Improving Coreference Resolution by Learning Entity-Level Distributed Representations

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

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

Abstract

A long-standing challenge in coreference resolution has been the incorporation of entity-level information – features defined over clusters of mentions instead of mention pairs. We present a neural network based coreference system that produces high-dimensional vector representations for pairs of coreference clusters. Using these representations, our system learns when combining clusters is desirable. We train the system with a learning-to-search algorithm that teaches it which local decisions (cluster merges) will lead to a high-scoring final coreference partition. The system substantially outperforms the current state-of-the-art on the English and Chinese portions of the CoNLL 2012 Shared Task dataset despite using few hand-engineered features.

1 Introduction

Coreference resolution, the task of identifying which mentions in a text refer to the same real-world entity, is fundamentally a clustering problem. However, many recent state-of-the-art coreference systems operate solely by linking pairs of mentions together (Durrett and Klein, 2013; Martschat and Strube, 2015; Wiseman et al., 2015).

An alternative approach is to use agglomerative clustering, treating each mention as a singleton cluster at the outset and then repeatedly merging clusters of mentions deemed to be referring to the same entity. Such systems can take advantage of entity-level information, i.e., features between clusters of mentions instead of between just two mentions. As an example for why this is useful, it is clear that the clusters {*Bill Clinton*} and {*Clinton*, *she*} are not referring to the same entity, but it is ambiguous whether the pair of mentions *Bill Clinton* and *Clinton* are coreferent.

Previous work has incorporated entity-level information through features that capture hard constraints like having gender or number agreement between clusters (Raghunathan et al., 2010; Durrett et al., 2013). In this work, we instead train a deep neural network to build distributed representations of pairs of coreference clusters. This captures entity-level information with a large number of learned, continuous features instead of a small number of hand-crafted categorical ones.

Using the cluster-pair representations, our network learns when combining two coreference clusters is desirable. At test time it builds up coreference clusters incrementally, starting with each mention in its own cluster and then merging a pair of clusters each step. It makes these decisions with a novel easy-first cluster-ranking procedure that combines the strengths of cluster-ranking (Rahman and Ng, 2011) and easy-first (Stoyanov and Eisner, 2012) coreference algorithms.

Training incremental coreference systems is challenging because the coreference decisions facing a model depend on previous decisions it has already made. We address this by using a learning-to-search algorithm inspired by SEARN (Daumé III et al., 2009) to train our neural network. This approach allows the model to learn which action (a cluster merge) available from the current state (a partially completed coreference clustering) will eventually lead to a high-scoring coreference partition.

Our system uses little manual feature engineering, which means it is easily extended to multiple languages. We evaluate our system on the English and Chinese portions of the CoNLL 2012 Shared Task dataset. The cluster-ranking model significantly outperforms a mention-ranking model that

does not use entity-level information. We also show that using an easy-first strategy improves the performance of the cluster-ranking model. Our final system achieves CoNLL F_1 scores of 65.29 for English and 63.66 for Chinese, substantially outperforming other state-of-the-art systems.[1]

2 System Architecture

Our cluster-ranking model is a single neural network that learns which coreference cluster merges are desirable. However, it is helpful to think of the network as being composed of distinct subnetworks. The *mention-pair encoder* produces distributed representations for pairs of mentions by passing relevant features through a feedforward neural network. The *cluster-pair encoder* produces distributed representations for pairs of clusters by applying a pooling operation over the representations of relevant mention pairs, i.e., pairs where one mention is in each cluster. The *cluster-ranking model* then scores pairs of clusters by passing their representations through a single neural network layer.

We also train a *mention-ranking model* that scores pairs of mentions by passing their representations through a single neural network layer. Its parameters are used to initialize the cluster-ranking model, and the scores it produces are used to prune which candidate cluster merges the cluster-ranking model considers, allowing the cluster-ranking model to run much faster. The system architecture is summarized in Figure 1.

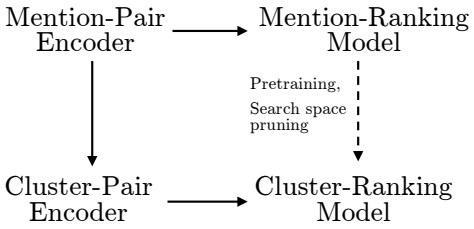

Figure 1: System architecture. Solid arrows indicate one neural network is used as a component of the other; the dashed arrow indicates other dependencies.

3 Building Representations

In this section, we describe the neural networks producing distributed representations of pairs of mentions and pairs of coreference clusters. We assume that a set of mentions has already been extracted from each document using a method such as the one in Raghunathan et al. (2010).

