Latent Trees for Coreference Resolution

Eraldo Rezende Fernandes*
Instituto Federal de Goiás

Cícero Nogueira dos Santos**
Brazilian Research Lab
IBM Research

Ruy Luiz Milidiú†
PUC-Rio

We describe a structure learning system for unrestricted coreference resolution that explores two key modeling techniques: latent coreference trees and automatic entropy-guided feature induction. The latent tree modeling makes the learning problem computationally feasible because it incorporates a meaningful hidden structure. Additionally, using an automatic feature induction method, we can efficiently build enhanced nonlinear models using linear model learning algorithms. We present empirical results that highlight the contribution of each modeling technique used in the proposed system. Empirical evaluation is performed on the multilingual unrestricted coreference CoNLL-2012 Shared Task data sets, which comprise three languages: Arabic, Chinese, and English. We apply the same system to all languages, except for minor adaptations to some language-dependent features such as nested mentions and specific static pronoun lists. A previous version of this system was submitted to the CoNLL-2012 Shared Task closed track, achieving an official score of 58.69*, the best among the competitors. The unique enhancement added to the current system version is the inclusion of candidate arcs linking nested mentions for the Chinese language. By including such arcs, the score increases by almost 4.5 points for that language. The current system shows a score of* 60.15*, which corresponds to a* 3.5% *error reduction, and is the best performing system for each of the three languages.*

1. Introduction

Mentions are textual references to real-world entities or events. In a given document, mentions that refer to the same entity are called **coreferring mentions** and form a **mention cluster**. Coreference resolution is the task of identifying the mention clusters in a document and has been a core research topic in natural language processing. It has wide applications in question answering, machine translation, automatic summarization, and information extraction. Coreference resolution systems have been evaluated for

* E-mail: eraldo.fernandes@ifg.edu.br. This work was developed when the first author was at PUC-Rio.
** E-mail: cicerons@br.ibm.com.
† E-mail: milidiu@inf.puc-rio.br.

Submission received: 5 February 2013; revised submission received: 22 December 2013; accepted for publication: 5 January 2014.

doi:10.1162/COLI_a_00200

several decades, beginning with MUC-6 (Sundheim and Grishman 1995). Following those evaluation efforts, the CoNLL-2011 Shared Task (Pradhan et al. 2011) has been dedicated to the modeling of unrestricted coreference resolution for English text. The CoNLL-2012 Shared Task (Pradhan et al. 2012) extends the task to a multilingual scope, considering three languages: Arabic, Chinese, and English.

A singleton is a mention cluster containing exactly one mention. The unrestricted coreference resolution task consists of identifying the non-singleton mention clusters in a document. This task is usually split into three subtasks: mention detection, mention clustering, and singleton elimination. In Figure 1, we present an illustrative example. First, ten mentions are detected and shown in bold. They are sequentially tagged with the numbers $1, 2, \ldots, 10$. Next, four mention clusters are identified by tagging each mention with one of the tags a, b, c, d to indicate its cluster, where $a = \{1, 2, 8, 9\}$, $b = \{3, 7\}$, $c = \{4, 5, 6\}$, $d = \{10\}$ are the four mention clusters. Finally, clusters that contain only one mention are ignored, such as the one with *Iran* as its unique mention. In this example, we ignore some noun phrases in the mention detection subtask to simplify the illustration.

The final subtask is trivial, given the solution of the previous one. For the first subtask, several specific heuristics have been proposed, enabling the construction of high-recall mention detectors. The second subtask is harder, as it requires a complex output. Most of the current effort toward solving the coreference resolution task is focused on the mention clustering subtask.

Here, we propose an approach to unrestricted coreference resolution that is based on two key modeling techniques: latent coreference trees and entropy-guided feature induction. Our approach is based on a graph whose nodes are the mentions in the given document. The arcs of this graph link mention pairs that are coreferent candidates. The resulting structure predictor has similar steps for training and testing.

Predictor training can be summarized as follows:

1. *Mention Detection* – where we build a graph node for each mention by adapting a predictor proposed by dos Santos and Carvalho (2011);
2. *Candidate Pair Generation* – where we add a directed arc for each candidate coreferent mention pair by adapting the sieves proposed by Lee et al. (2013);

1. *Mention Detection*

North Korea$_1$ opened **its**$_2$ doors to the **U.S.**$_3$ today, welcoming **Secretary of State Madeleine Albright**$_4$. **She**$_5$ says **her**$_6$ visit is a good start. The **U.S.**$_7$ remains concerned about **North Korea's**$_8$ missile development program and **its**$_9$ exports of missiles to **Iran**$_{10}$.

2. *Mention Clustering*

North Korea$_a$ opened **its**$_a$ doors to the **U.S.**$_b$ today, welcoming **Secretary of State Madeleine Albright**$_c$. **She**$_c$ says **her**$_c$ visit is a good start. The **U.S.**$_b$ remains concerned about **North Korea's**$_a$ missile development program and **its**$_a$ exports of missiles to **Iran**$_d$.

3. *Singleton Elimination*

North Korea$_a$ opened **its**$_a$ doors to the **U.S.**$_b$ today, welcoming **Secretary of State Madeleine Albright**$_c$. **She**$_c$ says **her**$_c$ visit is a good start. The **U.S.**$_b$ remains concerned about **North Korea's**$_a$ missile development program and **its**$_a$ exports of missiles to Iran.

Figure 1
Unrestricted coreference resolution subtasks: mention detection, clustering, and singleton elimination.

3. *Basic Feature Setting* – where we set basic features that indicate whether an arc is likely to be connecting a coreferent pair by adapting the features used by dos Santos and Carvalho (2011);
4. *Context Feature Induction* – where we conjoin basic features to generate complex features with high discriminating power by means of the entropy-guided feature induction method proposed by Fernandes and Milidiú (2012) and Milidiú, dos Santos, and Duarte (2008); and
5. *Coreference Tree Learning* – where we learn how to extract the trees that connect coreferent mentions in the graph of mentions by applying a large margin latent perceptron structure learning algorithm.

To provide features with high predicting power to our model, we use *entropy-guided feature induction*. Using this mechanism, we automatically generate several feature templates that capture coreference-specific local context knowledge. Furthermore, this feature induction mechanism extends the structured perceptron framework by providing an efficient general method to build strong nonlinear classifiers.

Predictor testing uses the same first three steps as in predictor training, followed by three further steps:

4. *Context Feature Setting* – where we set the values of the additional induced features selected at training;
5. *Coreference Tree Prediction* – where we apply the Chu-Liu-Edmonds algorithm to solve an optimal branching problem to find the maximum score coreference trees; and
6. *Coreference Cluster Extraction* – where we extract the clusters of coreferring mentions from the coreference trees.

For the Coreference Tree Prediction step, we solve an optimal branching problem to find the maximum score coreference trees. This process is efficiently performed by the Chu-Liu-Edmonds algorithm (Chu and Liu 1965; Edmonds 1967). A tree score is simply the sum of its arc scores, which are given by a weighted sum of the arc features. The feature weights are learned during training in the Coreference Tree Learning step, which is based on the structured perceptron algorithm. Because coreference trees are not given in the training data, we assume that these structures are *latent* and use the latent structured perceptron (Sun et al. 2009; Yu and Joachims 2009) as the learning algorithm. In fact, we use a large margin extension of this algorithm (Fernandes and Brefeld 2011).

The Coreference Cluster Extraction step is trivial by construction because its input is a set of coreference trees. Each coreference tree corresponds to a cluster of coreferring mentions.

Due to the different application set-ups for coreference resolution as a subtask, multiple metrics have been proposed for evaluating coreference performance. No single metric is clearly superior to the others. We follow the CoNLL-2012 Shared Task evaluation scheme, adopting the unweighted average of the MUC, B^3, and $CEAF_e$ metrics. We also use the multilingual data sets provided in the CoNLL-2012 Shared Task to assess our system. The official ranking for this task is given by the mean of the system scores on three languages: Arabic, Chinese, and English.

We apply the same system to all three languages with only some minor adaptations, such as language-dependent static pronoun lists. Our system does not consider verbs

when creating candidate mentions. Therefore, it does not resolve coreferences involving events. We participated in the CoNLL-2012 Shared Task with a previous version of our system (Fernandes, dos Santos, and Milidiú 2012). The system submitted to this task achieved scores of 54.22, 58.49, and 63.37 on the Arabic, Chinese, and English test sets, respectively. Its official score is thus 58.69, which is the best among the competitors. Later, we extended this system by including candidate arcs linking *nested* mentions for the Chinese language. This version shows an official score of 60.15, corresponding to a 1.46 point absolute improvement or a 3.5% error reduction. As far as we know, this is currently the best performing system on the CoNLL-2012 Shared Task Arabic, Chinese, and English test sets.

