Learning Structured Perceptrons for Coreference Resolution with Latent Antecedents and Non-local Features

Anders Björkelund and **Jonas Kuhn**
Institute for Natural Language Processing
University of Stuttgart
{anders,jonas}@ims.uni-stuttgart.de

Abstract

We investigate different ways of learning structured perceptron models for coreference resolution when using non-local features and beam search. Our experimental results indicate that standard techniques such as early updates or Learning as Search Optimization (LaSO) perform worse than a greedy baseline that only uses local features. By modifying LaSO to delay updates until the end of each instance we obtain significant improvements over the baseline. Our model obtains the best results to date on recent shared task data for Arabic, Chinese, and English.

1 Introduction

This paper studies and extends previous work using the structured perceptron (Collins, 2002) for complex NLP tasks. We show that for the task of coreference resolution the straightforward combination of beam search and early update (Collins and Roark, 2004) falls short of more limited feature sets that allow for exact search. This contrasts with previous work on, e.g., syntactic parsing (Collins and Roark, 2004; Huang, 2008; Zhang and Clark, 2008) and linearization (Bohnet et al., 2011), and even simpler structured prediction problems, where early updates are not even necessary, such as part-of-speech tagging (Collins, 2002) and named entity recognition (Ratinov and Roth, 2009).

The main reason why early updates underperform in our setting is that the task is too difficult and that the learning algorithm is not able to profit from all training data. Put another way, early updates happen too early, and the learning algorithm rarely reaches the end of the instances as it halts, updates, and moves on to the next instance.

An alternative would be to continue decoding the same instance after the early updates, which is equivalent to *Learning as Search Optimization* (LaSO; Daumé III and Marcu (2005b)). The learning task we are tackling is however further complicated since the target structure is under-determined by the gold standard annotation. Coreferent mentions in a document are usually annotated as sets of mentions, where all mentions in a set are coreferent. We adopt the recently popularized approach of inducing a latent structure within these sets (Fernandes et al., 2012; Chang et al., 2013; Durrett and Klein, 2013). This approach provides a powerful boost to the performance of coreference resolvers, but we find that it does not combine well with the LaSO learning strategy. We therefore propose a modification to LaSO, which delays updates until after each instance. The combination of this modification with non-local features leads to further improvements in the clustering accuracy, as we show in evaluation results on all languages from the CoNLL 2012 Shared Task – Arabic, Chinese, and English. We obtain the best results to date on these data sets.[1]

2 Background

Coreference resolution is the task of grouping referring expressions (or *mentions*) in a text into disjoint *clusters* such that all mentions in a cluster refer to the same entity. An example is given in Figure 1 below, where mentions from two clusters are marked with brackets:

> [Drug Emporium Inc.]$_{a_1}$ said [Gary Wilber]$_{b_1}$ was named CEO of [this drugstore chain]$_{a_2}$. [He]$_{b_2}$ succeeds his father, Philip T. Wilber, who founded [the company]$_{a_3}$ and remains chairman. Robert E. Lyons III, who headed the [company]$_{a_4}$'s Philadelphia region, was appointed president and chief operating officer, succeeding [Gary Wilber]$_{b_3}$.

Figure 1: An excerpt of a document with the mentions from two clusters marked.

[1] Our system is available at http://www.ims.uni-stuttgart.de/~anders/coref.html

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 47–57,
Baltimore, Maryland, USA, June 23-25 2014.

In recent years much work on coreference resolution has been devoted to increasing the expressivity of the classical *mention-pair model*, in which each coreference classification decision is limited to information about two mentions that make up a pair. This shortcoming has been addressed by *entity-mention models*, which relate a candidate mention to the full cluster of mentions predicted to be coreferent so far (for more discussion on the model types, see, e.g., (Ng, 2010)).

Nevertheless, the two best systems in the latest CoNLL Shared Task on coreference resolution (Pradhan et al., 2012) were both variants of the mention-pair model. While the second best system (Björkelund and Farkas, 2012) followed the widely used baseline of Soon et al. (2001), the winning system (Fernandes et al., 2012) proposed the use of a **tree representation**.

The tree-based model of Fernandes et al. (2012) construes the representation of coreference clusters as a rooted tree. Figure 2 displays an example tree over the clusters from Figure 1. Every mention corresponds to a node in the tree, and arcs between mentions indicate that they are coreferent. The tree additionally has a dummy root node. Every subtree under the root node corresponds to a cluster of coreferent mentions.

Since coreference training data is typically not annotated with trees, Fernandes et al. (2012) proposed the use of **latent** trees that are induced during the training phase of a coreference resolver. The latent tree provides more meaningful antecedents for training.[2] For instance, the popular pair-wise instance creation method suggested by Soon et al. (2001) assumes non-branching trees, where the antecedent of every mention is its linear predecessor (i.e., *he*$_{b_2}$ is the antecedent of *Gary Wilber*$_{b_3}$). Comparing the two alternative antecedents of *Gary Wilber*$_{b_3}$, the tree in Figure 2 provides a more reliable basis for training a coreference resolver, as the two mentions of *Gary Wilber* are both proper names and have an exact string match.