[1]Code and trained models are available at `https://github.com/clarkkev/deep-coref`.

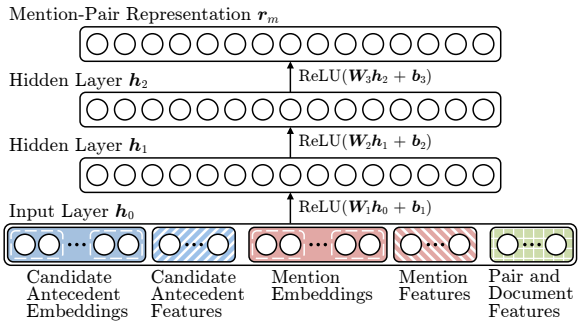

Figure 2: Mention-pair encoder.

3.1 Mention-Pair Encoder

Given a mention m and candidate antecedent a, the mention-pair encoder produces a distributed representation of the pair $\boldsymbol{r}_m(a, m) \in \mathbb{R}^d$ with a feedforward neural network, which is shown in Figure 2. The candidate antecedent may be any mention that occurs before m in the document or NA, indicating that m has no antecedent. We also experimented with models based on Long Short-Term Memory recurrent neural networks (Hochreiter and Schmidhuber, 1997), but found these to perform slightly worse when used in an end-to-end coreference system due to heavy overfitting to the training data.

Input Layer. For each mention, the model extracts various words and groups of words that are fed into the neural network. Each word is represented by a vector $\boldsymbol{w}_i \in \mathbb{R}^{d_w}$. Each group of words is represented by the average of the vectors of each word in the group. For each mention and pair of mentions, a small number of binary features and distance features are also extracted. Distances and mention lengths are binned into one of the buckets $[0, 1, 2, 3, 4, 5\text{-}7, 8\text{-}15, 16\text{-}31, 32\text{-}63, 64+]$ and then encoded in a one-hot vector in addition to being included as continuous features. The full set of features is as follows:

Embedding Features: Word embeddings of the head word, dependency parent, first word, last word, two preceding words, and two following words of the mention. Averaged word embeddings of the five preceding words, five following

words, all words in the mention, all words in the mention's sentence, and all words in the mention's document.

Additional Mention Features: The type of the mention (pronoun, nominal, proper, or list), the mention's position (index of the mention divided by the number of mentions in the document), whether the mentions is contained in another mention, and the length of the mention in words.

Document Genre: The genre of the mention's document (broadcast news, newswire, web data, etc.).

Distance Features: The distance between the mentions in sentences, the distance between the mentions in intervening mentions, and whether the mentions overlap.

Speaker Features: Whether the mentions have the same speaker and whether one mention is the other mention's speaker as determined by string matching rules from Raghunathan et al. (2010).

String Matching Features: Head match, exact string match, and partial string match.

The vectors for all of these features are concatenated to produce an I-dimensional vector $\boldsymbol{h}_0$, the input to the neural network. If $a = \text{NA}$, the features defined over mention pairs are not included. For this case, we train a separate network with an identical architecture to the pair network except for the input layer to produce anaphoricity scores.

Our set of hand-engineered features is much smaller than the dozens of complex features typically used in coreference systems. However, we found these features were crucial for getting good model performance. See Section 6.1 for a feature ablation study.

Hidden Layers. The input gets passed through three hidden layers of rectified linear (ReLU) units (Nair and Hinton, 2010). Each unit in a hidden layer is fully connected to the previous layer:

$$\boldsymbol{h}_i(a, m) = \max(0, \boldsymbol{W}_i \boldsymbol{h}_{i-1}(a, m) + \boldsymbol{b}_i)$$

where $\boldsymbol{W}_1$ is a $M_1 \times I$ weight matrix, $\boldsymbol{W}_2$ is a $M_2 \times M_1$ matrix, and $\boldsymbol{W}_3$ is a $d \times M_2$ matrix.

The output of the last hidden layer is the vector representation for the mention pair: $\boldsymbol{r}_m(a, m) = \boldsymbol{h}_3(a, m)$.

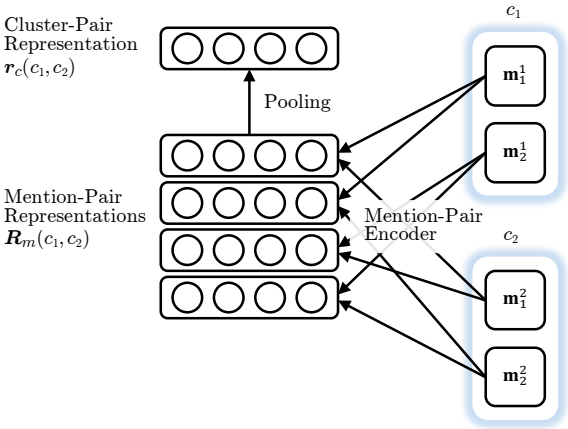

Figure 3: Cluster-pair encoder.