The remainder of this article is organized as follows. In Section 2, we review related work. In Section 3, we detail our approach to the mention detection subtask. In Section 4, we describe coreference trees, a key element in solving the mention clustering subtask in our system. We also examine the coreference tree prediction problem. In Section 5, we detail the entropy-guided feature induction method and its application to coreference resolution. In Section 6, we describe the large margin latent structured perceptron that we use to learn the coreference trees. In Section 7, we present our experimental setting. The experimental findings are provided in Section 8. Finally, in Section 9, we present our concluding remarks.

2. Related Work

Over the last two decades, many different machine learning–based approaches to coreference resolution have been proposed. Most of them use supervised learning and divide the task into two phases: the detection of potential mentions and the linking of mentions to form coreference chains, that is, mention clustering. In Ng (2010), the author presents a detailed review of supervised approaches to coreference resolution.

Many works that report results for the MUC-6 corpus (Sundheim and Grishman 1995) and the Automated Content Extraction (ACE) corpus (Doddington et al. 2004) use gold mention boundaries and, hence, do not address the mention detection task (Culotta, Wick, and Mccallum 2007; Finkel and Manning 2008; Poon and Domingos 2008; Haghighi and Klein 2009). Most of the works that perform mention detection use a set of heuristics. The common approach consists of extracting all noun phrases from the parse tree and considering them to be candidate mentions (Soon, Ng, and Lim 2001; dos Santos and Carvalho 2011; Haghighi and Klein 2010; Stoyanov et al. 2010; Chang et al. 2011; Bansal and Klein 2012; Lee et al. 2013; Sapena, Padró, and Turmo 2013). A few works approach the mention detection task by training classifiers (Bengtson and Roth 2008; Yuan et al. 2012). For the CoNLL data sets, training mention detectors is not a suitable approach because singleton mentions are not annotated in this corpus. Systems trained and tested with these data sets frequently privilege mention recall over precision (Chang et al. 2011; Lee et al. 2011; Sapena, Padró, and Turmo 2011). Therefore, these systems usually extract noun phrases that are not identified in the parse tree. This strategy increases consistency with the corpus annotation (Chang et al. 2011; Sapena, Padró, and Turmo 2011; Björkelund and Farkas 2012; Lee et al. 2013). The system proposed in this work uses a rule-based approach to mention detection, which is similar to previous work with the CoNLL data sets (dos Santos and Carvalho 2011).

The usual strategy for mention clustering consists of recasting the problem as a pairwise classification task (McCarthy and Lehnert 1995; Soon, Ng, and Lim 2001; Ng and Cardie 2002; Ponzetto and Strube 2006; Bengtson and Roth 2008; Ng 2009; Stoyanov et al. 2009; Björkelund and Nugues 2011; Uryupina et al. 2011; Uryupina, Moschitti, and

Poesio 2012). In this strategy, a preprocessing step is used to generate pairs of candidate coreferring mentions. In the training phase, it is necessary to generate positive and negative examples of coreferring pairs. For this purpose, the approach proposed by Soon, Ng, and Lim (2001) is normally used. For instance, in the CoNLL 2011 Shared Task (Pradhan et al. 2011), 11 of the 18 machine learning–based participant systems used Soon, Ng, and Lim's approach or a variation of it. In the classification phase, each candidate pair of mentions is classified as coreferring or not, using the classifier learned from the annotated corpus. When using the mention-pair approach, a linking step is necessary to remove inconsistencies that would result from the pairwise classifications and to construct a partition on the set of mentions. An aggressive strategy for this last step consists of merging each mention with all preceding mentions that are classified as coreferent with it (McCarthy and Lehnert 1995; dos Santos and Carvalho 2011). More sophisticated strategies, such as inference methods, have also been proposed (Bengtson and Roth 2008; Chang et al. 2011). One of the main disadvantages of the mention-pair classification approach is the lack of global information. Moreover, the lack of information on the clusters already formed can lead to contradictory links.

Methods have been proposed to overcome the limitations of mention-pair classification. Entity-mention models address the lack of information about the clusters already formed by seeking to classify whether a mention is coreferent with a preceding cluster (Luo et al. 2004; McCallum and Wellner 2005; Culotta, Wick, and Mccallum 2007; Yang et al. 2008). While entity-mention models have the advantage of allowing the creation of cluster-level features, they fail to identify the most probable candidate antecedent, just as the mention-pair classification approach does. Ranking models address this issue by directly comparing different candidate antecedents for the mention part of the training criterion (Yang et al. 2003; Yang, Su, and Tan 2008; Denis and Baldridge 2008). However, ranking models are unable to exploit cluster-level features. An approach that combines the advantages of entity-mention and ranking models was proposed by Rahman and Ng (2009). Their method, the cluster-ranking model, ranks preceding clusters rather than candidate antecedents, thereby enabling the use of cluster-level features.

Global inference methods that combine classification and clustering in one step have also been proposed. Cai and Strube (2010), Cai, Mujdricza-Maydt, and Strube (2011) and Sapena, Padró, and Turmo (2013) present systems that implement global decision via hypergraph partitioning. Whereas Cai and Strube (2010) and Cai, Mujdricza-Maydt, and Strube (2011) use a spectral clustering algorithm to perform hypergraph partitioning, Sapena, Padró, and Turmo (2013) use relaxation labeling for the resolution process. There are also approaches that perform global inference using integer linear programming to enforce consistency on the extracted coreference chains (Denis and Baldridge 2007; Klenner 2007; Finkel and Manning 2008).

Finley and Joachims (2005) and McCallum and Wellner (2005) formulate coreference resolution as a correlation clustering problem. The former use structural SVMs as the learning algorithm, and the latter use Collins' structured perceptron. Our system is also based on the structured perceptron; however, we use a large margin extension of this algorithm and use latent trees to represent each coreferring cluster. Yu and Joachims (2009) propose structural SVMs with latent variables and apply this approach to coreference resolution. They compare their results with those of Finley and Joachims (2005) and show that the latent trees significantly improve performance. The main issue with correlation clustering for coreference resolution is that all pairwise links between coreferring mentions are considered to compute the solution score. However, many pairs of coreferring mentions do not have a direct relation. For example, in Figure 1, mentions 2 and 9 are coreferent pronouns, but they are in neither anaphoric

nor cataphoric relation. Mention 1 is an antecedent of mention 2; mention 8 is an antecedent of mention 9; and mentions 1 and 8 are obviously coreferent. Hence, these two pronouns can be predicted as coreferent only by *transition* between other direct references. By using trees, the natural hierarchy of coreference dependencies is better modeled.

The model proposed by Yu and Joachims (2009) is highly related to ours because it is also based on latent trees. However, they use *undirected* trees to represent clusters, whereas we use directed trees. This difference is most likely not very significant, but we think that most coreference dependencies are directed relations. Using the example from Figure 1 again, we can see that mention 2 is a reference to mention 1 and not the other way around. By using directed trees, we model coreference dependencies, such as the dependencies between nouns and pronouns referring to the same entity. Other differences between Yu and Joachims (2009) and our work are that they do not use an artificial root node and, moreover, that their empirical evaluation is based on the MUC-6 task (Sundheim and Grishman 1995), whereas ours is based on the CoNLL-2012 Shared Task. The CoNLL-2012 Shared Task considers multilingual unrestricted coreference resolution. It defines a more general task than the one proposed by the MUC-6, as the annotated mentions in OntoNotes cover entities and events not limited to noun phrases or a limited set of entity types (Pradhan et al. 2011).

One key contribution of our work is the application of entropy-guided feature induction to coreference resolution, which we experimentally demonstrate as highly effective. There are some *on-line* methods (della Pietra, della Pietra, and Lafferty 1997; McCallum 2003) that perform feature induction within the learning algorithm. These methods choose among candidate features by evaluating their impact on performance along with the learning process. Our method is *off-line*. In this case, feature induction is performed by an efficient procedure before the training step. Off-line methods have two main advantages over on-line ones: (i) they are usually faster, as inducing features during training requires substantially more learning iterations; and (ii) there is not a need to modify the learning algorithm, so we can use it as is. The proposed method extends the automatic template generation strategy proposed in Entropy-Guided Transformation Learning (dos Santos and Milidiú 2009) to structure learning. This strategy has been previously applied to the English unrestricted coreference resolution task (dos Santos and Carvalho 2011). However, this previous work is based on a completely different task modeling and a different learning algorithm.

