baseline performance vs. plain EFMP: MELA score, different versions of the CoNLL scorer.

module, we have started with the original sieves and the version used by Fernandes et al. (2012). We have changed some sieves and introduced several additional filters (cf. Table 1).

5.2 Baselines vs. EFMP

Table 2 shows the performance levels for different baseline algorithms, learners and features on both CoNLL-2012 development and test sets. Note that the development set was used for parameter tuning and does not therefore provide an accurate estimation of the system's performance.

The results suggest that our simplified version of the BART CONLL-2012 system can be considered an adequate starting point: it only shows a very minor performance drop, compared to the original submission (we believe that this drop can be attributed to the simpler `no-split` architecture that we are adopting in this study). The (Soon et al., 2001)-style mention-pair model, however, suffers from several problems. First of all, its performance is simply not good enough: thus, the winners of the CoNLL-2012 shared task reported a v4 score of 63.37 on the test data. With a v4 score of 55.84, our system would have achieved the 12th place in the competition (out of 15+1). Second, this approach only works with the decision tree-based classifier: with SVMs, the performance gets much lower. We believe that this can be caused by several factors: (a) decision trees perform some sort of feature combinations, whereas Liblinear only relies on a sum of individual features for its classification and (b) the (Soon et al., 2001)-style model employs different sampling strategies for training and testing data (in fact, testing instance are sampled dynamically, based on the decisions made by the classifier so far), leading to a misfit between the two sets that is more problematic for Liblinear. Third, even with the decision trees, the system performance does not improve substantially when we add a lot of manually engineered high-quality features.

The EFMP model, on the contrary, shows promising performance figures. With an F-score of 59.12, the system would have achieved the 8th place in the CoNLL-2012 competition, within the cluster of very similarly performing systems on places 2–8(9). It must be stressed that EFMP is a very simple and fast algorithm, much less complex than any of the high-performing CoNLL systems.

We have also evaluated EFMP against a mention-pair model with the same sieve-style prefiltering and a best-link decoder (Table 2, row 6). As the results suggest, the best-link decoder shows a better performance level compared to (Soon et al., 2001), since it relies on the most confident positive links. The EFMP decoder, however, brings a further improvement, by incorporating and propagating information on confident negative links as well.

5.3 Feature combinations

In our second experiment, we investigate the applicability of JIM to coreference resolution, comparing it against EFI. The latter has been proven to yield a performance gain of up to 10%, leading to a system, significantly outperforming all the other competitors at the CoNLL-2012 shared task. While the impact of EFI on the system of Fernandes et al. (2012) cannot be underestimated, the following points need further clarifications: (a) the algorithm of Fernandes et al. (2012) shows only very moderate performance without EFI—it is not yet clear if EFI is equally beneficial for more competitive approaches; and (b) the system of Fernandes et al. (2012) relies on a relatively complex model—it is not clear how model-specific the benefits of EFI are.

Confidently negative	
Expletive	M_i or M_j is an expletive pronoun
Span	one mention spans over the other
Agreement	M_i and M_j disagree in number, gender or semantic class
Syntax	M_i and M_j violate contra indexing constraints (c-command etc)
SpeakerAliasProFalse	heuristics for 1/2 person pronouns, based on the `speaker` value
Pronouns	M_i is a pronoun, M_i and M_j disagree in person (respecting the `speaker`)
Confidently positive	
SpeakerAliasPro	heuristics for 1/2 person pronouns, based on the `speaker` value
SpeakerAliasNE	heuristics for 1 person pronouns (M_j) and NE (M_i), based on the `speaker`
SameNE	M_i and M_j are exactly matching NEs
Irrelevant	
ProNonpro	M_j is a pronoun, M_i is not a pronoun
DistantPro	M_j is a pronoun, M_i is more than thr_1 sentences away ($dist(M_j, M_i) > thr_1$)
DistantNP	M_j is a common NP, $dist(M_j, M_i) > thr_2$, head nouns of M_i and M_j differ
DistantNE	M_j is an NE, $dist(M_j, M_i) > thr_2$, M_i and M_j do not match

Table 1: Sieves for pre-filtering of mention pairs: each sieve is applied to a pair of mentions $\{M_i, M_j\}$, $i < j$, where M_i is a candidate antecedent and M_j is a candidate anaphor.