3.2 Cluster-Pair Encoder

Given two clusters of mentions $c_i = \{m_1^i, m_2^i, ..., m_{|c_i|}^i\}$ and $c_j = \{m_1^j, m_2^j, ..., m_{|c_j|}^j\}$, the cluster-pair encoder produces a distributed representation $\boldsymbol{r}_c(c_i, c_j) \in \mathbb{R}^{2d}$. The architecture of the encoder is summarized in Figure 3.

The cluster-pair encoder first combines the information contained in the matrix of mention-pair representations $\boldsymbol{R}_m(c_i, c_j) = [\boldsymbol{r}_m(m_1^i, m_1^j), \boldsymbol{r}_m(m_1^i, m_2^j), ..., \boldsymbol{r}_m(m_{|c_i|}^i, m_{|c_j|}^j)]$ to produce $\boldsymbol{r}_c(c_i, c_j)$. This is done by applying a pooling operation. In particular it concatenates the results of max-pooling and average-pooling, which we found to be slightly more effective than using either one alone:

$$\boldsymbol{r}_c(c_i, c_j)_k = \begin{cases} \max\{\boldsymbol{R}_m(c_i, c_j)_{k,\cdot}\} & \text{for } 0 \le k < d \\ \text{avg}\{\boldsymbol{R}_m(c_i, c_j)_{k-d,\cdot}\} & \text{for } d \le k < 2d \end{cases}$$

4 Mention-Ranking Model

Rather than training a cluster-ranking model from scratch, we first train a mention-ranking model that assigns each mention its highest scoring candidate antecedent. There are two key advantages of doing this. First, it serves as pretraining for the cluster-ranking model; in particular the mention-ranking model learns effective weights for the mention-pair encoder. Second, the scores produced by the mention-ranking model are used to provide a measure of which coreference decisions are easy (allowing for an easy-first clustering strategy) and which decisions are clearly wrong (these decisions can be pruned away, significantly reducing the search space of the cluster-ranking model).

The mention-ranking model assigns a score $s_m(a, m)$ to a mention m and candidate an-

tecedent a representing their compatibility for coreference. This is produced by applying a single fully connected layer of size one to the representation $\boldsymbol{r}_m(a, m)$ produced by the mention-pair encoder:

$$s_m(a, m) = \boldsymbol{W}_m \boldsymbol{r}_m(a, m) + b_m$$

where $\boldsymbol{W}_m$ is a $1 \times d$ weight matrix. At test time, the mention-ranking model links each mention with its highest scoring candidate antecedent.

Training Objective. We train the mention-ranking model with the slack-rescaled max-margin training objective from Wiseman et al. (2015), which encourages separation between the highest scoring true and false antecedents of the current mention. Suppose the training set consists of N mentions $m_1, m_2, ..., m_N$. Let $\mathcal{A}(m_i)$ denote the set of candidate antecedents of a mention m_i (i.e., mentions preceding m_i and NA), and $\mathcal{T}(m_i)$ denote the set of true antecedents of m_i (i.e., mentions preceding m_i that are coreferent with it or $\{\text{NA}\}$ if m_i has no antecedent). Let $\hat{t}_i$ be the highest scoring true antecedent of mention m_i:

$$\hat{t}_i = \operatorname*{argmax}_{t \in \mathcal{T}(m_i)} s_m(t, m_i)$$

Then the loss is given by

$$\sum_{i=1}^{N} \max_{a \in \mathcal{A}(m_i)} \Delta(a, m_i)(1 + s_m(a, m_i) - s_m(\hat{t}_i, m_i))$$

where $\Delta(a, m_i)$ is the mistake-specific cost function

$$\Delta(a, m_i) = \begin{cases} \alpha_{\text{FN}} & \text{if } a = \text{NA} \ \wedge \mathcal{T}(m_i) \neq \{\text{NA}\} \\ \alpha_{\text{FA}} & \text{if } a \neq \text{NA} \ \wedge \mathcal{T}(m_i) = \{\text{NA}\} \\ \alpha_{\text{WL}} & \text{if } a \neq \text{NA} \ \wedge a \notin \mathcal{T}(m_i) \\ 0 & \text{if } a \in \mathcal{T}(m_i) \end{cases}$$

for "false new," "false anaphoric," "wrong link," and correct coreference decisions. The different error penalties allow the system to be tuned for coreference evaluation metrics by biasing it towards making more or fewer coreference links.