3 Representation and Learning

Let $M = \{m_0, m_1, ..., m_n\}$ denote the set of mentions in a document, including the artificial root mention (denoted by m_0). We assume that the mentions are ordered ascendingly with respect to the linear order of the document, where the document root precedes all other mentions.[3] For each mention m_j, let A_j denote the set of potential antecedents. That is, the set of all mentions that precede m_j according to the linear order including the root node, or, $A_j = \{m_i \mid i < j\}$. Finally, let $\mathcal{A}$ denote the set of all antecedent sets $\{A_0, A_1, ..., A_n\}$.

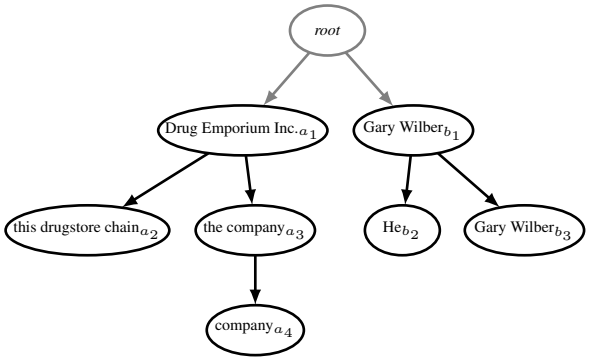

Figure 2: A tree representation of Figure 1.

In the tree model, each mention corresponds to a node, and an antecedent-anaphor pair $\langle a_i, m_i \rangle$, where $a_i \in A_i$, corresponds to a directed edge (or *arc*) pointing from antecedent to anaphor.

The score of an arc $\langle a_i, m_i \rangle$ is defined as the scalar product between a weight vector w and a feature vector $\Phi(\langle a_i, m_i \rangle)$, where Φ is a feature extraction function over an arc (thus extracting features from the antecedent and the anaphor). The score of a coreference tree $y = \{\langle a_1, m_1 \rangle, \langle a_2, m_2 \rangle, ..., \langle a_n, m_n \rangle\}$ is defined as the sum of the scores of all the mention pairs:

$$score(\langle a_i, m_i \rangle) = w \cdot \Phi(\langle a_i, m_i \rangle) \quad (1)$$
$$score(y) = \sum_{\langle a_i, m_i \rangle \in y} score(\langle a_i, m_i \rangle)$$

The objective is to find the output $\hat{y}$ that maximizes the scoring function:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}(\mathcal{A})} score(y) \quad (2)$$

where $\mathcal{Y}(\mathcal{A})$ denotes the set of possible trees given the antecedent sets $\mathcal{A}$. By treating the mentions as nodes in a directed graph and assigning scores to the arcs according to (1), Fernandes et al. (2012) solved the search problem using the Chu-Liu-Edmonds (CLE) algorithm (Chu and Liu, 1965;

[2] We follow standard practice and overload the terms anaphor and antecedent to be any type of mention, i.e., names as well as pronouns. An antecedent is simply the mention to the left of the anaphor.

[3] We impose a total order on mentions. In case of nested mentions, the mention that begins first is assumed to precede the embedded one. If two mentions begin at the same token, the longer one is taken to precede the shorter one.

Edmonds, 1967), which is a maximum spanning tree algorithm that finds the optimal tree over a connected directed graph. CLE, however, has the drawback that the scores of the arcs must remain fixed and can not change depending on other arcs and it is not clear how to include non-local features in a CLE decoder.

3.1 Online learning

We find the weight vector w by online learning using a variant of the structured perceptron (Collins, 2002). Specifically, we use the passive-aggressive (PA) algorithm (Crammer et al., 2006), since we found that this performed slightly better in preliminary experiments.[4]

The structured perceptron iterates over training instances $\langle x_i, y_i \rangle$, where x_i are inputs and y_i are outputs. For each instance it uses the current weight vector w to make a prediction $\hat{y}_i$ given the input x_i. If the prediction is incorrect, the weight vector is updated in favor of the correct structure. Otherwise the weight vector is left untouched. In our setting inputs x_i correspond to documents and outputs y_i are trees over mentions in a document. The training data is, however, not annotated with trees, but only with clusters of mentions. That is, the y_i's are not defined a priori.

3.2 Latent antecedents

In order to have a tree structure to update against, we use the current weight vector and apply the decoder to a constrained antecedent set and obtain a **latent tree** over the mentions in a document, where each mention is assigned a single correct antecedent (Fernandes et al., 2012). We constrain the antecedent sets such that only trees that correspond to the correct clustering can be built. Specifically, let $\tilde{A}_j$ denote the set of correct antecedents for a mention m_j, or

$$\tilde{A}_j = \begin{cases} \{m_0\} \text{ if } m_j \text{ has no correct antecedent} \\ \{a_i \mid \textsc{Coref}(a_i, m_j), a_i \in A_j\} \text{ otherwise} \end{cases}$$

that is, if mention m_j is non-referential or the first mention of its cluster, $\tilde{A}_j$ contains only the document root. Otherwise it is the set of all mentions to the left that belong to the same cluster as m_j. Analogously to $\mathcal{A}$, let $\tilde{\mathcal{A}}$ denote the set of constrained antecedent sets. The latent tree $\tilde{y}$ needed for updates is then defined to be the optimal tree over $\mathcal{Y}(\tilde{\mathcal{A}})$, subject to the current weight vector:

$$\tilde{y} = \arg\max_{y \in \mathcal{Y}(\tilde{\mathcal{A}})} score(y)$$

The intuition behind the latent tree is that during online learning, the weight vector will start favoring latent trees that are easier to learn (such as the one in Figure 2).

