Abstract

Mention pair models that predict whether or not two mentions are coreferent have historically been very effective for coreference resolution, but do not make use of entity-level information. However, we show that the scores produced by such models can be aggregated to define powerful entity-level features between clusters of mentions. Using these features, we train an entity-centric coreference system that learns an effective policy for building up coreference chains incrementally. The mention pair scores are also used to prune the search space the system works in, allowing for efficient training with an exact loss function. We evaluate our system on the English portion of the 2012 CoNLL Shared Task dataset and show that it improves over the current state of the art.

1 Introduction

Coreference resolution, the task of identifying mentions in a text that refer to the same real world entity, is an important aspect of text understanding and has numerous applications. Many approaches to coreference resolution learn a scoring function defined over mention pairs to guide the coreference decisions (Soon et al., 2001; Ng and Cardie, 2002; Bengtson and Roth, 2008). However, such systems do not make use of entity-level information, i.e., features between clusters of mentions instead of pairs.

Using entity-level information is valuable because it allows early coreference decisions to inform later ones. For example, finding that Clinton and she corefer makes it more likely that Clinton corefers with Hillary Clinton than Bill Clinton due to gender agreement constraints. Such information has been incorporated successfully into entity-centric coreference systems that build up coreference clusters incrementally, using the information from the partially completed coreference chains produced so far to guide later decisions (Raghunathan et al., 2010; Stoyanov and Eisner, 2012; Ma et al., 2014).

However, defining useful features between clusters of mentions and learning an effective policy for incrementally building up clusters can be challenging, and many recent state-of-the-art systems work entirely or almost entirely over pairs of mentions (Fernandes et al., 2012; Durrett and Klein, 2013; Chang et al., 2013). In this paper we introduce a novel coreference system that combines the advantages of mention pair and entity-centric systems with model stacking. We first propose two mention pair models designed to capture different linguistic phenomena in coreference resolution. We then describe how the probabilities produced by these models can be used to generate expressive features between clusters of mentions. Using these features, we train an entity-centric incremental coreference system.

The entity-centric system builds up coreference chains with agglomerative clustering: each mention starts in its own cluster and then pairs of clusters are merged each step. We train an agent to determine whether it is desirable to merge a particular pair of clusters using an imitation learning algorithm based on DAgger (Ross et al., 2011). Previous incremental coreference systems heuristically define which actions are beneficial for the agent to perform, but we instead propose a way of assigning exact costs to actions based on coreference evaluation metrics, adding a concept of the severity of a mistake. Furthermore, rather than considering all pairs of clusters as candidate merges, we use the scores of the pairwise models to reduce the search space, first by providing an ordering over which merges are considered and secondly by discarding merges that are not likely
to be good. This greatly reduces the time it takes to run the agent, making learning computationally feasible.

Imitation learning is challenging because it is a non-i.i.d. learning problem; the distribution of states seen by the agent depends on the agent’s parameters. Model stacking offers a way of decomposing the learning problem by training pairwise models with many parameters in a straightforward supervised learning setting and using their outputs for training a much simpler model in the more difficult imitation learning setting. Furthermore, mention pair scores can produce powerful features for training the agent because the scores indicate which mention pairs between the clusters in question are relevant; high scoring and low scoring pairs can indicate when a merge should be forced or disallowed while other mention pairs may provide little useful information.

We run experiments on the English portion of the 2012 CoNLL Shared Task dataset. The entity-centric clustering algorithm greatly outperforms commonly used heuristic methods for coordinating pairwise scores to produce a coreference partition. We also show that combining the scores of different pairwise models designed to capture different aspects coreference results in significant gains in accuracy. Our final system gets a combined score of 63.02 on the dataset, substantially outperforming other state of the art systems.

2 Mention Pair Models

Mention pair models predict whether or not a given pair of mentions belong in the same coreference cluster. We incorporate two different mention pair models into our system. However, other pairwise models could easily be added; one advantage of our model stacking approach is that it can combine different simple classifiers in a modular way.

Our two models are designed to capture different aspects of coreference. The first one is built to predict coreference for all of the candidate antecedents of a mention. This makes it useful for providing scores when the current mention has clear coreference links to many previous mentions. For example President Clinton might be linked to the president, Bill Clinton, and Mr. President.