Finding Effective Error Penalties. We fix $\alpha_{\text{WL}} = 1.0$ and search for α_{FA} and α_{FN} out of $\{0.1, 0.2, ..., 1.5\}$ with a variant of grid search. Each new trial uses the unexplored set of hyperparameters that has the closest Manhattan distance to the best setting found so far on the dev set. We stopped the search when all immediate neighbors (within 0.1 distance) of the best setting had been explored. We found $(\alpha_{\text{FN}}, \alpha_{\text{FA}}, \alpha_{\text{WL}}) = (0.8, 0.4, 1.0)$ to be best for English and $(\alpha_{\text{FN}}, \alpha_{\text{FA}}, \alpha_{\text{WL}}) = (0.7, 0.4, 1.0)$ to be best for Chinese on the CoNLL 2012 data. We attribute our smaller false new cost from the one used by Wiseman et al. (they set $\alpha_{\text{FN}} = 1.2$) to using more precise mention detection, which results in fewer links to NA.

Training Details. We initialized our word embeddings with 50 dimensional ones produced by `word2vec` (Mikolov et al., 2013) on the Gigaword corpus for English and 64 dimensional ones provided by Polyglot (Al-Rfou et al., 2013) for Chinese. Averaged word embeddings were held fixed during training while the embeddings used for single words were updated. We set our hidden layer sizes to $M_1 = 1000, M_2 = d$ = 500 and minimized the training objective using RMSProp (Hinton and Tieleman, 2012). To regularize the network, we applied L2 regularization to the model weights and dropout (Hinton et al., 2012) with a rate of 0.5 on the word embeddings and the output of each hidden layer.

Pretraining. As in Wiseman et al. (2015), we found that pretraining is crucial for the mention-ranking model's success. We pretrained the network in two stages, minimizing the following objectives from Clark and Manning (2015):

All-Pairs Classification

$$-\sum_{i=1}^{N}[\sum_{t \in \mathcal{T}(m_i)} \log p(t, m_i) + \sum_{f \in \mathcal{F}(m_i)} \log(1 - p(f, m_i))]$$

Top-Pairs Classification

$$-\sum_{i=1}^{N}[\max_{t \in \mathcal{T}(m_i)} \log p(t, m_i) + \min_{f \in \mathcal{F}(m_i)} \log(1 - p(f, m_i))]$$

Where $\mathcal{F}(m_i)$ is the set of false antecedents for m_i and $p(a, m_i) = \textit{sigmoid}(s(a, m_i))$. The top-pairs objective is a middle ground between the all-pairs classification and mention ranking objectives: it only processes high-scoring mentions, but is probabilistic rather than max-margin. We first pretrained the network with all-pairs classification for 150 epochs and then with top-pairs classification for 50 epochs. See Section 6.1 for experiments on the two-stage pretraining.

5 Cluster-Ranking Model

Although a strong coreference system on its own, the mention-ranking model has the disadvantage of only considering local information between pairs of mentions, so it cannot consolidate information at the entity-level. We address this problem by training a cluster-ranking model that scores pairs of clusters instead of pairs of mentions.

Given two clusters of mentions c_i and c_j, the cluster-ranking model produces a score $s_c(c_i, c_j)$ representing their compatibility for coreference. This is produced by applying a single fully connected layer of size one to the representation $\boldsymbol{r}_c(c_i, c_j)$ produced by the cluster-pair encoder:

$$s_c(c_i, c_j) = \boldsymbol{W}_c \boldsymbol{r}_c(c_i, c_j) + b_c$$

where $\boldsymbol{W}_c$ is a $1 \times 2d$ weight matrix. Our cluster-ranking approach also uses a measure of anaphoricity, or how likely it is for a mention m to have an antecedent. This is defined as

$$s_{\text{NA}}(m) = \boldsymbol{W}_{\text{NA}} \boldsymbol{r}_m(\text{NA}, m) + b_{\text{NA}}$$

where $\boldsymbol{W}_{\text{NA}}$ is a $1 \times d$ matrix.

5.1 Cluster-Ranking Policy Network

At test time, the cluster ranker iterates through every mention in the document, merging the current mention's cluster with a preceding one or performing no action. We view this procedure as a sequential decision process where at each step the algorithm observes the current state x and performs some action u.

Specifically, we define a state $x = (C, m)$ to consist of $C = \{c_1, c_2, ...\}$, the set of existing coreference clusters, and m, the current mention being considered. At a start state, each cluster in C contains a single mention. Let $c_m \in C$ be the cluster containing m and $\mathcal{A}(m)$ be a set of candidate antecedents for m: mentions occurring previously in the document. Then the available actions $U(x)$ from x are

- MERGE$[c_m, c]$, where c is a cluster containing a mention in $\mathcal{A}(m)$. This combines c_m and c into a single coreference cluster.
- PASS. This leaves the clustering unchanged.