Algorithm 1 PA algorithm with latent trees

Input: Training data D, number of iterations T
Output: Weight vector w

1: $w = \overrightarrow{0}$
2: **for** $t \in 1..T$ **do**
3: **for** $\langle M_i, \mathcal{A}_i, \tilde{\mathcal{A}}_i \rangle \in D$ **do**
4: $\hat{y}_i = \arg\max_{\mathcal{Y}(\mathcal{A})} score(y)$ ▷ Predict
5: **if** $\neg$ CORRECT$(\hat{y}_i)$ **then**
6: $\tilde{y}_i = \arg\max_{\mathcal{Y}(\tilde{\mathcal{A}})} score(y)$ ▷ Latent tree
7: $\Delta = \Phi(\hat{y}_i) - \Phi(\tilde{y}_i)$
8: $\tau = \frac{\Delta \cdot w + \text{Loss}(\hat{y}_i)}{\|\Delta\|^2}$ ▷ PA weight
9: $w = w + \tau\Delta$ ▷ PA update
10: **return** w

Algorithm 1 shows pseudocode for the learning algorithm, which we will refer to as the **baseline learning algorithm**. Instead of looping over pairs $\langle x, y \rangle$ of documents and trees, it loops over triples $\langle M, \mathcal{A}, \tilde{\mathcal{A}} \rangle$ that comprise the set of mentions M and the two sets of antecedent candidates (line 3). Moreover, rather than checking that the tree is identical to the latent tree, it only requires the tree to correctly encode the gold clustering (line 5). The update that occurs in lines 7-9 is the passive-aggressive update. A loss function LOSS that quantifies the error in the prediction is used to compute a scalar τ that controls how much the weights are moved in each update. If τ is set to 1, the update reduces to the standard structured perceptron update. The loss function can be an arbitrarily complex function that returns a numerical value of how bad the prediction is. In the simplest case, Hamming loss can be used, i.e., for each incorrect arc add 1. We follow Fernandes et al. (2012) and penalize erroneous root attachments, i.e., mentions that erroneously get the root node as their antecedent, with a loss of 1.5. For all other arcs we use Hamming loss.

4 Incremental Search

We now show that the search problem in (2) can equivalently be solved by the more intuitive **best-first decoder** (Ng and Cardie, 2002), rather than using the CLE decoder. The best-first decoder

[4] We also implement the feature mapping function Φ as a *hash kernel* (Bohnet, 2010) and apply *averaging* (Collins, 2002), though for brevity we omit this from the pseudocode.

works incrementally by making a left-to-right pass over the mentions, selecting for each mention the highest scoring antecedent.

The key aspect that makes the best-first decoder equivalent to the CLE decoder is that all arcs point from left to right, both in this paper and in the work of Fernandes et al. (2012). We sketch a proof that this decoder also returns the highest scoring tree.

First, note that this algorithm indeed returns a tree. This can be shown by assuming the opposite, in which case the tree has to have a cycle. Then there must be a mention that has its antecedent to the right. Though this is not possible since all arcs point from left to right.

Second, this tree is the highest scoring tree. Again, assume the contrary, i.e., that there is a higher scoring tree in $\mathcal{Y}(\mathcal{A})$. This implies that for some mention there is a higher scoring antecedent than the one selected by the decoder. This contradicts the fact that the best-first decoder selects the highest scoring antecedent for each mention.[5]

5 Introducing Non-local Features

Since the best-first decoder makes a left-to-right pass, it is possible to extract features on the partial structure on the left. Such **non-local features** are able to capture information beyond that of a mention and its potential antecedent, e.g., the size of a partially built cluster, or features extracted from the antecedent of the antecedent.

When only local features are used, greedy search (either with CLE or the best-first decoder) suffices to find the highest scoring tree. That is, greedy search provides an exact solution to equation 2. Non-local features, however, render the exact search problem intractable. This is because with non-local features, locally suboptimal (i.e., non-greedy) antecedents for some mentions may lead to a higher total score over a whole document.

In order to keep some options around during search, we extend the best-first decoder with **beam search**. Beam search works incrementally by keeping an *agenda* of *state items*. At each step, all items on the agenda are expanded. The subset of size k (the *beam size*) of the highest scoring expansions are retained and put back into the agenda for the next step. The feature extraction function Φ is also extended such that it also receives the current state s as an argument: $\Phi(\langle m_i, m_j \rangle, s)$. The state encodes the previous decisions and enables Φ to extract features from the partial tree on the left.

We now outline three different ways of learning the weight vector w with non-local features.

5.1 Early updates

The beam search decoder can be plugged into the training algorithm, replacing the calls to $\arg\max$. Since state items leading to the best tree may be pruned from the agenda before the decoder reaches the end of the document, the introduction of non-local features may cause the decoder to return a non-optimal tree. This is problematic as it might cause updates although the correct tree has a higher score than the predicted one. It has previously been observed (Huang et al., 2012) that substantial gains can be made by applying an **early update** strategy (Collins and Roark, 2004): if the correct item is pruned before reaching the end of the document, then stop and update.