3. Mention Detection

The first subtask in coreference resolution, mention detection, is formally solved by an auxiliary predictor A given by

$$\boldsymbol{m} = A(\boldsymbol{d})$$

where $\boldsymbol{m}$ is the list of detected mentions in the given document $\boldsymbol{d}$. Singleton mentions are not annotated in the CoNLL-2012 data sets. Hence, a specific noun phrase may be annotated as a mention in one document but not in another document due to the absence of coreferring mentions in the latter. Therefore, machine learning–based mention detectors are difficult to build in this case.

In our system, the auxiliary predictor A is based on a specific set of rules that implement heuristics that have been proven effective in previous works (dos Santos and

Carvalho 2011). For a given document $\boldsymbol{d}$, this predictor generates a list $\boldsymbol{m}$ of candidate mentions by including the following items:

1. noun phrases from the provided parse tree;
2. pronouns, even when they appear inside larger noun phrases;
3. named entities in the categories Person (PERSON), Organization (ORG), and Geo-Political Entity (GPE), even when they appear inside larger noun phrases;
4. and possessive marks, to better align with the CoNLL-2012 annotated mentions.

If a mention is not detected in this step, our system is not able to resolve references to that mention. Toward this goal, we define a set of rules that rely mainly on the use of parse tree information and that privilege recall over precision. Note that rules 2, 3, and 4 have the effect of expanding the set of extracted noun phrases, thereby aligning the set of detected mentions more fully with the CoNLL-2012 annotated mentions. Note also that all but the fourth rule are language independent. The fourth rule is used for the English language only. This set of rules can be further improved by including rules that contain language-specific knowledge. For instance, Lee et al. (2013) use rules to remove occurrences of pleonastic it, as in *It is possible that* and *It seems that*.

The CoNLL-2012 Shared Task data sets also include coreferring mentions of events. However, the current version of our system does not consider verbs when creating candidate mentions and therefore does not resolve coreferences involving events.

4. Mention Clustering

In the mention clustering subtask, we must find a structure predictor F that assigns to its input $\boldsymbol{x}$ a high-quality mention clustering $\hat{\boldsymbol{y}}$ given by

$$\hat{\boldsymbol{y}} = F(\boldsymbol{x})$$

where $\boldsymbol{x} = (\boldsymbol{d}, \boldsymbol{m})$, with $\boldsymbol{d}$ a new unlabeled document and $\boldsymbol{m}$ its corresponding mention set.

In machine learning–based predictors, we learn F from a training set $\mathcal{D} = \{(\boldsymbol{x}, \boldsymbol{y})\}$ of correct input–output pairs. More specifically, the structured perceptron algorithm learns the weight vector $\boldsymbol{w}$ of a parameterized predictor given by

$$\hat{\boldsymbol{y}} = F(\boldsymbol{x}; \boldsymbol{w}) = \arg\max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} s(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}),$$

where $\mathcal{Y}(\boldsymbol{x})$ is the set of feasible outputs for $\boldsymbol{x}$, that is, all clusterings over the detected mentions $\boldsymbol{m}$ in $\boldsymbol{x}$, and s is a $\boldsymbol{w}$-parameterized scoring function over the clusterings. The prediction $\hat{\boldsymbol{y}}$ is the solution of an optimization problem and is called the *prediction problem*. The objective function of this problem is given by s, and it scores the candidate clusterings for the given document.

We introduce **coreference trees** to represent mention clusters. A coreference tree is a directed tree whose nodes are the coreferring mentions in a cluster and whose arcs

represent strong coreference relations between mentions. To illustrate this concept, we use the labeled document in Figure 1. In Figure 2, we present a plausible coreference tree for the $\{1, 2, 8, 9\}$ mention cluster.

We are not concerned about the semantics underlying coreference trees because they are simply auxiliary structures for the clustering task. Nevertheless, we have two arguments to justify the application of these structures. First, the concept is linguistically plausible because there is a dependency relation between coreferring mentions. Based on the aforementioned example, it may be observed that mention 8 (*North Korea's*) is indeed more likely to be associated with mention 1 (*North Korea*) than with mention 2 (*its*), even considering that, in the text, mention 8 is closer to mention 2 than to mention 1. Hence, the coreference trees should capture this dependency structure. Second, the concept is algorithmically plausible because most hierarchical clustering methods incrementally link an unclustered mention to a previous cluster. This operation can be viewed as adding a directed arc from the mention to its closest cluster member. We use a data set to learn a scoring function that captures the strength of the coreference relation between two mentions. This scoring function on the arcs guides the construction of the coreference trees with the largest aggregate scores.

This modeling approach is closely related to the one proposed by Yu and Joachims (2009). However, Yu and Joachims use undirected spanning trees instead of directed ones. We think that directed trees are more appropriate for use in coreference resolution to represent dependencies between mentions. Recalling the example from Figure 2, it is clear that mention 2 (*its*) is a reference to mention 1 (*North Korea*), not the other way around. Hence, there is indeed a direction related to the dependency between a pair of coreferring mentions.

For a whole document comprising certain coreferring clusters, we have a forest of coreference trees, one tree for each cluster. Hence, for the sake of simplicity, we link the root node of every coreference tree to an *artificial* root node, obtaining the *document tree*. In Figure 3, we depict a document tree for the current example.

Finally, it is important to remark that from a document tree, one can easily generate a mention clustering as follows:

1. remove the artificial root node and all its outgoing arcs, splitting the document tree into a forest;
2. from each tree in the resulting forest, output its node set as a mention cluster.

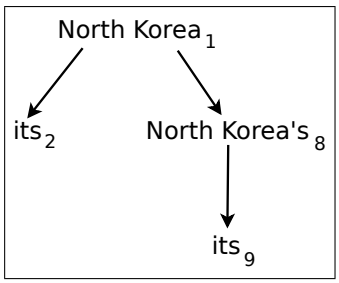

Figure 2
A possible coreference tree for the mention cluster $\{1, 2, 8, 9\}$ from the document in Figure 1.

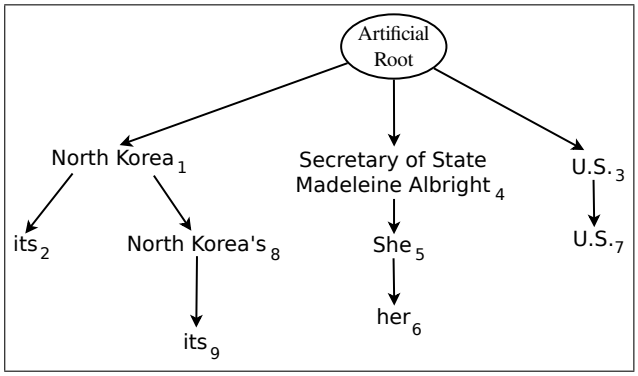

Figure 3
Document tree with artificial root node and its three coreference subtrees.

Hence, coreference trees provide a powerful auxiliary structure for solving the mention clustering subtask. Formally, we decompose the original predictor into two predictors, as follows:

$$F(\boldsymbol{x}) \equiv F_y(F_h(\boldsymbol{x}))$$

where $F_y(\boldsymbol{h})$ is the straightforward two-step procedure described here to generate a mention clustering from a document tree $\boldsymbol{h}$, and $F_h(\boldsymbol{x})$ is the *document tree predictor*, which is defined as

$$\hat{\boldsymbol{h}} = F_h(\boldsymbol{x}) = \arg\max_{h \in \mathcal{H}(\boldsymbol{x})} \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{h}) \rangle$$

with $\mathcal{H}(\boldsymbol{x})$ defined as the set of feasible document trees for the mentions in $\boldsymbol{x}$ and $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{h})$ as the joint feature vector representation of $\boldsymbol{x}$ and document tree $\boldsymbol{h}$.

The document tree predictor $F_h(\boldsymbol{x})$ finds a maximum scoring rooted tree over the feasible document trees for mentions in $\boldsymbol{x}$. A tree score is given by a linear function over its features. To understand how this predictor works, we must define the set $\mathcal{H}(\boldsymbol{x})$ of feasible document trees, the feature vector $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{h})$, and the algorithm that solves the arg max combinatorial problem.