After determining the new clustering C' based on the existing clustering C and action u, we consider another mention m' to get the next state $x' = (C', m')$.

Using the scoring functions s_c and s_{NA}, we define a policy network π that assigns a probability distribution over $U(x)$ as follows:

$$\pi(\text{MERGE}[c_m, c]|x) \propto e^{s_c(c_m, c)}$$
$$\pi(\text{PASS}|x) \propto e^{s_{\text{NA}}(m)}$$

During inference, π is executed by taking the highest-scoring (most probable) action at each step.

5.2 Easy-First Cluster Ranking

The last detail needed is the ordering in which to consider mentions. Cluster-ranking models in prior work order the mentions according to their positions in the document, processing them left-to-right (Rahman and Ng, 2011; Ma et al., 2014). However, we instead sort the mentions in descending order by their highest scoring candidate coreference link according to the mention-ranking model. This causes inference to occur in an easy-first fashion where hard decisions are delayed until more information is available. Easy-first orderings have been shown to improve the performance of other incremental coreference strategies (Raghunathan et al., 2010; Stoyanov and Eisner, 2012) because they reduce the problem of errors compounding as the algorithm runs.

We also find it beneficial to prune the set of candidate antecedents $\mathcal{A}(m)$ for each mention m. Rather than using all previously occurring mentions as candidate antecedents, we only include high-scoring ones, which greatly reduces the size of the search space. This allows for much faster learning and inference; we are able to remove over 95% of candidate actions with no decrease in the model's performance. For both of these two preprocessing steps, we use $s(a, m) - s(\text{NA}, m)$ as the score of a coreference link between a and m.

5.3 Deep Learning to Search

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. assumption made in machine learning. Learning-to-search algorithms are effective for this sort of problem, and have been applied successfully to coreference resolution (Daumé III and Marcu, 2005; Clark and Manning, 2015) as well as other structured prediction tasks in natural language processing (Daumé III et al., 2014;

Algorithm 1 Deep Learning to Search

for $i = 1$ **to** num_epochs **do**
  Initialize the current training set $\Gamma = \emptyset$
  for each example $(x, y) \in \mathcal{D}$ **do**
    Run the policy π to completion from start state x to obtain a trajectory of states $\{x_1, x_2, ..., x_n\}$
    for each state x_i in the trajectory **do**
      for each possible action $u \in U(x_i)$ **do**
        Execute u on x_i and then run the reference policy π^{ref} until reaching an end state e
        Assign u a cost by computing the loss on the end state: $l(u) = \mathcal{L}(e, y)$
      end for
      Add the state x_i and associated costs l to Γ
    end for
  end for
  Update π with gradient descent, minimizing $\sum_{(x,l)\in\Gamma} \sum_{u\in U(x)} \pi(u|x)l(u)$
end for

Chang et al., 2015a).

We train the cluster-ranking model using a learning-to-search algorithm inspired by SEARN (Daumé III et al., 2009), which is described in Algorithm 1. The algorithm takes as input a dataset $\mathcal{D}$ of start states x (in our case documents with each mention in its own singleton coreference cluster) and structured labels y (in our case gold coreference clusters). Its goal is to train the policy π so when it executes from x, reaching a final state e, the resulting loss $\mathcal{L}(e, y)$ is small. We use the negative of the B^3 coreference metric for this loss (Bagga and Baldwin, 1998). Although our system evaluation also includes the MUC (Vilain et al., 1995) and CEAF_{ϕ_4} (Luo, 2005) metrics, we do not incorporate them into the loss because MUC has the flaw of treating all errors equally and CEAF_{ϕ_4} is slow to compute.

For each example $(x, y) \in \mathcal{D}$, the algorithm obtains a trajectory of states $x_1, x_2, ..., x_n$ visited by the current policy by running it to completion (i.e., repeatedly taking the highest scoring action until reaching an end state) from the start state x. This exposes the model to states at train time similar to the ones it will face at test time, allowing it to learn how to cope with mistakes.

Given a state x in a trajectory, the algorithm then assigns a cost $l(u)$ to each action $u \in U(x)$ by executing the action, "rolling out" from the resulting state with a reference policy π^{ref} until reaching an end state e, and computing the resulting loss $\mathcal{L}(e, y)$. This rolling out procedure allows the model to learn how a local action will affect the final score, which cannot be otherwise computed because coreference evaluation metrics do not decompose over cluster merges. The policy network is then trained to minimize the risk associated with taking each action: $\sum_{u\in U(x)} \pi(u|x)l(u)$.