However, mentions often only have one clear antecedent. This is especially common in pronominal anaphora resolution, such as in the sentence Bill arrived, but nobody saw him. The pronoun him is directly referring back to a previous part of the discourse, not some entity that other mentions may also refer to. However, there still might be coreference links between him and previous mentions in the text because of transitivity: any other mention about Bill would be coreferent with him. For such mentions, there may be very little evidence in the discourse to suggest a coreference link, so attempting to train a model to predict these will bear little fruit. With this as motivation, we also train a model to predict only one correct antecedent of the current mention.

We found a classification model to be well suited for the first task and and a ranking model to be well suited for the second one. These two models differ only in the training criteria used. Both models use a logistic classifier to assign a probability to a mention and a candidate antecedent representing the likelihood that the two mentions are coreferent. The candidate antecedent may take on the value NA indicating that has no antecedent. The probability of coreference takes the standard logistic form:

\[ p_\theta(a, m) = \left(1 + e^{\theta^T f(a, m)}\right)^{-1} \]

where \( f(a, m) \) is a vector of feature functions on \( a \) and \( m \) and \( \theta \) are the feature weights we wish to learn. Let \( M \) denote the set of all mentions in the training set, \( T(m) \) denote the set of true antecedents of a mention \( m \) (i.e., mentions that occur before \( m \) in the text that are coreferent with \( m \) or \{NA\} if \( m \) has no antecedent), and \( F(m) \) denote the set of false antecedents of \( m \). We want to find a parameter vector \( \theta \) that assigns high probabilities to the candidate antecedents in \( T(m) \) and low probabilities to the ones in \( F(m) \).

2.1 Classification Model

For the classification model, we consider each pair of mentions independently with the goal of predicting coreference correctly for as many of them as possible. The model is trained by minimizing negative conditional log likelihood augmented with L1 regularization:

\[
\mathcal{L}_c(\theta_c) = - \sum_{m \in M} \left( \sum_{t \in T(m)} \log p_{\theta_c}(t, m) \right) + \sum_{f \in F(m)} \log(1 - p_{\theta_c}(f, m))) + \lambda ||\theta_c||_1
\]
By summing over all candidate antecedents, the objective encourages the model to produce good probabilities for all of them.

2.2 Ranking Model

For the ranking model, candidate antecedents for a mention are considered simultaneously and compete with each other to be matched with the current mention. This makes the model well suited to the task of finding a single best antecedent for a mention. A natural learning objective for such a model would be a max-margin training criteria that encourages separation between the highest scoring true antecedent and highest scoring false antecedent of the current mention. However, we found such models to be poor at producing scores useful for a downstream clustering model because a max-margin objective encourages scores for true antecedents to be high only relative to other candidate antecedents. It is much more beneficial to have mention pair scores that are comparable across different mentions as well as different candidate antecedents. For this reason, we instead train the model with an objective that maximizes the conditional log likelihood of the highest scoring true and false antecedents under the logistic model:

$$L_r(\theta_r) = -\sum_{m \in M} \left( \max_{t \in T(m)} \log p_{\theta_r}(t,m) + \min_{f \in F(m)} \log(1 - p_{\theta_r}(f,m)) \right) + \lambda \|	heta_r\|_1$$

For both models, we set $\lambda = 0.001$ and optimize their objectives using AdaGrad (Duchi et al., 2011).

2.3 Features

Our mention pair models use a variety of common features for mention pair classification (for more details see (Bengtson and Roth, 2008; Steyanov et al., 2010; Lee et al., 2011; Recasens et al., 2013)). These include

- **Distance** features, e.g., the distance between the two mentions in sentences or number of mentions.
- **Syntactic** features, e.g., number of embedded NPs under a mention, POS tags of the first, last, and head word.
- **Semantic** features, e.g., named entity type, speaker identification.
- **Rule-based** features, e.g., exact and partial string matching.
- **Lexical Features**, e.g., the first, last, and head word of the current mention.

We also employ a feature conjunction scheme similar to the one described by Durrett and Klein (2013).