The trees in $\mathcal{H}(\boldsymbol{x})$ are subgraphs of the directed graph $G(\boldsymbol{x})$, which is called the *candidate pairs graph*. Therefore, we must describe how to build the nodes and the directed arcs of this graph. The input $\boldsymbol{x} = (\boldsymbol{d}, \boldsymbol{m})$ provides $\boldsymbol{m}$, the list of mentions in document $\boldsymbol{d}$. For each mention in $\boldsymbol{m}$, a node is created in $G(\boldsymbol{x})$. Thus, an arc in $G(\boldsymbol{x})$ connects a pair of mentions in $\boldsymbol{x}$. Ideally, we would consider the complete graph for each document. However, because the number of mentions may be large and because most mention pairs are not coreferent, we filter the arcs simply by using sieves from the method proposed by Lee et al. (2013) for English coreference resolution. To further reduce the total number of arcs, we only consider forward arcs, which means that a directed arc (i, j) from mention i to mention j is only included in $G(\boldsymbol{x})$ if i appears before j in the document text. In this way, we avoid including many unnecessary arcs in $G(\boldsymbol{x})$, which speeds up the training. Each forward arc that passes through any of the used sieves is included in $G(\boldsymbol{x})$ and is called a **candidate arc**. We also refer to the two mentions of a candidate arc as a **candidate pair**. Because we heuristically filter arcs out of $G(\boldsymbol{x})$,

the correct cluster $\boldsymbol{y}$ can be disconnected in $G(\boldsymbol{x})$. In such a case, the correct prediction of $\boldsymbol{y}$ is impossible. However, such cases are rare, given that the recall of the used sieves is approximately 90%. To complete $G(\boldsymbol{x})$, we add an **artificial root node** and connect it to each mention in $\boldsymbol{x}$. These extra arcs are the **artificial arcs**, and under the current setting, they also point to the first mention in a cluster. Hence, we can learn how to use them to control the number of mention clusters.

The next predictor ingredient to be defined is the feature vector $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{h})$. This vector is linearly decomposed as

$$\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{h}) = \sum_{e \in \boldsymbol{h}} \boldsymbol{\Phi}(\boldsymbol{x}, e)$$

where $e = (i, j)$ is a directed arc in $\boldsymbol{h}$ linking mention i to mention j, and $\boldsymbol{\Phi}(\boldsymbol{x}, e) = (\phi_1(\boldsymbol{x}, e), \ldots, \phi_M(\boldsymbol{x}, e))$ is a feature vector that describes e. Each of these features helps to identify whether i and j are coreferent or not. To define these features, we utilize the entropy-guided feature induction method described in the next section.

Our coreference resolution approach is similar to previous structure learning approaches for dependency parsing (McDonald, Crammer, and Pereira 2005; Fernandes and Milidiú 2012). Thus, the arg max combinatorial problem reduces to a maximum branching problem, which can be efficiently solved using the Chu-Liu-Edmonds algorithm (Chu and Liu 1965; Edmonds 1967).

5. Entropy-Guided Feature Induction

The CoNLL-2012 Shared Task data sets include features that are either naturally present in documents, such as words, or automatically generated by external systems, such as POS tags and named entities information. We call this information provided by the data sets **basic features**. At the same time, most structure learning algorithms are based on linear models, as such algorithms have strong theoretical guarantees regarding their prediction performance and, moreover, are computationally efficient. However, linear models using basic features alone do not capture enough information to effectively represent coreference dependencies. Conjoining basic features to derive new features is a common way to introduce nonlinear contextual patterns into linear models.

The entropy-guided feature induction (EFI) method automatically derives a set of basic feature conjunctions, which we call **feature templates**. These templates are later used to generate the **derived features**, which comprise the input feature vectors $\boldsymbol{\Phi}(\boldsymbol{x}, e)$ used in the structured modeling described in the previous sections.

EFI conjoins the basic features that are useful to predict whether the mentions linked by an arc are coreferring. To solve this auxiliary binary classification problem, we derive a new set of data sets from the original one. Using all arcs in each of the candidate pair graphs, we obtain the *arc data set* $D = \{(\boldsymbol{\Psi}(e), c(e))\}$, comprising the arc basic feature vectors $\boldsymbol{\Psi}(e)$ along with their binary classification $c(e)$. In Table 1, we depict an example of such a data set for the document in Figure 1. This example includes the following features: *i-head* is the head word of mention i; *i-pos* is the head POS tag of mention i; *j-head* is the head word of mention j; *j-pos* is the head POS tag of mention j; *sameNE* indicates whether both mentions have the same named entity type; and *dist* is the number of mentions in the document text that occur between mentions i and j.

Table 1
Arc data set with some arcs from the document in Figure 1.

e		$\Psi(e)$						$c(e)$
i	j	**i-head**	**i-pos**	**j-head**	**j-pos**	**sameNE**	**dist**	
1	2	Korea	Noun	its	Pronoun	N	0	1
1	3	Korea	Noun	U.S.	Noun	Y	1	0
2	4	its	Pronoun	Secretary	Noun	N	1	0
1	9	Korea	Noun	its	Pronoun	N	7	1

The EFI method automatically generates feature templates for a structure learning problem by conjoining basic features that are highly discriminative. This method is based on the conditional entropy of arc label $c(e)$ given the basic features $\Psi(e)$.

The first step of the proposed method is to train a decision tree on the arc data set. The decision variable indicates whether the arc links two coreferring mentions. We use the arcs of all training examples; that is, for each training document, we generate an example for each candidate arc. Thus, the learned decision tree (DT) predicts whether the mentions linked by an arc are coreferring. In Figure 4, we present a decision tree learned from an arc data set. Each *internal* node in the DT corresponds to a feature; each *leaf* node has a label value (0 or 1, in the binary case); and each arc is labeled with a value of the source node feature. Note that in this sample DT, we suppress several possible feature values to simplify the presentation.

The most popular decision tree learning algorithms use the Gain Ratio to select the most informative feature (Quinlan 1992; Su and Zhang 2006). The Gain Ratio is simply a normalized version of the information gain measure. Hence, these algorithms provide a quick way to obtain entropy-guided feature selection. We propose a new automatic feature generation method for structure learning algorithms. The key idea is to use decision tree induction to conjoin the basic features. One of the most frequently used algorithms for DT induction is C4.5 (Quinlan 1992). We use Quinlan's C4.5 system to obtain the required entropy-guided selected features.

Our method uses a very simple scheme to extract feature templates. We consider all partial paths starting at the root node. For each path, a template is created by conjoining all of its node features. Because we aim to generate feature *templates*—conjunctions of basic features that do not include feature values—we ignore the feature values and the decision variable values in the DT. Thus, we do not use arc labels or leaf nodes. Figure 5 illustrates our method. The tree in the left side of this figure is the skeleton obtained from the decision tree in Figure 4 by discarding the aforementioned pieces of information.

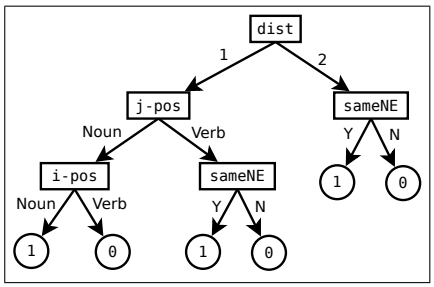

Figure 4
A decision tree.

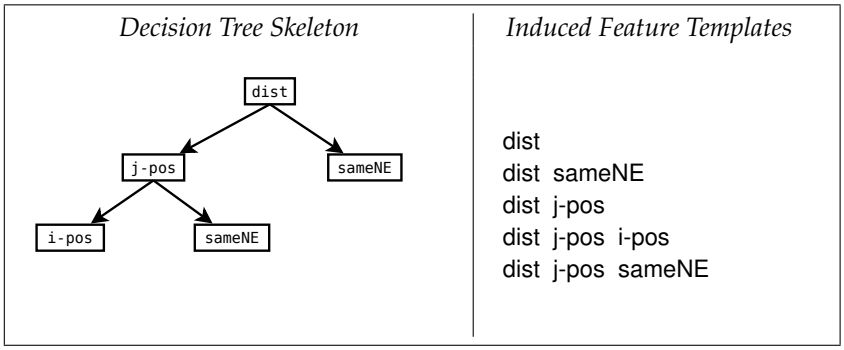

Figure 5
Feature template induction from a decision tree.