While beam search and early updates have been successfully applied to other NLP applications, our task differs in two important aspects: First, coreference resolution is a much more difficult task, which relies on more (world) knowledge than what is available in the training data. In other words, it is unlikely that we can devise a feature set that is informative enough to allow the weight vector to converge towards a solution that lets the learning algorithm see the entire documents during training, at least in the situation when no external knowledge sources are used.

Second, our gold structure is not known but is induced latently, and may vary from iteration to iteration. With non-local features this is troublesome since the best latent tree of a complete document may not necessarily coincide with the best *partial* tree at some intermediate mention m_j, $j < n$, i.e., a mention before the last in a document. We therefore also apply beam search to find the latent tree to have a partial gold structure for every mention in a document.

Algorithm 2 shows pseudocode for the beam search and early update training procedure. The algorithm maintains two parallel agendas, one for gold items and one for predicted items. At every mention, both agendas are expanded and thus cover the same set of mentions. Then the predicted agenda is checked to see if it contains any correct

[5] In case there are multiple maximum spanning trees, the best-first decoder will return one of them. This also holds for the CLE algorithm. With proper definitions, the proof can be constructed to show that both search algorithms return trees belonging to the *set* of maximum spanning trees over a graph.

Algorithm 2 Beam search and early update

Input: Data set D, epochs T, beam size k
Output: weight vector w

```
 1: w = 0⃗
 2: for t ∈ 1..T do
 3:     for ⟨M_i, A_i, Ã_i⟩ ∈ D do
 4:         Agenda_G = {}
 5:         Agenda_P = {}
 6:         for j ∈ 1..n do
 7:             Agenda_G = EXPAND(Agenda_G, Ã_j, m_j, k)
 8:             Agenda_P = EXPAND(Agenda_P, A_j, m_j, k)
 9:             if ¬ CONTAINSCORRECT(Agenda_P) then
10:                 ỹ = EXTRACTBEST(Agenda_G)
11:                 ŷ = EXTRACTBEST(Agenda_P)
12:                 update                          ▷ PA update
13:                 GOTO 3   ▷ Skip and move to next instance
14:         ŷ = EXTRACTBEST(Agenda_P)
15:         if ¬ CORRECT(ŷ) then
16:             ỹ = EXTRACTBEST(Agenda_G)
17:             update                              ▷ PA update
```

item. If there is no correct item in the predicted agenda, search is halted and an update is made against the best item from the gold agenda. The algorithm then moves on to the next document. If the end of a document is reached, the top scoring predicted item is checked for correctness. If it is not, an update is made against the best gold item.

A drawback of early updates is that the remainder of the document is skipped when an early update is applied, effectively discarding some training data.[6] An alternative strategy that makes better use of the training data is to apply the max-violation procedure suggested by Huang et al. (2012). However, since our gold trees change from iteration to iteration, and even inside of a single document, it is not entirely clear with respect to what gold tree the maximum violation should be computed. Initial experiments with max-violation updates indicated that they did not improve much over early updates, and also had a tendency to only consider a smaller portion of the training data.

5.2 LaSO

To make full use of the training data we implemented Learning as Search Optimization (**LaSO**; Daumé III and Marcu, 2005b). It is very similar to early updates, but differs in one crucial respect: When an early update is made, search is continued rather than aborted. Thus the learning algorithm always reaches the end of a document, avoiding the problem that early updates discard parts of the training data.

[6] In fact, after 50 iterations about 70% of the mentions in the training data are still being ignored due to early updates.

Correct items are computed the same way as with early updates, where an agenda of gold items is maintained in parallel. When search is resumed after an intermediate LaSO update, the prediction agenda is re-seeded with gold items (i.e., items that are all correct). This is necessary since the update influences what the partial gold structure looks like, and the gold agenda therefore needs to be recreated from the beginning of the document. Specifically, after each intermediate LaSO update, the gold agenda is expanded repeatedly from the beginning of the document to the point where the update was made, and is then copied over to seed the prediction agenda. In terms of pseudocode, this is accomplished by replacing lines 12 and 13 in Algorithm 2 with the following:

```
12:                 update                          ▷ PA update
13:                 Agenda_G = {}
14:                 for m_i ∈ {m_1, ..., m_j}   ▷ Recreate gold agenda
15:                     Agenda_G = EXPAND(Agenda_G, Ã_i, m_i, k)
16:                 Agenda_P = COPY(Agenda_G)
17:                 GOTO 6                          ▷ Continue
```

5.3 Delayed LaSO updates

When we applied LaSO, we noticed that it performed worse than the baseline learning algorithm when only using local features. We believe that the reason is that updates are made in the middle of documents which means that lexical forms of antecedents are "fresh in memory" of the weight vector. This results in fewer mistakes during training and leads to fewer updates. While this feedback makes it easier during training, such feedback is not available during test time, and the LaSO learning setting therefore mimics the testing setting to a lesser extent.

