Deep Reinforcement Learning for Mention-Ranking Coreference Models

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

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

Abstract
Coreference resolution systems are typically trained with heuristic loss functions that require careful tuning. In this paper we instead apply reinforcement learning to directly optimize a neural mention-ranking model for coreference evaluation metrics. We experiment with two approaches: the REINFORCE policy gradient algorithm and a reward-rescaled max-margin objective. We find the latter to be more effective, resulting in significant improvements over the current state-of-the-art on the English and Chinese portions of the CoNLL 2012 Shared Task.

1 Introduction
Coreference resolution systems typically operate by making sequences of local decisions (e.g., adding a coreference link between two mentions). However, most measures of coreference resolution performance do not decompose over local decisions, which means the utility of a particular decision is not known until all other decisions have been made.

Due to this difficulty, coreference systems are usually trained with loss functions that heuristically define the goodness of a particular coreference decision. These losses contain hyperparameters that are carefully selected to ensure the model performs well according to coreference evaluation metrics. This complicates training, especially across different languages and datasets where systems may work best with different settings of the hyperparameters.

To address this, we explore using two variants of reinforcement learning to directly optimize a coreference system for coreference evaluation metrics. In particular, we modify the max-margin coreference objective proposed by Wiseman et al. (2015) by incorporating the reward associated with each coreference decision into the loss’s slack rescaling. We also test the REINFORCE policy gradient algorithm (Williams, 1992).

Our model is a neural mention-ranking model. Mention-ranking models score pairs of mentions for their likelihood of coreference rather than comparing partial coreference clusters. Hence they operate in a simple setting where coreference decisions are made independently. Although they are less expressive than entity-centric approaches to coreference (e.g., Haghighi and Klein, 2010), mention-ranking models are fast, scalable, and simple to train, causing them to be the dominant approach to coreference in recent years (Durrett and Klein, 2013; Wiseman et al., 2015). Having independent actions is particularly useful when applying reinforcement learning because it means a particular action’s effect on the final reward can be computed efficiently.

We evaluate the models on the English and Chinese portions of the CoNLL 2012 Shared Task. The REINFORCE algorithm is competitive with a heuristic loss function while the reward-rescaled objective significantly outperforms both1. We attribute this to reward rescaling being well suited for a ranking task due to its max-margin loss as well as benefiting from directly optimizing for coreference metrics. Error analysis shows that using the reward-rescaling loss results in a similar number of mistakes as the heuristic loss, but the mistakes tend to be less severe.

1Code and trained models are available at https://github.com/clarkkev/deep-coref.
2 Neural Mention-Ranking Model

We use the neural mention-ranking model described in Clark and Manning (2016), which we briefly go over in this section. Given a mention \( m \) and candidate antecedent \( c \), the mention-ranking model produces a score for the pair \( s(c, m) \) indicating their compatibility for coreference with a feedforward neural network. The candidate antecedent may be any mention that occurs before \( m \) in the document or NA, indicating that \( m \) has no antecedent.

**Input Layer.** For each mention, the model extracts various words (e.g., the mention’s head word) and groups of words (e.g., all words in the mention’s sentence) that are fed into the neural network. Each word is represented by a vector \( w_i \in \mathbb{R}^{d_w} \). Each group of words is represented by the average of the vectors of each word in the group. In addition to the embeddings, a small number of additional features are used, including distance, string matching, and speaker identification features. See Clark and Manning (2016) for the full set of features and an ablation study.

These features are concatenated to produce an \( I \)-dimensional vector \( h_0 \), the input to the neural network. If \( c = \text{NA} \), features defined over pairs of mentions 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.

**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:

\[
h_i(c, m) = \max(0, W_i h_{i-1}(c, m) + b_i)
\]

where \( W_1 \) is a \( M_1 \times I \) weight matrix, \( W_2 \) is a \( M_2 \times M_1 \) matrix, and \( W_3 \) is a \( M_3 \times M_2 \) matrix.

**Scoring Layer.** The final layer is a fully connected layer of size 1:

\[
s(c, m) = W_4 h_3(c, m) + b_4
\]

where \( W_4 \) is a \( 1 \times M_3 \) weight matrix. At test time, the mention-ranking model links each mention with its highest scoring candidate antecedent.

3 Learning Algorithms

Mention-ranking models are typically trained with heuristic loss functions that are tuned via hyperparameters. These hyperparameters are usually given as costs for different error types, which are used to bias the coreference system towards making more or fewer coreference links.

In this section we first describe a heuristic loss function incorporating this idea from Wiseman et al. (2015). We then propose new training procedures based on reinforcement learning that instead directly optimize for coreference evaluation metrics.

3.1 Heuristic Max-Margin Objective

The heuristic loss from Wiseman et al. is governed by the following error types, which were first proposed by Durrett et al. (2013).