3 Entity-Centric Coreference Model

Mention pair scores alone are not enough to produce a final set of coreference clusters because they do not enforce transitivity: if the pair of mentions $(a, b)$ and the pair of mentions $(b, c)$ are deemed coreferent by the model, there is no guarantee that the model will also classify $(a, c)$ as coreferent. Thus a second step is needed to coordinate the scores to produce a final coreference partition. A widely used approach for this is best-first clustering (Ng and Cardie, 2002). For each mention, the best-first algorithm assigns the most probable preceding mention classified as coreferent with it as the antecedent.

The primary weakness of this approach is that it only relies on local information to make decisions, so it cannot consolidate information at the entity level. As a result, coreference chains produced by such algorithms can exhibit low coherency. For example, a cluster may consist of [Hillary Clinton, Clinton, he] because the coreference decision between Hillary Clinton and Clinton, he] because the coreference decision between Hillary Clinton and Clinton is made independently of the one between Clinton and he.

To tackle this problem, we build an entity-centric model that operates between pairs of clusters instead of pairs of mentions, guided by scores produced by the pairwise models. It builds up clusters of mentions believed to refer to the same entity as it goes, relying on the partially formed clusters produced so far to make decisions. For example, the system could reject linking [Hillary Clinton] with [Clinton, he] because of the low score between the pair (Hillary Clinton, he).

Our entity-centric “agent” builds up coreference chains with agglomerative clustering. It begins in a start state where each mention is in a separate single-element cluster. At each step, it observes the current state $s$, which consists of all partially formed coreference clusters produced so far, and selects some action $a$ which merges two existing clusters. The action will result in a new state with new candidate actions and the process is repeated. The model is entity-centric in that it builds...
up clusters of mentions representing entities and merges clusters if it predicts they are representing the same one.

### 3.1 Test-time Inference

The agent assigns a score to each action $a$ using a linear model with feature function $f_e$ and weight vector $\theta_e$: $s_{\theta_e}(a) = \theta_e^T f_e(a)$. A particular setting of $\theta_e$ defines a policy $\pi$ that determines which action $a = \pi(s)$ the agent will take in state $s$. This policy is to greedily take highest scoring candidate action available from the current state.

Rather than using all possible cluster merges as the candidate set of actions the agent selects from, we use the scores produced by mention pair models to reduce the search space. First, we order all mention pairs in the document in descending order according to their pairwise scores. This causes clustering to occur in an easy-first fashion, where harder decisions are delayed until more information is available. Secondly, we discard all mention pairs that score below a threshold $t$ under the assumption that the clusters containing these pairs are unlikely to be coreferent. In our experiments we were able able set $t$ so that over 95% of pairs were removed with no decrease in accuracy. Lastly, we iterate through this list of pairs in order. For each pair, we make a binary decision on whether or not the clusters containing these pairs should be merged. This formulates the agent’s task so it only has two actions to chose from instead of a number of actions proportional to the number of clusters squared. Algorithm 1 shows the full test-time procedure.

### 3.2 Learning

**Imitation Learning with DAgger**

We face a sequential prediction problem where future observations (visited states) depend on previous actions. This is challenging because it violates the common i.i.d. assumptions made in statistical learning. *Imitation learning*, where expert demonstrations of good behavior are used to teach the agent, has proven very useful in practice for this sort of problem (Argall et al., 2009). We use imitation learning to set the parameters $\theta_e$ of our agent by training it to classify whether a particular action is the one an expert policy would take in the current state. In particular, we use $\theta_e$ as parameters for a binary logistic classifier that predicts which action (merge or do not merge) matches the expert policy.

---

**Algorithm 1** Inference method: agglomerative clustering

**Input:** Set of mentions in document $\mathcal{M}$, pairwise classifier with parameters $\theta_e$, agent with parameters $\theta_e$, cutoff threshold $t$

**Output:** Clustering $C$

```python
# Initialize list of mention pairs $P \rightarrow []$
for each pair $(m_i, m_j) \in \mathcal{M}^2$ with $i < j$
  if $p_{\theta_e}(m_i, m_j) > t$ then
    $P.append((m_i, m_j))$
end if
end for

# Sort $P$ in descending order according to $p_{\theta_e}$

# Initialize $C \rightarrow$ initial clustering with each mention in $\mathcal{M}$ in its own cluster
for $(m_i, m_j) \in P$
  if $C[m_i] \neq C[m_j]$
    and $s_{\theta_e}(C[m_i], C[m_j]) > 0$ then
      DoMerge($C[m_i], C[m_j], C$)
    end if
end for
```