Reference policies typically refer to the gold labels to find actions that are likely to be beneficial. Our reference policy π^{ref} takes the action that increases the B^3 score the most each step, breaking ties randomly. It is generally recommended to use a stochastic mixture of the reference policy and the current learned policy during rollouts when the reference policy is not optimal (Chang et al., 2015b). However, we find only using the reference policy (which is close to optimal) to be much more efficient because it does not require neural network computations and is deterministic, which means the costs of actions can be cached.

Training details. We update π using RMSProp and apply dropout with a rate of 0.5 to the input layer. For most experiments, we initialize the mention-pair encoder component of the cluster-ranking model with the learned weights from the mention-ranking model, which we find to greatly improve performance (see Section 6.2).

Runtime. The full cluster-ranking system runs end-to-end in slightly under 1 second per document on the English test set when using a GPU (including scoring all pairs of mentions with the mention-ranking model for search-space pruning). This means the bottleneck for the overall system is the syntactic parsing required for mention detection (about 4 seconds per document on the English test set).

Model	English F_1	Chinese F_1
Full Model	65.52	64.41
– MENTION	–1.27	–0.74
– GENRE	–0.25	–2.91
– DISTANCE	–2.42	–2.41
– SPEAKER	–1.26	–0.93
– MATCHING	–2.07	–3.44

Table 1: CoNLL F_1 scores of the mention-ranking model on the dev sets without mention, document genre, distance, speaker, and string matching hand-engineered features.

6 Experiments and Results

Experimental Setup. We run experiments on the English and Chinese portions of the CoNLL 2012 Shared Task data (Pradhan et al., 2012). The models are evaluated using three of the most popular coreference metrics: MUC, B^3, and Entity-based CEAF ($CEAF_{\phi_4}$). We generally report the average F_1 score (CoNLL F_1) of the three, which is common practice in coreference evaluation. We used the most recent version of the CoNLL scorer (version 8.01), which implements the original definitions of the 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 and pleonastic *it* pronouns.

6.1 Mention-Ranking Model Experiments

Feature Ablations. We performed a feature ablation study to determine the importance of the hand-engineered features included in our model. The results are shown in Table 1. We find the small number of non-embedding features substantially improves model performance, especially the distance and string matching features. This is unsurprising, as the additional features are not easily captured by word embeddings and historically such features have been very important in coreference resolvers (Bengtson and Roth, 2008).

The Importance of Pretraining. We evaluate the benefit of the two-step pretraining for the mention-ranking model and report results in Table 2. Consistent with Wiseman et al. (2015), we find pretraining to greatly improve the model's accuracy. We note in particular that the model benefits from using both pretraining steps from Section 4, which more smoothly transitions the model from a mention-pair classification objective that is easy to optimize to a max-margin objective better suited for a ranking task.

All-Pairs	Top-Pairs	English F_1	Chinese F_1
Yes	Yes	65.52	64.41
Yes	No	–0.36	–0.24
No	Yes	–0.54	–0.33
No	No	–3.58	–5.43

Table 2: CoNLL F_1 scores of the mention-ranking model on the dev sets with different pretraining methods.

Model	English F_1	Chinese F_1
Full Model	66.01	64.86
– PRETRAINING	–5.01	–6.85
– EASY-FIRST	–0.15	–0.12
– L2S	–0.32	–0.25

Table 3: CoNLL F_1 scores of the cluster-ranking model on the dev sets with various ablations. – PRETRAINING: initializing model parameters randomly instead of from the mention-ranking model, – EASY-FIRST: iterating through mentions in order of occurrence instead of according to their highest scoring candidate coreference link, – L2S: training on a fixed trajectory of correct actions instead of using learning to search.

6.2 Cluster-Ranking Model Experiments

We evaluate the importance of three key details of the cluster ranker: initializing it with the mention-ranking model's weights, using an easy-first ordering of mentions, and using learning to search. The results are shown in Table 3.

Pretrained Weights. We compare initializing the cluster-ranking model randomly with initializing it with the weights learned by the mention-ranking model. Using pretrained weights greatly improves performance. We believe the cluster-ranking model has difficulty learning effective weights from scratch due to noise in the signal coming from cluster-level decisions (an overall bad cluster merge may still involve a few cor-

rect pairwise links) and the smaller amount of data used to train the cluster-ranking model (many possible actions are pruned away during preprocessing). We believe the score would be even lower without search-space pruning, which stops the model from considering many bad actions.