Its nodes are basic features with high discriminative power. The generated templates are listed on the right side of the figure. In other words, we create a template with the features in each path from the root node to every other internal node in the given decision tree. Additionally, we eliminate duplicate templates.

Because C4.5 greedily chooses the feature with the highest information ratio at each step, our method generates feature templates with high discriminative power based on entropy. This method can provide a very large number of templates. Hence, to limit the maximum template length, we use C4.5 pruned trees and limit the maximum template length when traversing the DT. This parameter is clearly task-dependent and must be calibrated by cross-validation or on a development set.

Finally, we utilize the generated templates to induce the *binary* contextual features that occur in the structured data set $\mathcal{D}$. For each template, we generate many binary features. These derived features comprise the structured model feature vectors $\boldsymbol{\Phi}(\boldsymbol{x}, e) = (\phi_1(\boldsymbol{x}, e), \ldots, \phi_M(\boldsymbol{x}, e))$. For instance, one of the derived features for the template *dist j-pos sameNE* is given by

$$\phi_m(\boldsymbol{x}, e) = \begin{cases} 1 & \text{if } \textit{dist=2} \text{ and } \textit{j-pos=Noun} \text{ and } \textit{sameNE=N}, \\ 0 & \text{otherwise.} \end{cases}$$

Note that this feature captures a context that is not used by the DT in Figure 4. Indeed, we eliminate the DT feature values when generating the templates and then instantiate these templates based on every context that occurs in a training example.

Thus, the derived feature vector $\boldsymbol{\Phi}(\boldsymbol{x}, e)$ is a binary vector whose 1-valued positions indicate the derived features that are *active* in arc e. These active derived features depend on the values of the basic features $\boldsymbol{\Psi}(e)$. The number of derived features M is very large because several combinations of basic feature values arise for each template. However, the number of active derived features in a specific arc is much smaller. In fact, for each arc, there is one active derived feature for each template; all others are set to zero.

Observe that integer basic features are transformed into categorical features before running the DT algorithm. For such features, each integer value is considered a different category. This approach works better than slicing the value range of these features, as the integer features considered in our system have quite narrow ranges.

6. Large Margin Latent Tree Learning

In our learning set-up for the mention clustering subtask, we are given a training set $\mathcal{D} = \{(\boldsymbol{x}, \boldsymbol{y})\}$, where each example consists of a mention set $\boldsymbol{x}$ and its corresponding mention clustering $\boldsymbol{y}$. Beforehand, we also generate the candidate pairs graph $G(\boldsymbol{x})$ for each example, as described in Section 4. The generation of these graphs for the CoNLL-2012 data sets is detailed in Section 7.2. Observe that document trees are not given in $\mathcal{D}$. Thus, we assume that these structures are *latent* and use the latent structured perceptron algorithm (Sun et al. 2009; Yu and Joachims 2009; Fernandes and Brefeld 2011) to train our models.

First, we predict the **constrained document tree** $\tilde{\boldsymbol{h}}$ for the training instance $(\boldsymbol{x}, \boldsymbol{y})$, using a specialization of the document tree predictor, the **constrained tree predictor** $F_h(\boldsymbol{x}, \boldsymbol{y})$. This predictor finds a maximum scoring document tree for $\boldsymbol{x}$ that follows the correct clustering $\boldsymbol{y}$. Therefore, its search is constrained to a subset $\mathcal{H}(\boldsymbol{x}, \boldsymbol{y})$ contained in $\mathcal{H}(\boldsymbol{x})$.

The constrained tree predictor is given by

$$\tilde{\boldsymbol{h}} = F_h(\boldsymbol{x}, \boldsymbol{y}) = \arg \max_{h \in \mathcal{H}(\boldsymbol{x}, \boldsymbol{y})} \langle \boldsymbol{w}_t, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{h}) \rangle$$

where $\boldsymbol{w}_t$ comprises the model parameters in the t-th training iteration. The trees in $\mathcal{H}(\boldsymbol{x}, \boldsymbol{y})$ are subgraphs of the *constrained* candidate pairs graph $G(\boldsymbol{x}, \boldsymbol{y})$, which is obtained by removing intercluster arcs regarding the correct clustering $\boldsymbol{y}$ from the candidate pairs graph $G(\boldsymbol{x})$. Regarding the artificial arcs, for each cluster in $\boldsymbol{y}$, $G(\boldsymbol{x}, \boldsymbol{y})$ includes only one arc that connects the artificial node to the first mention of the cluster. Hence, the constrained document tree can only include arcs between mentions that lie in the same cluster plus one arc from the artificial root node to each cluster. We use the constrained document tree $\tilde{\boldsymbol{h}}$ as the ground truth in the current perceptron iteration.

It is worth noting that, as we heuristically filter arcs out of $G(\boldsymbol{x})$, a correct cluster can be disconnected in $G(\boldsymbol{x}, \boldsymbol{y})$. In this situation, the prediction of a completely correct clustering is impossible; that is, $F_y(F_h(\boldsymbol{x}, \boldsymbol{y})) \neq \boldsymbol{y}$ for any model parameters. However, such cases are rare, given that the recall of the used sieves is approximately 90%. Moreover, the essential arcs are missing in both $G(\boldsymbol{x})$ and $G(\boldsymbol{x}, \boldsymbol{y})$, and thus the constrained document tree $\tilde{\boldsymbol{h}}$, which is used as the ground truth, corresponds to the best possible clustering given the candidate arcs.

Next, we predict the document tree $\hat{\boldsymbol{h}}$ for the training instance $(\boldsymbol{x}, \boldsymbol{y})$, using a *large margin* version of the document tree predictor. This modified predictor uses a large margin trick (Tsochantaridis et al. 2005; Fernandes and Milidiú 2012) that embeds a loss function into the prediction problem. As is usual in such methods, the large margin training improves the quality of the learned model. In Section 8, we present experimental evidence of this behavior. The large margin predictor for a training example $(\boldsymbol{x}, \boldsymbol{y})$ is given by

$$\hat{\boldsymbol{h}} = \arg \max_{\boldsymbol{h} \in \mathcal{H}(\boldsymbol{x})} \langle \boldsymbol{w}_t, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{h}) \rangle + C \cdot \ell_r(\boldsymbol{h}, \tilde{\boldsymbol{h}})$$

where ℓ_r is a non-negative loss function that measures how much a document tree $\boldsymbol{h}$ differs from the constrained document tree $\tilde{\boldsymbol{h}}$, which is the ground truth for the current iteration. The loss function measures the impurity in the predicted document tree, and C is a loss parameter tuned on the development set. In our model, we use a loss function

that simply counts how many arcs are different in a given tree $\boldsymbol{h}$ compared to the current iteration ground truth $\tilde{\boldsymbol{h}}$, that is,

$$\ell_r(\boldsymbol{h}, \tilde{\boldsymbol{h}}) = \sum_{j=1,\ldots,N} \mathbf{1}_{[\boldsymbol{h}(j) \neq \tilde{\boldsymbol{h}}(j)]} \cdot (1 + r \cdot \mathbf{1}_{[\boldsymbol{h}(j)=0]})$$

where $\mathbf{1}_{[q]}$ is equal to 1 if predicate q is true and 0 otherwise; $\boldsymbol{h}(j)$ indicates the parent of mention j in tree $\boldsymbol{h}$; and r is the *root loss value* that scales the loss value for artificial arcs. This large margin trick is only applied during training because its purpose is to learn a model that separates the ground truth tree from any alternative tree by a distance proportional to the loss of the alternative tree. The greater the loss $\ell_r(\boldsymbol{h}, \tilde{\boldsymbol{h}})$ of a candidate tree $\boldsymbol{h}$, the smaller its score $\langle \boldsymbol{w}_t, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{h}) \rangle$ must be to avoid model updates during the current iteration.

Observe that to find $\tilde{\boldsymbol{h}}$ and $\hat{\boldsymbol{h}}$, we use the Chu-Liu-Edmonds algorithm with different arc sets and, due to the loss function, different arc costs. Nevertheless, the optimization algorithm is identical in both cases because the loss function can be factored along arcs. Thus, in the large margin predictor, we simply add the loss function values to the original arc weights. Then, we use the Chu-Liu-Edmonds algorithm to predict the optimum tree on the graph with modified arc weights. Different loss functions can be used in the large margin predictor. However, if the loss function does not factor along arcs, the prediction problem will be different, and, consequently, another optimization algorithm will be required. For instance, to use the CoNLL metrics as loss functions may be practically infeasible. These metrics are based on very strong, global dependencies along the output structure, and the resulting optimization problem will thus be much harder. Our simple loss function relies on the latent trees to be effective and computationally efficient.