EFI and JIM use very different intuitions for combining atomic features. It is therefore not surprising, that the outputs of these two algorithms are different. Figure 1 summarizes the distribution of EFI vs. JIM-induced combinations of different lengths, normalized by the total number of combinations extracted by each method. EFI outputs around 20 times more sets than JIM (2k vs. 90). Most of them, however, are too long and do not provide good features. By definition, EFI cannot produce a lot of short combinations, since all the EFI paths must start from the root. JIM, on the contrary, tends to produce combinations of smaller lengths that are more likely to yield high-quality features.

Table 3 shows the performance of EFMP, augmented with EFI or JIM-induced features. We see that both techniques bring an improvement over the plain EFMP (significant, per-document t-test, $p < 0.05$). Even though JIM produces much fewer combinations, it still outperforms EFI ($p < 0.05$).

5.4 EFMP and State of the art

Table 4 compares the performance level of the EFMP approach, plain and enhanced with the JIM-based feature induction module, against the top 5 CoNLL-2012 systems on the CoNLL-2012 test set.

As the results show, the EFMP approach achieves

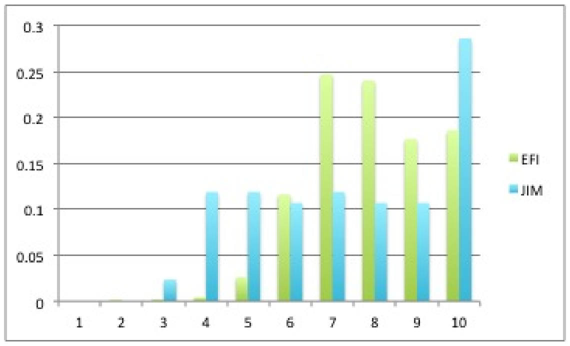

Figure 1: Normalized combination length for JIM and EFI: number of induced sets of size 1..9, 10+ divided by the total number of induced sets

results comparable to the state-of-the-art. At the same time, it's much faster than more complex approaches. The vast majority of high-performance coreference resolution systems (in particular, the CoNLL-2012 winning algorithm by Fernandes et al. (2012)) rely on complex structural representations and are therefore slow at the training stage. Our system only needs a simple binary mention-pair classifier that can be trained very efficiently. Some high-performance approaches rely on the same classifier, postponing a heavy global inference step to the decoding stage, for example, through Integer Linear Programming (Denis and Baldridge, 2009; Finkel

	development	test	
	v4	v4	v7
EFMP, all features, SVM			
none	60.02	59.12	55.38
EFI	61.66	60.75	57.56
JIM	62.53	61.82	59.14

Table 3: Feature combinations, JIM vs. EFI: MELA score, different versions of the CoNLL scorer.

	test	
	v4	v7
1 fernandes	63.37	60.65
EFMP+JIM	61.82	59.14
2 martschat	61.31	57.68
3 bjorkelund	61.24	57.42
EFMP	59.12	55.38
4 chang	60.18	56.10
5 chen	59.69	54.52

Table 4: EFMP and top-5 CoNLL-2012 systems: MELA score, systems ranked by the v7 score on the test set.

and Manning, 2008). While these systems have the same training requirements as EFMP, their decoding (ILP with binary variables) is known to be NP-complete. In practice, ILP-based approaches incorporating any forms of global modeling via transitivity constraints (Denis and Baldridge, 2009; Finkel and Manning, 2008) are known to be particularly slow. Our simple decoding algorithm runs in $O(p * log(p))$, where p is the total number of mention pairs: for the plain EFMP, $p = n * (n - 1)/2$, for the EFMP with sieves, $p = const * n$, where n is the number of mentions in the document.