We also found that LaSO updates change the shape of the latent tree and that the average distance between mentions connected by an arc increased. This problem can also be attributed to how lexical items are fresh in memory. Such trees tend to deviate from the intuition that the latent trees are easier to learn. They also render distance-based features (which are standard practice and generally rather useful) less powerful, as distance in sentences or mentions becomes less of a reliable indicator for coreference.

To cope with this problem, we devised the **delayed LaSO** update, which differs from LaSO only in the respect that it postpones the actual updates until the end of a document. This is accomplished by summing the distance vectors Δ at every point where LaSO would make an update. At

Algorithm 3 Delayed LaSO update

Input: Data set D, iterations T, beam size k
Output: weight vector w

$w = \overrightarrow{0}$
for $t \in 1..T$ **do**
  for $\langle M_i, \mathcal{A}_i, \tilde{\mathcal{A}}_i \rangle \in D$ **do**
    $Agenda_G = \{\}$
    $Agenda_P = \{\}$
    $\Delta_{acc} = \overrightarrow{0}$
    $loss_{acc} = 0$
    for $j \in 1..n$ **do**
      $Agenda_G = \text{EXPAND}(Agenda_G, \tilde{A}_j, m_j, k)$
      $Agenda_P = \text{EXPAND}(Agenda_P, A_j, m_j, k)$
      if $\neg\,\text{CONTAINSCORRECT}(Agenda_P)$ **then**
        $\tilde{y} = \text{EXTRACTBEST}(Agenda_G)$
        $\hat{y} = \text{EXTRACTBEST}(Agenda_P)$
        $\Delta_{acc} = \Delta_{acc} + \Phi(\hat{y}) - \Phi(\tilde{y})$
        $loss_{acc} = loss_{acc} + \text{LOSS}(\hat{y})$
        $Agenda_P = Agenda_G$
    $\hat{y} = \text{EXTRACTBEST}(Agenda_P)$
    if $\neg\,\text{CORRECT}(\hat{y})$ **then**
      $\tilde{y} = \text{EXTRACTBEST}(Agenda_G)$
      $\Delta_{acc} = \Delta_{acc} + \Phi(\hat{y}) - \Phi(\tilde{y})$
      $loss_{acc} = loss_{acc} + \text{LOSS}(\hat{y})$
    if $\Delta_{acc} \neq \overrightarrow{0}$ **then**
      update w.r.t. Δ_{acc} and $loss_{acc}$

the end of a document, an update is made with respect to the sum of all Δ's. Similarly, a running sum of the partial loss is maintained within a document. Since the PA update only depends on the distance vector Δ and the loss, it can be applied with respect to these sums at the end of the document. When only local features are used, this update is equivalent to the updates in the baseline learning algorithm. This follows because greedy search finds the optimal tree when only local features are used. Similarly, using only local features, the beam-based best-first decoder will also return the optimal tree. Algorithm 3 shows the pseudocode for the delayed LaSO learning algorithm.

6 Features

In this section we briefly outline the type of features we use. The feature sets are customized for each language. As a baseline we use the features from Björkelund and Farkas (2012), who ranked second in the 2012 CoNLL shared task and is publicly available. The exact definitions and feature sets that we use are available as part of the download package of our system.

6.1 Local features

Basic features that can be extracted on one or both mentions in a pair include (among others): **Mention type**, which is either root, pronoun, name, or common; **Distance** features, e.g., the distance in sentences or mentions; **Rule-based** features, e.g., StringMatch or SubStringMatch; **Syntax-based** features, e.g., category labels or paths in the syntax tree; **Lexical** features, e.g., the head word of a mention or the last word of a mention.

In order to have a strong local baseline, we applied greedy forward/backward feature selection on the training data using a large set of local feature templates. Specifically, the training set of each language was split into two parts where 75% was used for training, and 25% for testing. Feature templates were incrementally added or removed in order to optimize the mean of MUC, B^3, and $CEAF_e$ (i.e., the CoNLL average).

6.2 Non-local Features

We experimented with non-local features drawn from previous work on entity-mention models (Luo et al., 2004; Rahman and Ng, 2009), however they did not improve performance in preliminary experiments. The one exception is the size of a cluster (Culotta et al., 2007). Additional features we use are

Shape encodes the linear "shape" of a cluster in terms of mention type. For instance, the clusters representing *Gary Wilber* and *Drug Emporium Inc.* from the example in Figure 1, would be represented as RNPN and RNCCC, respectively. Where R, N, P, and C denote the root node, names, pronouns, and common noun phrases, respectively.

Local syntactic context is inspired by the Entity Grid (Barzilay and Lapata, 2008), where the basic assumption is that references to an entity follow particular syntactic patterns. For instance, an entity may be introduced as an object in one sentence, whereas in subsequent sentences it is referred to in subject position. Grammatical functions are approximated by the path in the syntax tree from a mention to its closest S node. The partial paths of a mention and its linear predecessor, given the cluster of the current antecedent, informs the model about the local syntactic context.

Cluster start distance denotes the distance in mentions from the beginning of the document where the cluster of the antecedent in consideration begins.

Additionally, the non-local model also has access to the basic properties of other mentions in the partial tree structure, such as head words. The

non-local features were selected with the same greedy forward strategy as the local features, starting from the optimized local feature sets.