We found the DAgger (Ross et al., 2011) imitation learning method (see Algorithm 2) to be effective for this task. DAgger is an iterative algorithm that aggregates a dataset $\mathcal{D}$ consisting of states and the actions performed by the expert policy in those states. At each iteration, it first samples a trajectory of states visited by the current policy by running the policy to completion from the start state. It then labels those states with the best action according to the expert policy, adds those labeled examples to the dataset, and then trains a new classifier over the dataset to get a new policy. When producing a trajectory to train on, the expert policy is stochastically mixed with the current policy; with probability $\beta_i$ the expert’s action is chosen instead of the current policy’s. We set $\beta$ so it decays exponentially as the iteration number increases.

By sampling trajectories under the current policy, DAgger exposes the system to states at train time similar to the ones it will face at test time. In contrast, training the agent on the gold labels alone would unrealistically teach it to make decisions under the assumption that all previous decisions were correct, potentially causing it to over-rely on information from past actions. This is especially problematic in coreference, where the error rate is quite high. Even when using DAgger,
Assigning Costs to Actions

A key aspect of incrementally building coreference clusters is that some local decisions are much more important than others. For example, a merge between two large clusters influences the score far more than a merge between two small ones. Additionally, getting early decisions correct is crucial because later actions are dependent on early ones, causing errors to compound if mistakes are made early. To capture this, we take an approach inspired by the SEARN learning algorithm (Daumé et al., 2009) and add costs to the actions in the aggregated dataset. We then train the agent to do cost-sensitive classification. Using these costs, we can exploit this with memoization, significantly improving the algorithm’s runtime. In particular, we store the following values:

- Given a state \( s \) and action \( a \), the value of the cost function, \( r(s, a) \).
- Given an action \( a \), the score the model assigns that action, \( s_\theta_e(a) \).
- Given an action \( a \), the result of the feature function on that action, \( f_e(a) \).

The first two values depend on the current model, so the saved values must be cleared between iterations of training. In experiments on the development set of the CoNLL 2012 corpus, these tables had 76%, 94%, and 93% hit rates respectively after 50 passes over the dataset.

### 3.3 Features

Our agent uses features that are derived from the scores produced by the two mention pair models. Although these scores only operate on mention pairs, they are combined to capture cluster-level interactions by being aggregated in different ways over pairs of mentions from the clusters. Mention pair scores can produce powerful features for training the agent because they show which mention pairs between the clusters in question are relevant, and often a small subset of the mention pairs provide far more information than the rest. For example, a strong negative pairwise
Figure 1: Examples of features generated for a candidate cluster merge. Weights on edges are the probabilities of coreference produced by a mention pair model.

link like *Hillary Clinton* and *he* should disallow a merge, while other mention pairs, such as two instances of the pronoun *she* far apart in the text, might provide very little information. Using the mention pair models for probabilities, we compute the following features over all pairs of mentions between the clusters (i.e., each mention is in a different cluster).

- The minimum and maximum probability of coreference.
- The average probability and average log probability of coreference.
- The average probability and log probability of coreference for a particular pair of grammatical types of mentions (either pronoun or non-pronoun). For example, **Avg-Prob_non-pronoun_pronoun** gives the average probability of coreference when the candidate antecedent is not a pronoun and the candidate anaphor is a pronoun.

Note that the averaged features have a natural probabilistic interpretation; the average probability corresponds to the expected number of coreference links between the involved mention pairs while the average log probability corresponds to the probability that all mention pairs will have a coreference link. All of these features are computed twice: once with the classification model and once with the ranking model.

We also compute the following features based on other aspects of the current state:

- Whether a preceding mention pair in the list of mention pairs has the same candidate anaphor as the current one.
- The index of the current mention pair in the list divided by the size of the list, i.e., what percentage of the list have we seen so far.
- The number of mentions in the current document.
- The probability of the first-occurring mention in the second-occurring cluster not being anaphoric (i.e., $p_\theta((\text{NA}, m))$). This prevents producing clusters that, for example, start with a pronoun.