Easy-First Cluster Ranking. We compare the effectiveness of easy-first cluster-ranking with the commonly used left-to-right approach. Using a left-to-right strategy simply requires changing the preprocessing step ordering the mentions so mentions are sorted by their position in the document instead of their highest scoring coreference link according to the mention-ranking model. We find the easy-first approach slightly outperforms using a left-to-right ordering of mentions. We believe this is because delaying hard decisions until later reduces the problem of early mistakes causing later decisions to be made incorrectly.

Learning to Search. We also compare learning to search with the simpler approach of training the model on a trajectory of gold coreference decisions (i.e., training on a fixed cost-sensitive classification dataset). Using this approach significantly decreases performance. We attribute this to the model not learning how to deal with mistakes when it only sees correct decisions during training.

6.3 Capturing Semantic Similarity

Using semantic information to improve coreference accuracy has had mixed in results in previous research, and has been called an "uphill battle" in coreference resolution (Durrett and Klein, 2013). However, word embeddings are well known for being effective at capturing semantic relatedness, and we show here that neural network coreference models can take advantage of this.

Perhaps the case where semantic similarity is most important is in linking nominals with no head match (e.g., "the nation" and "the country"). We compare the performance of our neural network model with our earlier statistical system (Clark and Manning, 2015) at classifying mention pairs of this type as being coreferent or not. The neural network shows substantial improvement (18.9 F_1 vs. 10.7 F_1) on this task compared to the more modest improvement it gets at classifying any pair of mentions as coreferent (68.7 F_1 vs. 66.1 F_1). Some example wins are shown in Table 4. These types of coreference links are quite rare in the CoNLL data (about 1.2% of the positive coreference links in the test set), so the improvement does not significantly contribute to the final system's score, but it does suggest progress on this difficult type of coreference problem.

Antecedent	Anaphor
the country's leftist rebels	the guerrillas
the company	the New York firm
the suicide bombing	the attack
the gun	the rifle
the U.S. carrier	the ship

Table 4: Examples of nominal coreferences with no head match that the neural model gets correct, but the system from Clark and Manning (2015) gets incorrect.

6.4 Final System Performance

In Table 5 we compare the results of our system with state-of-the-art approaches for English and Chinese. Our mention-ranking model surpasses all previous systems. We attribute its improvement over the neural mention ranker from Wiseman et al. (2015) to our model using a deeper neural network, pretrained word embeddings, and more sophisticated pretraining.

The cluster-ranking model improves results further across both languages and all evaluation metrics, demonstrating the utility of incorporating entity-level information. The improvement is largest in CEAF_{ϕ_4}, which is encouraging because CEAF_{ϕ_4} is the most recently proposed metric, designed to correct flaws in the other two (Luo, 2005). We believe entity-level information is particularly useful for preventing bad merges between large clusters (see Figure 4 for an example). However, it is worth noting that in practice the much more complicated cluster-ranking model brings only fairly modest gains in performance.

7 Related Work

There has been extensive work on machine learning approaches to coreference resolution (Soon et al., 2001; Ng and Cardie, 2002), with mention-ranking models being particularly popular (Denis and Baldridge, 2007; Durrett and Klein, 2013; Martschat and Strube, 2015).

We train a neural mention-ranking model inspired by Wiseman et al. (2015) as a starting point, but then use it to pretrain a cluster-ranking model that benefits from entity-level information. Wise-

	MUC			B^3			$CEAF_{\phi_4}$			
	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Avg. F_1
CoNLL 2012 English Test Data										
Clark and Manning (2015)	76.12	69.38	72.59	65.64	56.01	60.44	59.44	52.98	56.02	63.02
Peng et al. (2015)	–	–	72.22	–	–	60.50	–	–	56.37	63.03
Wiseman et al. (2015)	76.23	69.31	72.60	66.07	55.83	60.52	59.41	54.88	57.05	63.39
Wiseman et al. (2016)	77.49	69.75	73.42	66.83	56.95	61.50	62.14	53.85	57.70	64.21
NN Mention Ranker	79.77	69.10	74.05	69.68	56.37	62.32	63.02	53.59	57.92	64.76
NN Cluster Ranker	78.93	69.75	**74.06**	70.08	56.98	**62.86**	62.48	55.82	**58.96**	**65.29**
CoNLL 2012 Chinese Test Data										
Chen & Ng (2012)	64.69	59.92	62.21	60.26	51.76	55.69	51.61	58.84	54.99	57.63
Björkelund & Kuhn (2014)	69.39	62.57	65.80	61.64	53.87	57.49	59.33	54.65	56.89	60.06
NN Mention Ranker	72.53	65.72	68.96	65.49	56.87	60.88	61.93	57.11	59.42	63.09
NN Cluster Ranker	73.85	65.42	**69.38**	67.53	56.41	**61.47**	62.84	57.62	**60.12**	**63.66**

Table 5: Comparison with the current state-of-the-art approaches on the CoNLL 2012 test sets. NN Mention Ranker and NN Cluster Ranker are contributions of this work.