7 Experimental Setup

We apply our model to the CoNLL 2012 Shared Task data, which includes a training, development, and test set split for three languages: Arabic, Chinese and English. We follow the *closed track* setting where systems may only be trained on the provided training data, with the exception of the English gender and number data compiled by Bergsma and Lin (2006). We use automatically extracted mentions using the same mention extraction procedure as Björkelund and Farkas (2012).

We evaluate our system using the CoNLL 2012 scorer, which computes several coreference metrics: MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and $CEAF_e$ and $CEAF_m$ (Luo, 2005). We also report the CoNLL average (also known as MELA; Denis and Baldridge (2009)), i.e., the arithmetic mean of MUC, B^3, and $CEAF_e$. It should be noted that for B^3 and the CEAF metrics, multiple ways of handling *twinless mentions*[7] have been proposed (Rahman and Ng, 2009; Stoyanov et al., 2009). We use the most recent version of the CoNLL scorer (version 7), which implements the original definitions of these metrics.[8]

Our system is evaluated on the version of the data with automatic preprocessing information (e.g., predicted parse trees). Unless otherwise stated we use 25 iterations of perceptron training and a beam size of 20. We did not attempt to tune either of these parameters. We experiment with two feature sets for each language: the optimized local feature sets (denoted *local*), and the optimized local feature sets extended with non-local features (denoted *non-local*).

8 Results

Learning strategies. We begin by looking at the different learning strategies. Since early updates do not always make use of the complete documents during training, it can be expected that it will require either a very wide beam or more iterations to get up to par with the baseline learning algorithm. Figure 3 shows the CoNLL average on the English development set as a function of number of training iterations with two different beam sizes, 20 and 100, over the local and non-local feature sets. The figure shows that even after 50 iterations, early update falls short of the baseline, even when the early update system has access to more informative non-local features.[9]

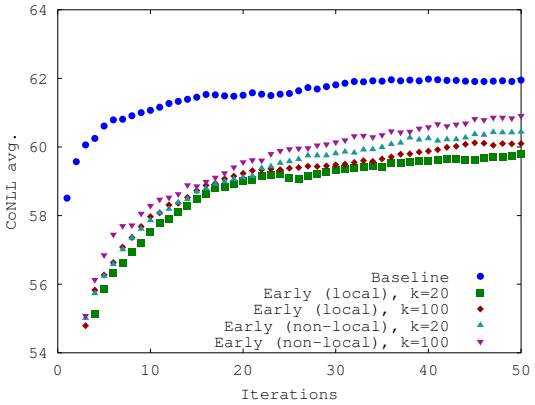

Figure 3: Comparing early update training with the baseline training algorithm.

In Figure 4 we compare early update with LaSO and delayed LaSO on the English development set. The left half uses the local feature set, and the right the extended non-local feature set. Recall that with only local features, delayed LaSO is equivalent to the baseline learning algorithm. As before, early update is considerably worse than other learning strategies. We also see that delayed LaSO outperforms LaSO, both with and without non-local features. Note that plain LaSO with non-local features only barely outperforms the delayed LaSO with only local features (i.e., the baseline), which indicates that only delayed LaSO is able to fully leverage non-local features. From these results we conclude that we are better off when the learning algorithm handles one document at a time, instead of getting feedback within documents.

Local vs. Non-local feature sets. Table 1 displays the differences in F-measures and CoNLL average between the local and non-local systems when applied to the development sets for each language. All metrics improve when more informative non-local features are added to the local feature set. Arabic and English show considerable improvements, and the CoNLL average increases

[7] i.e., mentions that appear in the prediction but not in gold, or the other way around

[8] Available at `http://conll.cemantix.org/2012/software.html`

[9] Although the Early systems still seem to show slight increases after 50 iterations, it needs a considerable number of iterations to catch up with the baseline – after 100 iterations the best early system is still more than half a point behind the baseline.

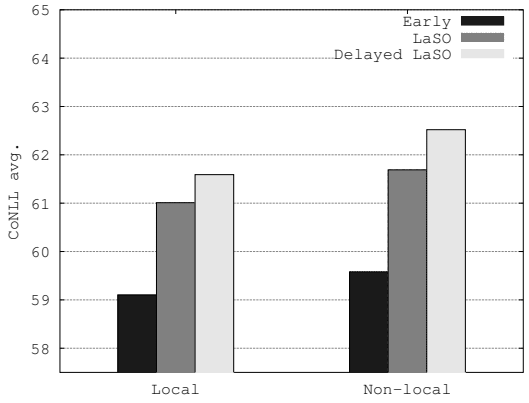

Figure 4: Comparison of learning algorithms evaluated on the English development set.

	MUC	B^3	$CEAF_m$	$CEAF_e$	CoNLL
Arabic					
local	47.33	42.51	49.71	46.49	45.44
non-local	49.31	43.52	50.96	47.18	46.67
Chinese					
local	65.84	57.94	62.23	57.05	60.27
non-local	66.4	57.99	62.37	57.12	60.5
English					
local	69.95	58.7	62.91	56.03	61.56
non-local	70.74	60.03	65.01	56.8	62.52

Table 1: Comparison of local and non-local feature sets on the development sets.

about one point. For Chinese the gains are generally not as pronounced, though the MUC metric goes up by more than half a point.