Lastly, we take one feature conjunction with a boolean representing whether both clusters are size 1. In total, there are only 56 features after the feature conjunction. However, these features provide strong signal because they are directly related to the probabilities of mentions being coreferent. In contrast, the pairwise models use thousands of features (after feature conjunctions), including lexical features that are extremely sparse. The pairwise models can easily exploit this much bigger feature set because they operate in a classic supervised learning setting. The entity-centric model, on the other hand, learns in a much more challenging non-i.i.d. setting. Model stacking avoids the difficulty of directly training the entity-centric model with a large set of weak features by decomposing the task into first learning to produce good pairwise scores and then using those scores to generate a manageable set of strong features.

### 3.4 Training Details

Because the entity-centric agent relies on the output of pairwise classifiers, they should not be trained on the same data. Therefore we split the
training set into two sections and use one for training the pairwise models and the other for training the agent. When evaluating on the development set, we use 80% of the documents in the training set to train the mention pair models and the rest to train the entity-centric model. When evaluating on the test set we use the whole training set for the mention pair models and the development set for the entity-centric model. We also tried using cross-validation instead of a single split, but found this did not improve performance, which we believe to be because this trains the agent with different pairwise models than the ones used at test time.

For our initial policy \(\hat{\pi}_1\), we set the parameters of the agent so it operates with simple best-first clustering (initializing all feature weights to 0 except for the maximum-score, anaphor-seen, and bias features). For \(m\), the performance metric determining the action costs, we use a linear combination of the \(B^3\) (Bagga and Baldwin, 1998) and MUC (Vilain et al., 1995) metrics, which are both commonly used for evaluating coreference systems. The other metric used in our evaluation, Entity-based CEAFE (CEAF\(_{\phi_4}\)) (Luo, 2005), was not used because it is expensive to compute. We found weighting \(B^3\) three times as much as MUC to be effective on the development set.

### 4 Experiments and Results

#### Experimental Setup

We apply our model to the English portion of the CoNLL 2012 Shared Task data (Pradhan et al., 2012), which is derived from the OntoNotes corpus (Hovy et al., 2006). The data is split into a training set of 2802 documents, development set of 343 documents, and a test set of 345 documents. We use the provided preprocessing for parse trees, named entity tags, etc. The models are evaluated using three of the most popular metrics for coreference resolution: MUC, \(B^3\), and Entity-based CEAFE (CEAF\(_{\phi_4}\)). We also include the average \(F_1\) score (CoNLL \(F_1\)) of these three metrics, as is commonly done in CoNLL Shared Tasks. We used the most recent version of the CoNLL scorer (version 8.01), which implements the original definitions of these metrics.

#### Mention Detection

Our experiments were run using system-produced predicted mentions. We used the rule-based mention detection algorithm from Raghunathan et al. (2010), which first extracts pronouns and maximal NP projections as candidate mentions and then filters this set with rules that remove spurious mentions such as numeric entities or pleonastic \(it\) pronouns.

#### Comparison of Models

We compare the effectiveness of the entity-centric model with the commonly used best-first clustering approach, which assigns mentions the highest scoring previous mention as the antecedent. Unlike the entity-centric model, the best-first approach only relies on local information to make decisions. We also compare the effectiveness of the ranking and classification pairwise models. Table 1 shows the results of these models on the development set.

<table>
<thead>
<tr>
<th>Model Type</th>
<th>MUC</th>
<th>(B^3)</th>
<th>CEAF(_{\phi_4})</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification, B.F.</td>
<td>72.00</td>
<td>60.01</td>
<td>55.63</td>
<td>62.55</td>
</tr>
<tr>
<td>Ranking, B.F.</td>
<td>71.91</td>
<td>60.63</td>
<td>56.38</td>
<td>62.97</td>
</tr>
<tr>
<td>Classification, E.C.</td>
<td>72.34</td>
<td>61.46</td>
<td>57.16</td>
<td>63.65</td>
</tr>
<tr>
<td>Ranking, E.C.</td>
<td>72.37</td>
<td>61.34</td>
<td>57.13</td>
<td>63.61</td>
</tr>
<tr>
<td>Both, E.C.</td>
<td>72.52</td>
<td>62.02</td>
<td>57.69</td>
<td>64.08</td>
</tr>
</tbody>
</table>

Table 1: Metric scores on the development set for the classification and ranking pairwise models when using best-first clustering (B.F.) or the entity-centric model (E.C.).