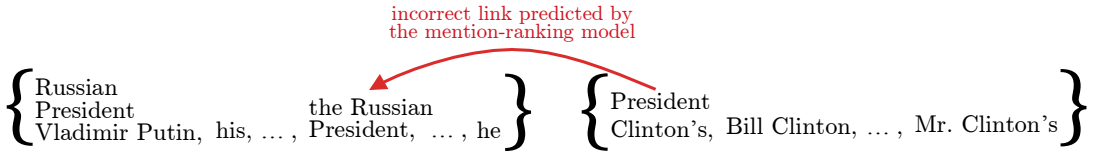

Figure 4: Thanks to entity-level information, the cluster-ranking model correctly declines to merge these two large clusters when running on the test set. However, the mention-ranking model incorrectly links *the Russian President* and *President Clinton's*, which greatly reduces the final precision score.

man et al. (2016) extend their mention-ranking model by incorporating entity-level information produced by a recurrent neural network running over the candidate antecedent-cluster. However, this is an augmentation to a mention-ranking model, and not fundamentally a clustering model as our cluster ranker is.

Entity-level information has also been incorporated in coreference systems using joint inference (McCallum and Wellner, 2003; Poon and Domingos, 2008; Haghighi and Klein, 2010) and systems that build up coreference clusters incrementally (Luo et al., 2004; Yang et al., 2008; Raghunathan et al., 2010). We take the latter approach, and in particular combine the cluster-ranking (Rahman and Ng, 2011; Ma et al., 2014) and easy-first (Stoyanov and Eisner, 2012; Clark and Manning, 2015) clustering strategies. These prior systems all express entity-level information in the form of hand-engineered features and constraints instead of entity-level distributed representations that are learned from data.

We train our system using a learning-to-search algorithm similar to SEARN (Daumé III et al., 2009). Learning-to-search style algorithms have 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), Ma et al. (2014), and Clark and Manning (2015). Other works use structured perceptron models for the same purpose (Stoyanov and Eisner, 2012; Fernandes et al., 2012; Björkelund and Kuhn, 2014).

8 Conclusion

We have presented a coreference system that captures entity-level information with distributed representations of coreference cluster pairs. These learned, dense, high-dimensional feature vectors provide our cluster-ranking coreference model with a strong ability to distinguish beneficial cluster merges from harmful ones. The model is trained with a learning-to-search algorithm that allows it to learn how local decisions will affect the final coreference score. We evaluate our system on the English and Chinese portions of the CoNLL 2012 Shared Task and report a substantial improvement over the current state-of-the-art.

Acknowledgments

We thank Will Hamilton, Jon Gauthier, and the anonymous reviewers for their thoughtful comments and suggestions. This work was supported by NSF Award IIS-1514268. 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

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. *Conference on Natural Language Learning (CoNLL)*, pages 183–192.

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.

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 Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Association of Computational Linguistics (ACL)*, pages 47–57.

Kai-Wei Chang, He He, Hal Daumé III, and John Langford. 2015a. Learning to search for dependencies. *arXiv preprint arXiv:1503.05615*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015b. Learning to search better than your teacher. In *International Conference on Machine Learning (ICML)*.

Chen Chen and Vincent Ng. 2012. Combining the best of two worlds: A hybrid approach to multilingual 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 56–63.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Association for Computational Linguistics (ACL)*.

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.

Hal Daumé III, John Langford, and Stephane Ross. 2014. Efficient programmable learning to search. *arXiv preprint arXiv:1406.1837*.

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

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, David Leo Wright Hall, and Dan Klein. 2013. Decentralized entity-level modeling for coreference resolution. In *Association for Computational Linguistics (ACL)*, pages 114–124.

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.

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.

Geoffrey Hinton and Tijmen Tieleman. 2012. Lecture 6.5-RmsProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning, 4*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

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)*.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics (TACL)*, 3:405–418.

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*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 807–814.

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

Haoruo Peng, Kai-Wei Chang, and Dan Roth. 2015. A joint framework for coreference resolution and mention head detection. *Conference on Natural Language Learning (CoNLL)*, 51:12.

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

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 *International Conference on Computational Linguistics (COLING)*, pages 2519–2534.

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.

Sam Wiseman, Alexander M. Rush, Stuart M. Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Association of Computational Linguistics (ACL)*, pages 92–100.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*.

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.