The model update is determined by the difference between the constrained document tree $\tilde{\boldsymbol{h}}$ and the document tree $\hat{\boldsymbol{h}}$ predicted by the large margin prediction algorithm, that is,

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \boldsymbol{\Phi}(\boldsymbol{x}, \tilde{\boldsymbol{h}}) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{h}})$$

In Figure 6, we depict the proposed averaged large margin latent structured perceptron algorithm for the mention clustering subtask. Similar to its univariate counterpart (Rosenblatt 1957), the averaged large margin latent structured perceptron is an

$\boldsymbol{w}_0 \leftarrow \mathbf{0}$
$t \leftarrow 0$
while no convergence
  for each $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$
    $\tilde{\boldsymbol{h}} \leftarrow \arg\max_{h \in \mathcal{H}(\boldsymbol{x}, \boldsymbol{y})} \langle \boldsymbol{w}_t, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{h}) \rangle$
    $\hat{\boldsymbol{h}} \leftarrow \arg\max_{h \in \mathcal{H}(\boldsymbol{x})} \langle \boldsymbol{w}_t, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{h}) \rangle + C \cdot \ell_r(\boldsymbol{h}, \tilde{\boldsymbol{h}})$
    $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \boldsymbol{\Phi}(\boldsymbol{x}, \tilde{\boldsymbol{h}}) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{h}})$
    $t \leftarrow t + 1$
$\boldsymbol{w} \leftarrow \frac{1}{t} \sum_{i=1}^{t} \boldsymbol{w}_i$

Figure 6
Averaged large margin latent structured perceptron algorithm.

on-line algorithm that iterates through the training set. For each training instance, it uses the given input and the current model estimate to perform three major steps: (i) a constrained document tree prediction $\tilde{\boldsymbol{h}}$, which we call the iteration ground truth; (ii) an output prediction $\hat{\boldsymbol{h}}$; and (iii) a model update based on the difference between the predicted output features and the current iteration ground truth features. We use the *averaged* structured perceptron, as suggested by Collins (2002), because it provides a more robust model.

The latent structured perceptron algorithm learns to predict document trees that assist in the target task, which is clustering. Thereafter, for the prediction of an unseen document $\boldsymbol{x}$, the document tree predictor $F_h(\boldsymbol{x})$ is applied using the learned model $\boldsymbol{w}$. It finds the maximum scoring document tree $\hat{\boldsymbol{h}} \in \mathcal{H}(\boldsymbol{x})$ over the given candidate pairs graph by applying the Chu-Liu-Edmonds algorithm. Its output, in turn, is fed to $F_y(\hat{\boldsymbol{h}})$ to finally give the predicted clusters.

7. Empirical Evaluation Setting

We evaluate our system using the data sets from the CoNLL-2012 Shared Task. In this section, we first briefly describe these data sets. Next, we detail the system settings adopted in the experiments for different steps of our system, including candidate pair generation, basic feature setting, and entropy-guided feature induction. We also describe the minor adaptations that are necessary for each language. Finally, we present the evaluation metrics used to assess our system.

7.1 Data Sets

The CoNLL-2012 Shared Task is dedicated to the modeling of unrestricted coreference resolution in three languages: Arabic, Chinese, and English. Its data sets are provided by the OntoNotes project (Weischedel et al. 2011). In addition to coreference information, the shared task data sets contain various annotation layers, namely, part-of-speech (POS) tags, syntactic parses, word senses, named entities (NE), and semantic roles (SRL). The task consists of the automatic identification of coreferring mentions of entities and events given the predicted information on other layers.

The data set for each language comprises three subsets: training, development, and test. We report our system performances on the development and test sets. The development results are obtained using systems that have been trained only on the training sets. However, the test set results are obtained by training on a larger data set that is obtained by concatenating the training and development sets. During training, we use the gold standard input features, which produce better performing models than do those trained on the automatic values. We believe that the use of gold values during training avoids the additional noise introduced by automatic features. However, during evaluation, we always use the automatic values provided in the data sets.

7.2 Candidate Pair Generation

The input for the prediction problem is the candidate pairs graph $G(x)$. Recall that for a given input $\boldsymbol{x} = (\boldsymbol{d}, \boldsymbol{m})$, there is a node in $G(x)$ for each mention in $\boldsymbol{m}$. Ideally, we could consider the complete graph for each document, where every mention pair would be an option for building the document tree. However, for a document with many mentions, the resulting graph might be very large; moreover, a large portion of its arcs can be

easily identified as unnecessary or even incorrect. Thus, we filter the mention pairs that are less likely to be coreferent out of the candidate pairs graph.

We choose the candidate arcs by simply adapting the sieves method proposed by Lee et al. (2013) to English coreference resolution. Lee et al. propose a list of handcrafted rules that are sequentially applied to mention pairs to iteratively merge mentions into entity clusters. These rules are termed **sieves** because they filter the correct mention pairs. In Lee et al.'s system, sieves are ordered from higher to lower precision. However, in our filtering strategy, precision is not a concern, and the application order is not important. The objective here is to build a small set of candidate arcs that show good recall. Additionally, we do not want sieves that are strongly language dependent, as our target is multilingual coreference resolution. Hence, we select the most general of Lee et al.'s sieves. The resulting sieves can also be directly applied to the Arabic and Chinese data sets provided in the CoNLL-2012 Shared Task. Therefore, given a mention pair (i, j), where i appears before j in the text, we create a directed arc $e = (i, j)$ if *at least one* of the following conditions holds:

1. *Distance* – the number of mentions between i and j is not greater than a given threshold.
2. *Named Entity Alias* – i and j are named entities of the same kind, plus:
   (a) *Person* – the head word of one mention is part of the other mention, such as *Obama* and *Barack Obama*.
   (b) *Organization* – the head word of one mention is contained in the other, or one is an acronym of the other.
3. *Head Word Match* – the head word of i matches the head word of j.
4. *Shallow Discourse* – shallow discourse attributes match for both mentions. This sieve comprises a set of rules proposed by Lee et al. (2013) based on mention and speaker attributes that are available in the data set. Basically, the number and gender of speakers and mentions must follow a small set of patterns. For instance, two first-person pronouns assigned to the same speaker is a match.
5. *Pronouns* – j is a pronoun, and i has the same gender, number, speaker, and animacy as j, using the number and gender data from Bergsma and Lin (2006).
6. *Pronouns and NE* – j is a pronoun, and i is a compatible pronoun or named entity.

Sieves 2–6 are adapted from Lee et al. (2013). Most of these sieves are relaxed versions of the ones proposed by Lee et al. (2013). Sieve 1 is introduced by us to raise recall while avoiding strongly language-dependent sieves.

7.3 Basic Feature Setting

We use 70 basic features to describe a candidate arc. All of them give hints on the coreference strength of the mention pair connected by the arc. These features provide lexical, syntactic, semantic, and positional information. One of the semantic features is the prediction of the baseline system proposed by dos Santos and Carvalho (2011), which classifies the candidate arc as either coreferent or not. The other features have

been adapted from previously proposed features (Ng and Cardie 2002; Sapena, Padró, and Turmo 2010; dos Santos and Carvalho 2011).

In Table 2, we describe the set of basic features used in our system. The *Id* column identifies each feature. The *Type* column indicates the value type of each feature, for example, Boolean (*yes*, *no*) or ternary (*yes*, *no*, *not applicable*). The # column indicates how many basic features correspond to each description.

Table 2
Description of all 70 basic features.