Incorporating other Entity-Level Features

Although the entity-centric model has so far only
used features derived from the scores produced by mention pair models, other entity-level features could easily be added. We experiment with this by adding four cluster-level agreement features based on gender, number, animacy, and named entity type. Each of these features can take on three values: “same” (e.g., both clusters have gender value feminine), “compatible” (e.g., one cluster has gender value feminine while the other has value unknown), or “incompatible” (one cluster has gender value feminine while the other has value masculine). The cluster-level value for a particular feature is the most common value among mentions in that cluster (e.g., if a cluster has 2 masculine mentions, 1 feminine mention, and 1 unknown mention) the value is considered masculine. Table 2 shows the results.

Adding the additional features had no substantial impact on scores, suggesting that features derived from pairwise scores are sufficient for capturing this kind of entity-level information. A disagreement between clusters necessarily means there will be disagreements between some of the involved mentions, so features like the average and minimum probability between mention pairs will have lower values when a disagreement is present.

### Final System Performance

In Table 3 we compare the results of our system with the following state-of-the-art approaches: the JOINT and INDEP models of the Berkeley system (Durrett and Klein, 2014) (the JOINT model jointly does NER and entity linking along with coreference); the Prune-and-Score system (Ma et al., 2014); the HOTCoref system (Björklund and Kuhn, 2014); the CPL3M system (Chang et al., 2013); and Fernandes et al (2012). We use the full entity-centric clustering algorithm drawing upon scores from both pairwise models. We do not make use of agreement features, as these did not increase accuracy and complicate the system. Our final model substantially outperforms the other systems on the CoNLL $F_1$ score. The largest improvement is in the $B^4$ metric, which is unsurprising because the entity-centric model primarily optimizes for this during training. However, our model also achieves the highest CEAFφ4 $F_1$ and second highest MUC $F_1$ scores among the other systems.

### 5 Related Work

Both mention pair (Soon et al., 2001; Ng and Cardie, 2002; Bengtson and Roth, 2008; Stoyanov et al., 2010; Björklund and Farkas, 2012) and mention ranking models (Denis and Baldridge, 2007b; Rahman and Ng, 2009) have been widely used for coreference resolution, and there have been many proposed ways of post-processing the pairwise scores to make predictions. Despite their simplicity, closest-first clustering (Soon et al., 2001) and best-first clustering (Ng and Cardie, 2002) are arguably the most widely used of these approaches. Other work uses global inference with integer linear programming to enforce transitivity (Denis and Baldridge, 2007a; Finkel and Manning, 2008), graph partitioning algorithms (McCallum and Wellner, 2005; Nicolae and Nicolae, 2006), the Dempster-Shafer rule (Kehler, 1997; Bean and Riloff, 2004), or correlational clustering (McCallum and Wellner, 2003; Finley and Joachims, 2005). In contrast to these methods, our entity-centric model directly learns how to use pairwise scores to produce a coreference partition that scores highly according to an evaluation metric, and can use the outputs of more than one mention pair model.

Recently, coreference models using latent antecedents have gained in popularity and achieved state-of-the-art results (Fernandes et al., 2012; Durrett and Klein, 2013; Chang et al., 2013; Björklund and Kuhn, 2014). These learn a scoring function over mention pairs, but are trained to maximize a global objective function instead of pairwise accuracy. Unlike in our system, these methods typically consider one pair of mentions at a time during inference.

Several works have explored using non-local entity-level features in mention-entity models that assign a single mention to a (partially completed) cluster (Luo et al., 2004; Yang et al., 2008; Rahman and Ng, 2011). Our system, however, builds clusters incrementally through merge operations, and so can operate in an easy-first fashion. Raghu-nathan et al. (2010) take this approach with a

<table>
<thead>
<tr>
<th>Scores Only</th>
<th>MUC</th>
<th>$B^4$</th>
<th>CEAFφ4</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>72.52</td>
<td>62.02</td>
<td>57.69</td>
<td>64.08</td>
</tr>
<tr>
<td>+Agreement</td>
<td>72.59</td>
<td>61.98</td>
<td>57.58</td>
<td>64.05</td>
</tr>
</tbody>
</table>

Table 2: Metric scores on the development set for the entity-centric model with and without the addition of entity-level agreement features.
Table 3: Comparison of this work with other state-of-the-art approaches on the test set.