6 Conclusion

In this study, we advocate an easy-first mention-pair model (EFMP). This approach combines the simplicity of mention-pair models with the high performance level of state-of-the-art systems. We believe that several research lines are open in the field of coreference resolution, ours being simple and allowing to focus more on low-level linguistic phenomena. Nevertheless, the approach shows a high performance level, despite the lack of any global inference (augmented with a feature induction module, our system would have achieved the second place at the CoNLL-2012 shared task, outperforming more complex algorithms). This suggests that there is still a lot of potential improvement that can be achieved within more complex frameworks, e.g., structural approaches that attempt at modeling links interdependence explicitly. One of our directions for future work involves comparing EFMP against other algorithms effectively combining positive and negative links, in particular, ILP-based approaches.

The proposed EFMP model allows for a straightforward investigation of possibilities for automatic feature induction. We have adapted the Jaccard Item Mining algorithm (JIM) to our task and compared its output against the Entropy-based Feature Induction (EFI) methodology proposed in the literature, showing that both techniques yield meaningful feature combinations and improve the system's performance. Yet, the JIM approach outputs smaller combinations, leading to a larger performance increase.

In our future work, we plan to focus further on the feature induction task, following several research directions. First, we want to apply automatic feature induction in a multilingual setting. Second, we plan to investigate other feature induction techniques: (i) comparing various similarity measures alternative to the Jaccard index in a JIM-style setting, (ii) trying to run EFI on different samples of the training set to obtain different decision trees and (iii) combining JIM and EFI-induced features. Finally, we want to verify our hypothesis that complex features represent meaningful linguistic combinations and as such can be used to enhance the performance level of more complex algorithms. This again would bridge the work on mention-pair and more advanced models.

Acknowledgments

This research is part of the Interactive sYstems for Answer Search (Iyas) project, conducted by the Arabic Language Technologies (ALT) group at the Qatar Computing Research Institute (QCRI) within the Qatar Foundation.

References

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 294–303, Stroudsburg, PA, USA. Association for Computational Linguistics.

Anders Björkelund and Richárd Farkas. 2012. Data-driven multilingual coreference resolution using resolver stacking. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 49–55, Jeju Island, Korea, July. Association for Computational Linguistics.

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.

Christian Borgelt. 2012. Frequent item set mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):437–456.

Pascal Denis and Jason Baldridge. 2009. Global joint models for coreference resolution and named entity classification. In *Procesamiento del Lenguaje Natural 42, Barcelona: SEPLN*.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Joint Conference on EMNLP and CoNLL - Shared Task*, CoNLL '12, pages 41–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jenny Rose Finkel and Christopher D. Manning. 2008. Enforcing transitivity in coreference resolution. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 45–48.

Hamidreza Kobdani, Hinrich Schütze, Andre Burkovski, Wiltrud Kessler, and Gunther Heidemann. 2010. Relational feature engineering of natural language processing. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM)*, pages 1705–1708.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, CONLL Shared Task '11, pages 28–34, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

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.

Cristina Nicolae and Gabriel Nicolae. 2006. Bestcut: A graph algorithm for coreference resolution. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 275–283, Stroudsburg, PA, USA. Association for Computational Linguistics.

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.

Marta Recasens and Eduard Hovy. 2009. A deeper look into features for coreference resolution. In *Anaphora Processing and Applications (DAARC 2009)*.

Marc Segond and Christian Borgelt. 2011. Item set mining based on cover similarity. In *Proceedings of the 15th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2011)*.

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.

Veselin Stoyanov and Jason Eisner. 2012. Easy-first coreference resolution. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 2519–2534.

Olga Uryupina, Alessandro Moschitti, and Massimo Poesio. 2012. BART goes multilingual: The UniTN / Essex submission to the CoNLL-2012 Shared Task. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning (CoNLL'12)*.

Olga Uryupina. 2006. Coreference resolution with and without linguistic knowledge. In *Proceedings of the Language Resources and Evaluation Conference (LREC'06)*.

Yannick Versley, Alessandro Moschitti, Massimo Poesio, and Xiaofeng Yang. 2008. Coreference systems based on kernels methods. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 961–968.