Id	Description	Type	#
	Lexical Features		*25*
L1	Head word of *i* (*j*)	word	2
L2	String matching of *i* and *j*	boolean	1
L3	String matching of the head words of *i* and *j*	boolean	1
L4	Both *i* and *j* are pronouns and their strings match	ternary	1
L5	Both *i* and *j* are *not* pronouns and their string match	ternary	1
L6	Previous and next two words of *i* (*j*)	word	8
L7	Number of tokens in *i* (*j*)	integer	2
L8	Edit distance of head words of *i* and *j*	integer	1
L9	Edit distance of *i* and *j* after removing determiners	integer	1
L10	*i* (*j*) is a definite noun phrase	boolean	2
L11	*i* (*j*) is a demonstrative noun phrase	boolean	2
L12	The head word of both *i* and *j* are proper nouns	boolean	1
L13	Both *i* and *j* are proper names and their strings match	ternary	1
L14	Both *i* and *j* are proper names and their head word strings match	ternary	1
	Syntactic Features		*28*
Sy1	POS tag of the head word of *i* (*j*)	POS tag	2
Sy2	Previous and next two POS tags of *i* (*j*)	POS tag	8
Sy3	*i* (*j*) is a pronoun	boolean	2
Sy4	Gender of *i* (*j*), if pronoun	f, m, n/a	2
Sy5	*i* and *j* are both pronouns and agree in gender	ternary	1
Sy6	*i* and *j* are both pronouns and agree in number	ternary	1
Sy7	*i* (*j*) is a proper name	boolean	2
Sy8	*i* and *j* are both proper names	boolean	1
Sy9	Previous and next predicate of *i* (*j*) within its sentence	verb	4
Sy10	*i* and *j* are pronouns and agree in number, gender, and person	ternary	1
Sy11	Noun phrase embedding level of *i* (*j*) in the syntactic parse	integer	2
Sy12	Number of embedded noun phrases in *i* (*j*)	integer	2
	Semantic Features		*13*
Se1	Baseline system prediction	binary	1
Se2	Sense of the head word of *i* (*j*)	sense	2
Se3	Named entity type of *i* (*j*)	NE tag	2
Se4	*i* and *j* have the same named entity type	ternary	1
Se5	Previous and next semantic roles for *i* (*j*) within its sentence	SRL tag	4
Se6	Concatenation of semantic roles of *i* and *j* for the same predicate, if they are in the same sentence	$(\text{SRL tag})^2$	1
Se7	*i* and *j* have the same speaker	ternary	1
Se8	*j* is an alias of *i* as suggested by Ng and Cardie (2002)	boolean	1
	Positional Features		*4*
P1	Distance between *i* and *j* in number of sentences	integer	1
P2	Distance between *i* and *j* in number of mentions	integer	1
P3	Distance between *i* and *j* in number of person names, when *i* and *j* are both pronouns or one of them is a person name	integer	1
P4	One mention is in apposition to the other	boolean	1

Although our feature set does not include negative features such as Ng and Cardie's (2002) binding constraints and maximal NP, this type of feature can be used in our system. An arc that activates one or more negative features is highly unlikely to be in the solution. In other words, if a mention pair has the value *yes* for one or more negative features, the two mentions are unlikely to be coreferent. When using negative features, the structured perceptron will most likely learn negative weights for these features. Consequently, arcs that activate negative features have a lower chance of inclusion in the solution because these arcs tend to decrease the coreference tree weight.

7.4 Language Specifics

Our system can be easily adapted to different languages given the basic features. In our experiments, only minor changes are needed to train and apply the system to three different languages. The adaptations are due to (i) the lack of input features for some languages; (ii) the different POS tagsets across data sets; and (iii) the creation of static lists of language-specific pronouns. The necessary adaptations are restricted to only two preprocessing steps: basic features and coreference arcs generation.

Some input features available in the English data set are not available in the Arabic or Chinese data sets. The Arabic data set does not contain named entity, semantic role labeling, and speaker annotations. Therefore, for Arabic, we do not derive the following basic features: Sy9, Se3, Se4, Se5, Se6, Se7, and P3. For Chinese, information related to named entities is not provided. Thus, we do not derive the following basic features: Se3, Se4, and P3. Additionally, the Chinese data set uses a different POS tagset. Hence, some mappings are used during the basic feature derivation stage.

The lack of input features for Arabic and Chinese also impact the sieve-based candidate pair generation step. For Chinese, we do not use Sieve 5, and, for Arabic, we only use Sieves 1, 3, and 6. Sieve 6 is not used for the English data set because it is a specialization of Sieve 5. The first sieve threshold is 4 for Arabic and Chinese and 8 for English. These values have been chosen to optimize recall in the development sets and simultaneously fit the training data sets in memory.

In the arc generation and basic feature derivation steps, our system uses static lists of language-specific pronouns. In our experiments, we use the POS tagging information and the gold entity clusters to automatically extract these pronoun lists from the training data.

Our system submitted to the CoNLL-2012 Shared Task ignores arcs linking *nested* mentions. Although such mentions are never coreferent in Arabic or English, the Chinese data sets include many nested coreferring mentions. Hence, in the latest version of our system, we include such arcs for the Chinese language.

7.5 EFI

Using EFI, we can automatically generate feature templates without human effort, which allows us to easily experiment with different template sets. In our experiments, we use 70 basic features and apply EFI to generate feature templates from them. Only 58 basic features appear at least once in the template set of any of the three languages. In decreasing frequency order, the ten most frequent basic features in the templates are Se1, L13, L3, $L1_j$, Se7, $Sy3_j$, $L1_i$, Se4, $Sy12_i$, and $Sy3_i$. These features appear closest to the root in the EFI auxiliary decision tree. When following our template generation method, they are the most discriminative features. Twelve of the 70 basic features do not appear in any template, namely, Sy4, $Sy7_j$, Sy9, Se5, and Se6. Thus, these features

are not used when training with the structured perceptron algorithm. Note that features Sy4 and $Sy7_j$ carry some information that is already present in other features, such as Sy10 and L12, respectively. Features Sy9, Se5, and Se6 are all derived from the SRL basic features. Therefore, when following our template generation strategy, SRL information contributes little to the coreference resolution task.

In our first experiments, for each language, we create a template set by applying EFI to the training corpus of that language. However, merging different template sets usually produces better results, even when merging template sets of different languages. For instance, for the Chinese and Arabic languages, merging their templates sets with a template set generated for the English language produces an improvement of approximately 2% on the MUC F-score. We believe this behavior to be due to the greedy nature of the decision tree induction algorithm used in EFI and the variation in annotation quality and size of the different data sets. It is quite likely that the automatically generated feature templates are not optimal.

In our final English language system, we use a set of 196 templates obtained by merging the output of two independent EFI executions. These two runs are based on two different training data sets comprising (a) the mention pairs produced by Sieves 2–5; and (b) the mention pairs produced by all sieves. For Chinese and Arabic, the template sets automatically generated using the corresponding training data sets are merged with template set (a) generated for the English language. The final set for Chinese comprises 197 templates, and the final set for Arabic comprises 223 templates.

Note that the number of templates generated by EFI depends on the number of basic features, the size of the training set, and the settings of the decision tree algorithm. The number of feature templates has a direct impact on memory use, as increasing the number of templates increases the number of binary contextual features instantiated. Hence, our main restriction when automatically creating and merging feature template sets is memory. Using the template sets previously mentioned, we are able to train our English system, which corresponds to the largest data set, using approximately 48 GB RAM. For the Arabic and Chinese systems, less than 20 GB RAM is used during training.

7.6 Evaluation Metrics

Evaluating coreference systems is a difficult task. The main issue is that coreference information is highly faceted, and the value of each facet varies substantially from one application to another. Thus, when reporting and comparing coreference system performances, it is very difficult to define *one* metric that fits all purposes. Therefore, we follow the methodology proposed in the CoNLL-2012 Shared Task to assess our systems, as it combines three of the most popular metrics. The metrics used are the link-based MUC metric (Vilain et al. 1995), the mention-based B^3 metric (Bagga and Baldwin 1998), and the entity-based $CEAF_e$ metric (Luo 2005). All these metrics are based on precision and recall measures, which are combined to produce an F-score value. The mean F-score of these three metrics gives a unique score for each language. Additionally, when appropriate, the official CoNLL-2012 Shared Task score is reported, which is the average of the F-scores for all languages. We denote this metric as the **CoNLL score**.

Another important aspect of coreference evaluation is mention matching. Some methodologies, such as the ones used in the MUC and ACE evaluations, consider approximate matching of mention spans. However, the CoNLL-2012 Shared Task evaluation considers only exact span matching.

We use the CoNLL-2012 Shared Task evaluation scripts version 4, which is the same version used to produce the official ranking in this shared task. We use this version

to keep our results comparable with most of the results reported in the literature. It is important to note that, in early 2014, a committee of researchers revised some of the evaluation metrics and released a new version of these scripts.[1] Although this revision changes the absolute scores, the ranking of the top performing systems, including ours, remain the same.