Suppose the training set consists of \( N \) mentions \( m_1, m_2, ..., m_N \). Let \( C(m_i) \) denote the set of candidate antecedents of a mention \( m_i \) (i.e., mentions preceding \( m_i \) and NA) and \( T(m_i) \) denote the set of true antecedents of \( m_i \) (i.e., mentions preceding \( m_i \) that are coreferent with it or \{NA\} if \( m_i \) has no antecedent). Then we define the following costs for linking \( m_i \) to a candidate antecedent \( c \in C(m_i) \):

\[
\Delta_h(c, m_i) = \begin{cases} 
\alpha_{\text{FN}} & \text{if } c = \text{NA} \land T(m_i) \neq \{\text{NA}\} \\
\alpha_{\text{FA}} & \text{if } c \neq \text{NA} \land T(m_i) = \{\text{NA}\} \\
\alpha_{\text{WL}} & \text{if } c \neq \text{NA} \land c \notin T(m_i) \\
0 & \text{if } c \in T(m_i)
\end{cases}
\]

for “false new,” “false anaphor,” “wrong link”, and correct coreference decisions.

The heuristic loss is a slack-rescaled max-margin objective parameterized by these error costs. Let \( \hat{t}_i \) be the highest scoring true antecedent of \( m_i \):

\[
\hat{t}_i = \arg\max_{c \in C(m_i) \land \Delta_h(c, m_i) = 0} s(c, m_i)
\]

Then the heuristic loss is given as

\[
\mathcal{L}(\theta) = \sum_{i=1}^{N} \max_{c \in C(m_i)} \Delta_h(c, m_i)(1 + s(c, m_i) - s(\hat{t}_i, m_i))
\]

**Finding Effective Error Penalties.** We fix \( \alpha_{\text{WL}} = 1.0 \) and search for \( \alpha_{\text{FA}} \) and \( \alpha_{\text{FN}} \) out of \{0.1, 0.2, ..., 1.5\} with a variant of grid search. Each new trial uses the unexplored set of hyperparamet-
ters that has the closest Manhattan distance to the best setting found so far on the dev set. The search is halted when all immediate neighbors (within 0.1 distance) of the best setting have 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 \((\alpha_{\text{FN}}, \alpha_{\text{FA}}, \alpha_{\text{WL}}) = (0.8, 0.5, 1.0)\) to be best for Chinese on the CoNLL 2012 data.

3.2 Reinforcement Learning

Finding the best hyperparameter settings for the heuristic loss requires training many variants of the model, and at best results in an objective that is correlated with coreference evaluation metrics. To address this, we pose mention ranking in the reinforcement learning framework (Sutton and Barto, 1998) and propose methods that directly optimize the model for coreference metrics.

We can view the mention-ranking model as an agent taking a series of actions \(a_{1:T} = a_1, a_2, \ldots, a_T\), where \(T\) is the number of mentions in the current document. Each action \(a_i\) links the \(i\)th mention in the document \(m_i\) to a candidate antecedent. Formally, we denote the set of actions available for the \(i\)th mention as \(A_i = \{(c, m_i) : c \in \mathcal{C}(m_i)\}\), where an action \((c, m)\) adds a coreference link between mentions \(c\) and \(m\). The mention-ranking model assigns each action the score \(s(c, m)\) and takes the highest-scoring action at each step.

Once the agent has executed a sequence of actions, it observes a reward \(R(a_{1:T})\), which can be any function. We use the \(B^3\) coreference metric for this reward (Bagga and Baldwin, 1998). Although our system evaluation also includes the MUC (Vilain et al., 1995) and CEAF_{\phi_4} (Luo, 2005) metrics, we do not incorporate them into the loss because MUC has the flaw of treating all errors equally and CEAF_{\phi_4} is slow to compute.

**Reward Rescaling.** Crucially, the actions taken by a mention-ranking model are independent. This means it is possible to change any action \(a_i\) to a different one \(a_i' \in A_i\) and see what reward the model would have gotten by taking that action instead: \(R(a_1, \ldots, a_{i-1}, a_i', a_{i+1}, \ldots, a_T)\). We use this idea to improve the slack-rescaling parameter \(\Delta\) in the max-margin loss \(L(\theta)\). Instead of setting its value based on the error type, we compute exactly how much each action hurts the final reward:

\[
\Delta(c, m_i) = -R(a_1, \ldots, (c, m_i), \ldots, a_T) + \max_{a_i' \in A_i} R(a_1, \ldots, a_i', \ldots, a_T)
\]

where \(a_1:T\) is the highest scoring sequence of actions according to the model’s current parameters. Otherwise the model is trained in the same way as with the heuristic loss.

