Entity-Centric Coreference Resolution with Model Stacking

Kevin Clark
Computer Science Department
Stanford University
kevclark@cs.stanford.edu

Christopher D. Manning
Computer Science Department
Stanford University
manning@cs.stanford.edu

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

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 1405–1415,
Beijing, China, July 26-31, 2015.

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 m and candidate antecedent a representing the likelihood that the two mentions are coreferent. The candidate antecedent a may take on the value NA indicating that m has no antecedent. The probability of coreference takes the standard logistic form:

$$p_{\boldsymbol{\theta}}(a, m) = (1 + e^{\boldsymbol{\theta}^T \boldsymbol{f}(a,m)})^{-1}$$

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

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:

$$\mathcal{L}_c(\boldsymbol{\theta}_c) = -\sum_{m \in \mathcal{M}} \Big(\sum_{t \in \mathcal{T}(m)} \log p_{\boldsymbol{\theta}_c}(t, m) + \sum_{f \in \mathcal{F}(m)} \log(1 - p_{\boldsymbol{\theta}_c}(f, m)) \Big) + \lambda ||\boldsymbol{\theta}_c||_1$$

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:

$$\mathcal{L}_r(\boldsymbol{\theta}_r) = -\sum_{m \in \mathcal{M}} \Big(\max_{t \in \mathcal{T}(m)} \log p_{\boldsymbol{\theta}_r}(t, m) + \min_{f \in \mathcal{F}(m)} \log(1 - p_{\boldsymbol{\theta}_r}(f, m)) \Big) + \lambda ||\boldsymbol{\theta}_r||_1$$

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; Stoyanov 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* 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 $\boldsymbol{f_e}$ and weight vector $\boldsymbol{\theta_e}$: $s_{\boldsymbol{\theta_e}}(a) = \boldsymbol{\theta_e}^T \boldsymbol{f_e}(a)$. A particular setting of $\boldsymbol{\theta_e}$ defines a *policy* π 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 $\boldsymbol{\theta_e}$ 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 $\boldsymbol{\theta_e}$ 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 $\boldsymbol{\theta}_c$, agent with parameters $\boldsymbol{\theta}_e$, cutoff threshold t
Output: Clustering C

Initialize list of mention pairs $P \rightarrow []$
for each pair $(m_i, m_j) \in \mathcal{M}^2$ with $i < j$ **do**
  if $p_{\boldsymbol{\theta}_c}(m_i, m_j) > t$ **then**
    P.append((m_i, m_j))
  end if
end for
Sort P in descending order according to $p_{\boldsymbol{\theta}_c}$

Initialize $C \rightarrow$ initial clustering with each mention in $\mathcal{M}$ in its own cluster
for $(m_i, m_j) \in P$ **do**
  if $C[m_i] \neq C[m_j]$
  and $s_{\boldsymbol{\theta_e}}(C[m_i], C[m_j]) > 0$ **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 β_i the expert's action is chosen instead of the current policy's. We set β 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,

this problem could exist to a lesser degree if the model heavily overfits to the training data. However, the agent has a small number of parameters thanks to our model stacking approach, reducing the risk of this happening.

Algorithm 2 Learning method: DAgger

Input: initial policy $\hat{\pi}_1$, expert policy π^*
Output: final policy $\hat{\pi}_N$

Initialize $\mathcal{D} \leftarrow \emptyset$
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$
  Sample a trajectory under the current policy using π_i
  Get dataset $\mathcal{D}_i = (s, \pi^*(s))$ of states visited by π_i and actions given by the expert
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$
end for

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 simply define the expert policy as the policy that takes the action with the lowest cost at each step.

We want our costs to represent how a particular local decision will affect the final score of the coreference system. Unfortunately, standard coreference evaluation metrics do not decompose over cluster merges. Instead, we compute the loss of an action by "rolling out" the current policy to completion. More concretely, let m be a function (such as a coreference evaluation metric) that assigns scores to states; we are interested in reaching a final state for which m is high. Suppose we are assigning costs to the set of actions $\mathcal{A}(s)$ that can be taken from some state s. For each action $a \in \mathcal{A}(s)$, we apply that action to s to get a new state s', run the current policy $\hat{\pi}_i$ from s' to completion, and then compute the value of m on the resulting final state. This gives exactly the final score the system would get if it made the action a from state s and then continued under the current policy. Let $f_m(s, a)$ denote this value for a particular metric, state, and action. We assign each action the regret r associated with taking that action under the current policy as a cost:

$$r(s, a) = \max_{a' \in \mathcal{A}(s)} f_m(s, a') - f_m(s, a)$$

The "rolling out" procedure means we naively have to visit $O(t^2)$ states each iteration instead of t, where t is the length of a trajectory. However, the highly constrained action space described in section 3.1 combined with the use of memoization allows the algorithm to still run efficiently.