<table>
<thead>
<tr>
<th></th>
<th>MUC</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fernandes et al.</td>
<td>75.91</td>
<td>65.83</td>
<td>70.51</td>
<td>65.19</td>
<td>51.55</td>
<td>57.58</td>
</tr>
<tr>
<td>Chang et al.</td>
<td>-</td>
<td>-</td>
<td>69.48</td>
<td>-</td>
<td>-</td>
<td>57.44</td>
</tr>
<tr>
<td>Björkelund &amp; Kuhn</td>
<td>74.3</td>
<td>67.46</td>
<td>70.72</td>
<td>62.71</td>
<td>54.96</td>
<td>58.58</td>
</tr>
<tr>
<td>Ma et al.</td>
<td>81.03</td>
<td>66.16</td>
<td>72.84</td>
<td>66.90</td>
<td>51.10</td>
<td>57.94</td>
</tr>
<tr>
<td>Durrett &amp; Klein (INDEP.)</td>
<td>72.27</td>
<td>69.30</td>
<td>70.75</td>
<td>60.92</td>
<td>55.73</td>
<td>58.21</td>
</tr>
<tr>
<td>Durrett &amp; Klein (JOINT)</td>
<td>72.61</td>
<td>69.91</td>
<td>71.24</td>
<td>61.18</td>
<td>56.43</td>
<td>58.71</td>
</tr>
<tr>
<td>This work</td>
<td>76.12</td>
<td>69.38</td>
<td>72.59</td>
<td>65.64</td>
<td>56.01</td>
<td>60.44</td>
</tr>
</tbody>
</table>

rule-based system that runs in multiple passes and Stoyanov and Eisner (2012) train a classifier to do this with a structured perceptron algorithm. Entity-level information has also been successfully incorporated in coreference systems using joint inference (McCallum and Wellner, 2003; Culotta et al., 2006; Poon and Domingos, 2008; Haghighi and Klein, 2010), but these approaches do not directly learn parameters tuned so the system runs effectively at test time, while our imitation learning approach does.

Imitation learning has been employed to train coreference resolvers on trajectories of decisions similar to those that would be seen at test-time by Daumé et al. (2005) and Ma et al. (2014). Other works use structured perceptron models for the same purpose (Stoyanov and Eisner, 2012; Fernandes et al., 2012; Björkelund and Kuhn, 2014). These systems all heuristically determine which actions are desirable for the system to perform. In contrast, our approach directly computes a cost for actions based on coreference evaluation metrics. This means our system directly learns which actions lead to good clusterings instead of which look good locally according to a heuristic. Furthermore, the costs provide our system a measure of the severity of a mistake, which we argue is very beneficial for the coreference task.

Our model stacking approach further distinguishes this work by providing a new way of defining cluster-level features. The majority of useful features for coreference systems operate on pairs of mentions (in one of our experiments we show the addition of classic entity-level features does not improve our system), but incremental coreference systems must make decisions involving many mention pairs. Other incremental coreference systems either incorporate features from a single pair (Stoyanov and Eisner, 2012) or average features across all pairs in the involved clusters (Ma et al., 2014). Our system instead combines information from the involved mention pairs in a variety of ways with with higher order features produced from the scores of mention pair models.

6 Conclusion

We introduced a new approach to coreference resolution that trains an entity-centric system using the scores produced by mention pair models as features. The brunt of task-specific learning occurs within the mention pair models, which are trained in a straightforward supervised manner. Guided by the pairwise scores, our entity-centric agent then learns an effective procedure for building up coreference clusters incrementally, using previous decisions to inform later ones. The agent benefits from using multiple mention pair models designed to capture different aspects of coreference. Experiments show that the agent, which learns how to coordinate mention pair scores, outperforms the commonly used best-first method. We evaluate our final system on the English portion of the CoNLL 2012 Shared Task and report a significant improvement over the current state of the art.

Acknowledgments

We thank the anonymous reviewers for their thoughtful comments. Stanford University gratefully acknowledges the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.
References