**The REINFORCE Algorithm.** We also explore using the REINFORCE policy gradient algorithm (Williams, 1992). We can define a probability distribution over actions using the mention-ranking model’s scoring function as follows:

\[
p(\theta)(a) \propto e^{s(c, m)}
\]

for any action \(a = (c, m)\). The REINFORCE algorithm seeks to maximize the expected reward

\[
J(\theta) = \mathbb{E}_{a_{1:T} \sim p_\theta} R(a_{1:T})
\]

It does this through gradient ascent. Computing the full gradient is prohibitive because of the expectation over all possible action sequences, which is exponential in the length of the sequence. Instead, it gets an unbiased estimate of the gradient by sampling a sequence of actions \(a_{1:T}\) according to \(p_\theta\) and computing the gradient only over the sample.

We take advantage of the independence of actions by using the following gradient estimate, which has lower variance than the standard REINFORCE gradient estimate:

\[
\nabla_\theta J(\theta) \approx \sum_{i=1}^{T} \sum_{a_i' \in A_i} \left[ \nabla_\theta p_\theta(a_i') \right] (R(a_1, \ldots, a_i', \ldots, a_T) - b_i)
\]

where \(b_i\) is a baseline used to reduce the variance, which we set to \(\mathbb{E}_{a_i' \in A_i \sim p_\theta} R(a_1, \ldots, a_i', \ldots, a_T)\).

4 Experiments and Results

We run experiments on the English and Chinese portions of the CoNLL 2012 Shared Task data (Pradhan et al., 2012) and evaluate with the MUC, \(B^3\), and CEAF_{\phi_4} metrics. Our experiments were run using predicted mentions from Stanford’s rule-based coreference system (Raghunathan et al., 2010).

We follow the training methodology from Clark and Manning (2016): hidden layers of sizes \(M_1 = 1000\), \(M_2 = M_3 = 500\), the RMSprop optimizer.
Table 1: Comparison of the methods together with other state-of-the-art approaches on the test sets.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CoNLL 2012 English Test Data</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wiseman et al. (2016)</td>
<td>77.49</td>
<td>69.75</td>
<td>73.42</td>
<td>66.83</td>
<td>56.95</td>
<td>61.50</td>
<td>62.14</td>
<td>53.85</td>
<td>57.70</td>
<td>64.21</td>
</tr>
<tr>
<td>Clark &amp; Manning (2016)</td>
<td>79.91</td>
<td>69.30</td>
<td>74.23</td>
<td>71.01</td>
<td>56.53</td>
<td>62.95</td>
<td>63.84</td>
<td>54.33</td>
<td>58.70</td>
<td>65.29</td>
</tr>
<tr>
<td>Heuristic Loss</td>
<td>79.63</td>
<td>70.25</td>
<td><strong>74.65</strong></td>
<td>69.21</td>
<td>57.87</td>
<td>63.40</td>
<td>63.62</td>
<td>53.97</td>
<td>58.40</td>
<td>65.36</td>
</tr>
<tr>
<td>REINFORCE</td>
<td>80.08</td>
<td>69.61</td>
<td>74.48</td>
<td>70.70</td>
<td>56.96</td>
<td>63.09</td>
<td>63.59</td>
<td>54.46</td>
<td>58.67</td>
<td>65.41</td>
</tr>
<tr>
<td>Reward Rescaling</td>
<td>79.19</td>
<td>70.44</td>
<td>74.56</td>
<td>69.93</td>
<td>57.99</td>
<td><strong>63.40</strong></td>
<td>63.46</td>
<td>55.52</td>
<td><strong>59.23</strong></td>
<td><strong>65.73</strong></td>
</tr>
<tr>
<td><strong>CoNLL 2012 Chinese Test Data</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Björkelund &amp; Kuhn (2014)</td>
<td>69.39</td>
<td>62.57</td>
<td>65.80</td>
<td>61.64</td>
<td>53.87</td>
<td>61.47</td>
<td>59.33</td>
<td>54.65</td>
<td>61.26</td>
<td>60.06</td>
</tr>
<tr>
<td>Clark &amp; Manning (2016)</td>
<td>73.85</td>
<td>65.42</td>
<td>69.38</td>
<td>67.53</td>
<td>56.41</td>
<td>61.47</td>
<td>62.84</td>
<td>57.62</td>
<td>60.12</td>
<td>63.66</td>
</tr>
<tr>
<td>Heuristic Loss</td>
<td>72.20</td>
<td>66.51</td>
<td>69.24</td>
<td>64.71</td>
<td>58.16</td>
<td>61.26</td>
<td>61.98</td>
<td>58.41</td>
<td>60.14</td>
<td>63.54</td>
</tr>
<tr>
<td>REINFORCE</td>
<td>74.05</td>
<td>65.38</td>
<td><strong>69.44</strong></td>
<td>67.52</td>
<td>56.43</td>
<td>61.48</td>
<td>62.38</td>
<td>57.77</td>
<td>59.98</td>
<td>63.64</td>
</tr>
<tr>
<td>Reward Rescaling</td>
<td>73.64</td>
<td>65.62</td>
<td>69.40</td>
<td>67.48</td>
<td>56.94</td>
<td><strong>61.76</strong></td>
<td>62.46</td>
<td>58.60</td>
<td><strong>60.47</strong></td>
<td><strong>63.88</strong></td>
</tr>
</tbody>
</table>

4.1 Results

We compare the heuristic loss, REINFORCE, and reward rescaling approaches on both datasets. We find that REINFORCE does slightly better than the heuristic loss, but reward rescaling performs significantly better than both on both languages.