Final results. In Table 2 we compare the results of the non-local system (*This paper*) to the best results from the CoNLL 2012 Shared Task.[10] Specifically, this includes Fernandes et al.'s (2012) system for Arabic and English (denoted Fernandes), and Chen and Ng's (2012) system for Chinese (denoted C&N). For English we also compare it to the Berkeley system (Durrett and Klein, 2013), which, to our knowledge, is the best publicly available system for English coreference resolution (denoted D&K). As a general baseline, we also include Björkelund and Farkas' (2012) system (denoted B&F), which was the second best system in the shared task. For almost all metrics our system is significantly better than the best competitor. For a few metrics the best competitor outperforms our results for either precision or recall, but in terms of F-measures and the CoNLL average our system is the best for all languages.

[10]Thanks to Sameer Pradhan for providing us with the outputs of the other systems for significance testing.

9 Related Work

On the machine learning side Collins and Roark's (2004) work on the early update constitutes our starting point. The LaSO framework was introduced by Daumé III and Marcu (2005b), but has, to our knowledge, only been applied to the related task of entity detection and tracking (Daumé III and Marcu, 2005a). The theoretical motivation for early updates was only recently explained rigorously (Huang et al., 2012). The delayed LaSO update that we propose decomposes the prediction task of a complex structure into a number of subproblems, each of which guarantee *violation*, using Huang et al.'s (2012) terminology. We believe this is an interesting novelty, as it leverages the complete structures for every training instance during every iteration, and expect it to be applicable also to other structured prediction tasks.

Our approach also resembles imitation learning techniques such as SEARN (Daumé III et al., 2009) and DAGGER (Ross et al., 2011), where the search problem is reduced to a sequence of classification steps that guide the search algorithm through the search space. These frameworks, however, rely on the notion of an *expert policy* which provides an optimal decision at each point during search. In our context that would require antecedents for every mention to be given a priori, rather than using latent antecedents as we do.

Perceptrons for coreference. The perceptron has previously been used to train coreference resolvers either by casting the problem as a binary classification problem that considers pairs of mentions in isolation (Bengtson and Roth, 2008; Stoyanov et al., 2009; Chang et al., 2012, *inter alia*) or in the structured manner, where a clustering for an entire document is predicted in one go (Fernandes et al., 2012). However, none of these works use non-local features. Stoyanov and Eisner (2012) train an Easy-First coreference system with the perceptron to learn a sequence of join operations between arbitrary mentions in a document and accesses non-local features through previous merge operations in later stages. Culotta et al. (2007) also apply online learning in a first-order logic framework that enables non-local features, though using a greedy search algorithm.

Latent antecedents. The use of latent antecedents goes back to the work of Yu and Joachims (2009), although the idea of determining

	MUC			B^3			$CEAF_m$			$CEAF_e$			CoNLL
	Rec	Prec	F_1	Rec	Prec	F_1	Rec	Prec	F_1	Rec	Prec	F_1	avg.
						Arabic							
B&F	43.9	52.51	47.82	35.7	49.77	41.58	43.8	50.03	46.71	40.45	41.86	41.15	43.51
Fernandes	43.63	49.69	46.46	38.39	47.7	42.54	47.6	50.85	49.17	48.16	45.03	46.54	45.18
This paper	**47.53**	**53.3**	**50.25**	**44.14**	49.34	**46.6**	**50.94**	**55.19**	**52.98**	49.2	**49.45**	**49.33**	48.72
						Chinese							
B&F	58.72	58.49	58.61	49.17	53.2	51.11	56.68	51.86	54.14	55.36	41.8	47.63	52.45
C&N	59.92	64.69	62.21	51.76	60.26	55.69	59.58	60.45	60.02	**58.84**	51.61	54.99	57.63
This paper	**62.57**	**69.39**	**65.8**	**53.87**	**61.64**	**57.49**	58.75	**64.76**	**61.61**	54.65	**59.33**	**56.89**	60.06
						English							
B&F	65.23	70.1	67.58	49.51	60.69	54.47	56.93	59.51	58.19	51.34	49.14	59.21	57.42
Fernandes	65.83	75.91	70.51	51.55	65.19	57.58	57.48	65.93	61.42	50.82	57.28	53.86	60.65
D&K	66.58	74.94	70.51	53.2	**64.56**	58.33	59.19	66.23	62.51	52.9	58.06	55.36	61.4
This paper	**67.46**	74.3	**70.72**	**54.96**	62.71	**58.58**	**60.33**	**66.92**	**63.45**	52.27	**59.4**	55.61	61.63

Table 2: Comparison with other systems on the test sets. Bold numbers indicate significance at the $p < 0.05$ level between the best and the second best systems (according to the CoNLL average) using a Wilcoxon signed rank sum test. We refrain from significance tests on the CoNLL average, as it is an average over other F-measures.

meaningful antecedents for mentions can be traced back to Ng and Cardie (2002) who used a rule-based approach. Latent antecedents have recently gained popularity and were used by two systems in the CoNLL 2012 Shared Task, including the winning system (Fernandes et al., 2012; Chang et al., 2012). Durrett and Klein (2013) present a coreference resolver with latent antecedents that predicts clusterings over entire documents and fit a log-linear model with a custom task-specific loss function using AdaGrad (Duchi et al., 2011). Chang et al. (2013) use a max-margin approach to learn a pairwise model and rely on stochastic gradient descent to circumvent the costly operation of decoding the entire training set in order to compute the gradients and the latent antecedents. None of the aforementioned works use non-local features in their models, however.