Improving Runtime with Memoization

During training, the agent will see many of the same states and actions multiple times. 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, $s_{\boldsymbol{\theta_e}}(a)$.
- Given an action a, the result of the feature function on that action, $\mathbf{f}_e(a)$.

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

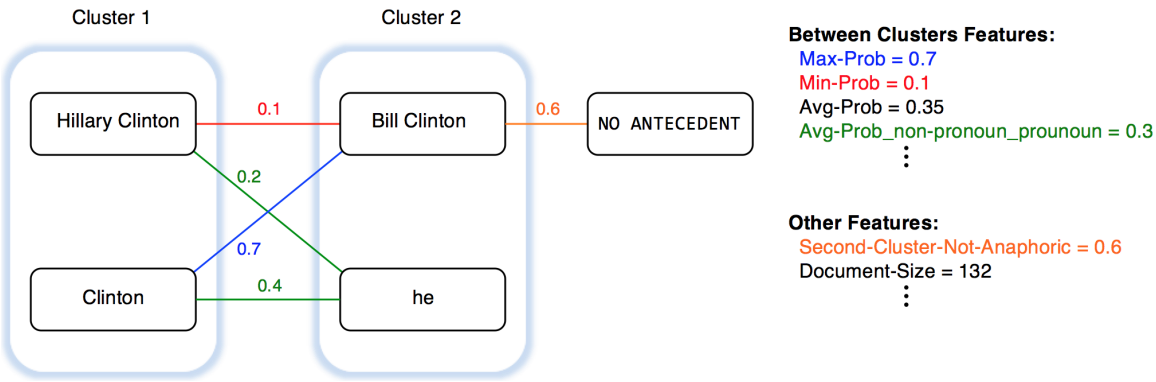

Figure 1: Examples of features generated for a candidate cluster merge. Weights on edges are the probabilities of coreference produced by a mention pair model.

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., $p_{\boldsymbol{\theta}_c}(\text{NA}, m)$). 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.

	MUC	B^3	$CEAF_{\phi_4}$	Avg.
Classification, B.F.	72.00	60.01	55.63	62.55
Ranking, B.F.	71.91	60.63	56.38	62.97
Classification, E.C.	72.34	61.46	57.16	63.65
Ranking, E.C.	72.37	61.34	57.13	63.61
Both, E.C.	72.52	62.02	57.69	64.08

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.).

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.

The entity-centric model outperforms best-first clustering for both mention pair models, demonstrating the utility of a learned, incremental clustering algorithm. The improvement is much greater for the classification pairwise model, causing it to outperform the ranking model with the entity-centric clustering algorithm even though it performs significantly worse than the ranking model with best-first clustering. This suggests that although the ranking model is better at finding a single correct antecedent for a mention, the classification model is more useful for producing cluster-level features. Incorporating probabilities from both pairwise models further improved scores over using either model alone, indicating that the mention pair classifiers were successful in learning scoring functions useful in different circumstances.

Incorporating other Entity-Level Features

Although the entity-centric model has so far only

	MUC	B^3	$CEAF_{\phi_4}$	Avg.
Scores Only	72.52	62.02	57.69	64.08
+Agreement	72.59	61.98	57.58	64.05

Table 2: Metric scores on the development set for the entity-centric model with and without the addition of entity-level agreement features.