We attribute the modest improvement of REINFORCE to it being poorly suited for a ranking task. During training it optimizes the model’s performance in expectation, but at test-time it takes the most probable sequence of actions. This mismatch occurs even at the level of an individual decision: the model only links the current mention to a single antecedent, but is trained to assign high probability to all correct antecedents. We believe the benefit of REINFORCE being guided by coreference evaluation metrics is offset by this disadvantage, which does not occur in the max-margin approaches. The reward-rescaled max-margin loss combines the best of both worlds, resulting in superior performance.

4.2 The Benefits of Reinforcement Learning

In this section we examine the reward-based cost function $\Delta_r$ and perform error analysis to determine how reward rescaling improves the mention-ranking model’s accuracy.

Comparison with Heuristic Costs. We compare the reward-based cost function $\Delta_r$ with the error types used in the heuristic loss. For English, the average value of $\Delta_r$ is 0.79 for FN errors and 0.38 for FA errors when the costs are scaled so the average value of a WL error is 1.0. These are very close to the hyperparameter values $(\alpha_{\text{FN}}, \alpha_{\text{FA}}, \alpha_{\text{WL}}) = (0.8, 0.4, 1.0)$ found by grid search. However, there is a high variance in costs for each error type, suggesting that using a fixed penalty for each type as in the heuristic loss is insufficient (see Figure 1).

Avoiding Costly Mistakes. Embedding the costs of actions into the loss function causes the reward-rescaling model to prioritize getting the more important coreference decisions (i.e., the ones with the biggest impact on the final score) correct. As a
Table 3: Examples of classes of mention on which the reward-rescaling loss improves upon the heuristic loss due to its reward-based cost function. Reported numbers are from the English CoNLL 2012 test set.

<table>
<thead>
<tr>
<th>Mention Type</th>
<th>Average Cost $\Delta_x$</th>
<th># Heuristic Loss Errors</th>
<th># Reward Rescaling Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FN FA WL</td>
<td>FN FA WL</td>
<td>FN FA WL</td>
</tr>
<tr>
<td>Proper nouns</td>
<td>0.90 0.38 1.02</td>
<td>403 597 221</td>
<td>334 660 233</td>
</tr>
<tr>
<td>Pronouns in phone conversations</td>
<td>0.86 0.39 1.21</td>
<td>82 85 81</td>
<td>90 78 67</td>
</tr>
</tbody>
</table>

Table 2: Number of “false new,” “false anaphoric,” and “wrong link” errors produced by the models on the English CoNLL 2012 test set.

Table: Model FN FA WL
Heuristic Loss 1719 1956 1258
Reward Rescaling 1725 1994 1247

5 Related Work

Mention-ranking models have been widely used for coreference resolution (Denis and Baldridge, 2007; Rahman and Ng, 2009; Durrett and Klein, 2013). These models are typically trained with heuristic loss functions that assign costs to different error types, as in the heuristic loss we describe in Section 3.1 (Fernandes et al., 2012; Durrett et al., 2013; Björkelund and Kuhn, 2014; Wiseman et al., 2015; Martschat and Strube, 2015; Wiseman et al., 2016).

To the best of our knowledge reinforcement learning has not been applied to coreference resolution before. However, imitation learning algorithms such as SEARN (Daumé III et al., 2009) have been used to train coreference resolvers (Daumé III, 2006; Ma et al., 2014; Clark and Manning, 2015). These algorithms also directly optimize for coreference evaluation metrics, but they require an expert policy to learn from instead of relying on rewards alone.

6 Conclusion

We propose using reinforcement learning to directly optimize mention-ranking models for coreference evaluation metrics, obviating the need for hyperparameters that must be carefully selected for each particular language, dataset, and evaluation metric. Our reward-rescaling approach also increases the model’s accuracy, resulting in significant gains over the current state-of-the-art.

Acknowledgments

We thank Kelvin Guu, William Hamilton, Will Monroe, 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


Pascal Denis and Jason Baldridge. 2007. A ranking approach to pronoun resolution. In International Joint Conferences on Artificial Intelligence (IJCAI).