Entity-mention models. Entity-mention models that compare a single mention to a (partial) cluster have been studied extensively and several works have evaluated non-local entity-level features (Luo et al., 2004; Yang et al., 2008; Rahman and Ng, 2009). Luo et al. (2004) also apply beam search at test time, but use a static assignment of antecedents and learns log-linear model using batch learning. Moreover, these works alter the basic feature definitions from their pairwise models when introducing entity-level features. This contrasts with our work, as our mention-pair model simply constitutes a special case of the non-local system.

10 Conclusion

We presented experiments with a coreference resolver that leverages non-local features to improve its performance. The application of non-local features requires the use of an approximate search algorithm to keep the problem tractable. We evaluated standard perceptron learning techniques for this setting both using early updates and LaSO. We found that the early update strategy is considerably worse than a local baseline, as it is unable to exploit all training data. LaSO resolves this issue by giving feedback within documents, but still underperforms compared to the baseline as it distorts the choice of latent antecedents.

We introduced a modification to LaSO, where updates are delayed until each document is processed. In the special case where only local features are used, this method coincides with standard structured perceptron learning that uses exact search. Moreover, it is also able to profit from non-local features resulting in improved performance. We evaluated our system on all three languages from the CoNLL 2012 Shared Task and present the best results to date on these data sets.

Acknowledgments

We are grateful to the anonymous reviewers as well as Christian Scheible and Wolfgang Seeker for comments on earlier versions of this paper. This research has been funded by the DFG via SFB 732, project D8.

References

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

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 294–303, Honolulu, Hawaii, October. Association for Computational Linguistics.

Shane Bergsma and Dekang Lin. 2006. Bootstrapping path-based pronoun resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 33–40, Sydney, Australia, July. Association for Computational Linguistics.

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

Bernd Bohnet, Simon Mille, Benoît Favre, and Leo Wanner. 2011. <stumaba >: From deep representation to surface. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 232–235, Nancy, France, September. Association for Computational Linguistics.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China, August.

Kai-Wei Chang, Rajhans Samdani, Alla Rozovskaya, Mark Sammons, and Dan Roth. 2012. Illinois-coref: The ui system in the conll-2012 shared task. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 113–117, Jeju Island, Korea, July. Association for Computational Linguistics.

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A constrained latent variable model for coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 601–612, Seattle, Washington, USA, October. Association for Computational Linguistics.

Chen Chen and Vincent Ng. 2012. Combining the best of two worlds: A hybrid approach to multilingual coreference resolution. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 56–63, Jeju Island, Korea, July. Association for Computational Linguistics.

Yoeng-jin Chu and Tseng-hong Liu. 1965. On the shortest aborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, July.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive–aggressive algorithms. *Journal of Machine Learning Reseach*, 7:551–585, March.

Aron Culotta, Michael Wick, and Andrew McCallum. 2007. First-order probabilistic models for coreference resolution. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 81–88, Rochester, New York, April. Association for Computational Linguistics.

Hal Daumé III and Daniel Marcu. 2005a. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 97–104, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Hal Daumé III and Daniel Marcu. 2005b. Learning as search optimization: approximate large margin methods for structured prediction. In *ICML*, pages 169–176.

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

Pascal Denis and Jason Baldridge. 2009. Global Joint Models for Coreference Resolution and Named Entity Classification. In *Procesamiento del Lenguaje Natural 42*, pages 87–96, Barcelona: SEPLN.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July.

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

Seattle, Washington, USA, October. Association for Computational Linguistics.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71(B):233–240.

Eraldo Fernandes, Cícero dos Santos, and Ruy Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 41–48, Jeju Island, Korea, July. Association for Computational Linguistics.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151, Montréal, Canada, June. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio, June. Association for Computational Linguistics.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 135–142, Barcelona, Spain, July.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

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

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1396–1411, Uppsala, Sweden, July. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea, July. Association for Computational Linguistics.

Altaf Rahman and Vincent Ng. 2009. Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 968–977, Singapore, August. Association for Computational Linguistics.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June. Association for Computational Linguistics.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, pages 627–635.

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 *Proceedings of COLING 2012*, pages 2519–2534, Mumbai, India, December.

Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. 2009. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 656–664, Suntec, Singapore, August. Association for Computational Linguistics.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model theoretic coreference scoring scheme. In *Proceedings MUC-6*, pages 45–52, Columbia, Maryland.

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 *Proceedings of ACL-08: HLT*, pages 843–851, Columbus, Ohio, June. Association for Computational Linguistics.

Chun-Nam Yu and T. Joachims. 2009. Learning structural svms with latent variables. In *International Conference on Machine Learning (ICML)*.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571, Honolulu, Hawaii, October. Association for Computational Linguistics.