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 (Björkelund and Kuhn, 2014); the CPL^{3}M sytem (Chang et al., 2013); and Fernandes et al. 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^3 metric, which is unsurprising because the entity-centric model primarily optimizes for this during training. However, our model also achieves the highest $CEAF_{\phi_4}$ 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; Björkelund 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; Björkelund 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. Raghunathan et al. (2010) take this approach with a rule-based system that runs in multiple passes

	MUC			B^3			$CEAF_{\phi_4}$			CoNLL
	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Avg. F_1
Fernandes et al.	75.91	65.83	70.51	65.19	51.55	57.58	57.28	50.82	53.86	60.65
Chang et al.	-	-	69.48	-	-	57.44	-	-	53.07	60.00
Björkelund & Kuhn	74.3	67.46	70.72	62.71	54.96	58.58	59.4	52.27	55.61	61.63
Ma et al.	81.03	66.16	**72.84**	66.90	51.10	57.94	68.75	44.34	53.91	61.56
Durrett & Klein (INDEP.)	72.27	69.30	70.75	60.92	55.73	58.21	55.33	54.14	54.73	61.23
Durrett & Klein (JOINT)	72.61	69.91	71.24	61.18	56.43	58.71	56.17	54.23	55.18	61.71
This work	76.12	69.38	72.59	65.64	56.01	**60.44**	59.44	52.98	**56.02**	**63.02**

Table 3: Comparison of this work with other state-of-the-art approaches on the test set.

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

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

David L Bean and Ellen Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. In *Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*, pages 297–304.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 294–303.

Anders Björkelund and Richárd Farkas. 2012. Data-driven multilingual coreference resolution using resolver stacking. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning - Shared Task*, pages 49–55.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Association of Computational Linguistics (ACL)*.

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A constrained latent variable model for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 601–612.

Aron Culotta, Michael Wick, Robert Hall, and Andrew McCallum. 2006. First-order probabilistic models for coreference resolution. In *Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*, pages 81–88.

Hal Daumé III and Daniel Marcu. 2005. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 97–104.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325.

Pascal Denis and Jason Baldridge. 2007a. Joint determination of anaphoricity and coreference resolution using integer programming. In *Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*, pages 236–243.

Pascal Denis and Jason Baldridge. 2007b. A ranking approach to pronoun resolution. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 1588–1593.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1971–1982.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics (TACL)*, 2:477–490.

Eraldo Rezende Fernandes, Cícero Nogueira Dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning - Shared Task*, pages 41–48.

Jenny Rose Finkel and Christopher D Manning. 2008. Enforcing transitivity in coreference resolution. In *Association for Computational Linguistics (ACL), Short Paper*, pages 45–48.

Thomas Finley and Thorsten Joachims. 2005. Supervised clustering with support vector machines. In *Proceedings of the 22nd international conference on Machine learning*, pages 217–224.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*, pages 385–393.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*, pages 57–60.

Andrew Kehler. 1997. Probabilistic coreference in information extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 163–173.

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 Conference on Computational Natural Language Learning: Shared Task*, pages 28–34.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Association for Computational Linguistics (ACL)*, page 135.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 25–32.

Chao Ma, Janardhan Rao Doppa, J Walker Orr, Prashanth Mannem, Xiaoli Fern, Tom Dietterich, and Prasad Tadepalli. 2014. Prune-and-score: Learning for greedy coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Andrew McCallum and Ben Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI Workshop on Information Integration on the Web*.

Andrew McCallum and Ben Wellner. 2005. Conditional models of identity uncertainty with application to noun coreference. In *Advances in Neural Information Processing Systems (NIPS)*, pages 905–912.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Association of Computational Linguistics (ACL)*, pages 104–111.

Cristina Nicolae and Gabriel Nicolae. 2006. Bestcut: A graph algorithm for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 275–283.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with markov logic. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 650–659.

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 Joint Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning - Shared Task*, pages 1–40.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 492–501.

Altaf Rahman and Vincent Ng. 2009. Supervised models for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 968–977.

Altaf Rahman and Vincent Ng. 2011. Narrowing the modeling gap: a cluster-ranking approach to coreference resolution. *Journal of Artificial Intelligence Research (JAIR)*, pages 469–521.

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. In *Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*, pages 627–633.

Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Artificial Intelligence and Statistics (AISTATS)*, pages 627–633.

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

Veselin Stoyanov and Jason Eisner. 2012. Easy-first coreference resolution. In *COLING*, pages 2519–2534.

Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. 2010. Reconcile: A coreference resolution research platform. Computer Science Technical Report, Cornell University, Ithaca, NY.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding*, pages 45–52.

Xiaofeng Yang, Jian Su, Jun Lang, Chew Lim Tan, Ting Liu, and Sheng Li. 2008. An entity-mention model for coreference resolution with inductive logic programming. In *Association of Computational Linguistics (ACL)*, pages 843–851.