8. Empirical Results

In this section, we present nine sets of empirical findings on the CoNLL-2012 Shared Task data sets. In Section 8.1, we demonstrate our system's overall quality and compare it with state-of-the-art systems. In Section 8.2, we assess the mention detection step. Section 8.3 details the results of the candidate pair generation step. In Section 8.4, we assess EFI, showing that this method significantly improves the resulting system performance. In Section 8.5, we present experimental evidence that latent trees improve the prediction performance of our system compared to simple static structures. In Section 8.6, we show that the root loss value also significantly improves our system performances. Section 8.7 presents results that show the contribution of the large margin trick. In Section 8.8, we show that by enhancing our Chinese modeling with nested mentions, we achieve state-of-the-art performance for this language. Finally, in Section 8.9, we present a detailed analysis of the different types of errors in our system output.

In the reported experiments, we use a computer with a 2.4-GHz quad-core processor and 48 GB RAM. We report the processing times for the English data sets. For the other languages, these times are proportionally lower. First, regarding training, the feature induction procedure takes 27.4 minutes; loading the training data set and generating features from the learned templates take 7.5 minutes; and the training algorithm takes 57.5 minutes. The test procedure on the development set takes 6 seconds to load the data set, 45 seconds to load the model and to generate features from templates, and 30 seconds to predict the output structures for all documents. Most of these procedures could be further optimized in terms of both execution time and memory use.

Regarding the size of the learned models, the Arabic model has approximately 8.1 million parameters; the Chinese model has approximately 9.7 million parameters; and the English model has approximately 15.5 million parameters.

8.1 State-of-the-Art Systems

In Table 3, we present the per-language CoNLL scores of the best performing systems on the CoNLL-2012 Closed Track Shared Task test sets. The first row of this table corresponds to the last version of our system, and the second row corresponds to our official entry in the CoNLL-2012 Shared Task. The difference between these two versions is the inclusion of candidate arcs linking *nested* mentions for the Chinese language. By including such arcs, the score increases by almost 4.5 points for that language. The last two rows of this table correspond to the competitors that are ranked second (Björkelund and Farkas 2012) and third (Chen and Ng 2012) in the shared task. Our system obtains the best score for each language.

From Table 3, it can be further observed that the best scores on Chinese and English are similar. However, the performances on the Arabic language are much lower. Given the smaller size of the Arabic training corpus, this reduction is expected.

1 http://code.google.com/p/reference-coreference-scorers/.

Table 3
State-of-the-art systems for multilingual unrestricted coreference resolution in the CoNLL-2012 Shared Task data sets. Performances on the test sets for Arabic (AR), Chinese (CH), English (EN), and the official shared task score.

Reference	AR	CH	EN	CoNLL Score
This work	**54.22**	**62.87**	**63.37**	**60.15**
Fernandes, dos Santos, and Milidiú (2012)	**54.22**	58.49	**63.37**	58.69
Björkelund and Farkas (2012)	53.55	59.97	61.24	58.25
Chen and Ng (2012)	47.13	62.24	59.69	56.35

The detailed performances of the best available systems for the Arabic CoNLL-2012 Shared Task test set are presented in Table 4, where we report the recall (R), precision (P), and F-score (F) for the three metrics considered in the CoNLL-2012 Shared Task. Additionally, in the final column of this table, we present the mean F-score over the three metrics, which gives a unique per-language score. In these results, we see that the runner-up system outperforms our system by approximately 1.4 points on both the MUC and B^3 F-scores. However, our system outperforms theirs on $CEAF_e$ by almost 5 points. In the end, our system achieves a mean score that is 0.67 point higher than Björkelund and Farkas' system. The third-ranked system (Uryupina, Moschitti, and Poesio 2012) scores much lower, being competitive only on the B^3 metric, in which it achieves a substantially higher recall.

The detailed results of the best performing systems for the Chinese language are presented in Table 5. For this language, our system is competitive on all metrics and particularly on MUC, where our system outperforms all competitors by more than 2.5 points of F-score. Note that this result is achieved by considering the nested mentions.

In Table 6, we present the detailed performances of the state-of-the-art systems for the English language. Our system outperforms all others by more than 2 points on the mean score. Moreover, it achieves the best F-scores on the three metrics used, being

Table 4
Detailed performance of state-of-the-art systems on the *Arabic* CoNLL-2012 Shared Task test set.

System	MUC			B^3			$CEAF_e$			Mean
	R	P	F	R	P	F	R	P	F	
This work	43.63	49.69	46.46	62.70	72.19	67.11	**52.49**	**46.09**	**49.08**	**54.22**
Björkelund and Farkas (2012)	**43.90**	**52.51**	**47.82**	62.89	**75.32**	**68.54**	48.45	40.80	44.30	53.55
Uryupina et al. (2012)	41.33	41.66	41.49	**65.77**	69.23	67.46	42.43	42.13	42.28	50.41

Table 5
Detailed performance of state-of-the-art systems on the *Chinese* CoNLL-2012 Shared Task test set.

System	MUC			B^3			$CEAF_e$			Mean
	R	P	F	R	P	F	R	P	F	
This work	59.20	**71.52**	**64.78**	67.17	**80.55**	73.25	**57.46**	45.20	50.59	**62.87**
Chen and Ng (2012)	59.92	64.69	62.21	69.73	77.81	**73.55**	53.43	48.73	**50.97**	62.24
Yuan et al. (2012)	**62.36**	58.42	60.33	**73.12**	72.67	72.90	47.10	**50.70**	48.83	60.69
Björkelund and Farkas (2012)	58.72	58.49	58.61	71.23	75.07	73.10	48.09	48.29	48.19	59.97

Table 6
Detailed performance of state-of-the-art systems on the *English* CoNLL-2012 Shared Task test set.

System	MUC			B^3			$CEAF_e$			Mean
	R	P	F	R	P	F	R	P	F	
This work	**65.83**	**75.91**	**70.51**	65.79	**77.69**	**71.24**	**55.00**	43.17	**48.37**	**63.37**
Martschat et al. (2012)	65.21	68.83	66.97	**66.50**	74.69	70.36	48.64	**44.72**	46.60	61.31
Björkelund and Farkas (2012)	65.23	70.10	67.58	65.90	75.24	70.26	48.60	43.42	45.87	61.24

outperformed only by Martschat et al.'s system on the B^3 recall and $CEAF_e$ precision values. These achievements are sound, especially considering the large research effort applied to the English language.

8.2 Mention Detection

The first step of our coreference resolution system is mention detection, where we seek to detect all possible mentions in the given document. In this preliminary step, recall is much more important than precision because missing mentions correspond to unrecoverable errors for the subsequent steps. Moreover, in the CoNLL-2012 unrestricted coreference resolution task, singleton mention clusters are treated as precision errors. Thus, the performance of our mention detector must be checked only on the set of non-singleton mentions.

In our system, we remove singleton mentions only after the mention clustering step. Hence, for each model, we assess mention detection twice: before and after the mention clustering step. The first evaluation is performed immediately after the mention detection step but before the mention clustering step, when our system has neither identified nor excluded singleton mentions. The second evaluation is performed on the final output of our system, when clusters comprising only one mention have been deleted. In Table 7, we present these performances on the development sets of the three languages. The *Total* column indicates how many mentions are annotated in the gold standard data set, which comprises non-singletons only. The *Extracted* column indicates the number of mentions detected by our system, and the *Correct* column indicates how many of them are correct. The columns R, P, and F correspond, respectively, to the recall, precision, and F-score of mentions. As in the CoNLL-2012 Shared Task, a mention is considered correct only when its exact span has been detected.

Table 7
Mention detection performances on the development sets before and after mention clustering.

Language	Total	After Mention Clustering?	Extracted	Correct	R	P	F
Arabic	3,316	No	16,209	2,916	87.93	17.99	29.86
		Yes	3,094	2,036	61.39	65.80	63.52
Chinese	14,183	No	39,475	12,558	88.54	31.81	46.80
		Yes	10,655	8,346	58.84	78.32	67.20
English	19,155	No	55,738	17,746	92.64	31.83	47.39
		Yes	16,738	13,795	72.01	82.41	76.86

In Table 7, we can observe that before mention clustering, the precision is very low for all languages: our system extracts many singleton mentions because it privileges recall over precision. Conversely, it correctly detects approximately 90% of the non-singleton mentions for all languages. After the mention clustering step, the mention detection precision increases because many correctly identified singletons are excluded. Of course, some non-singleton mentions are wrongly identified as singletons in the mention clustering step, and there is a consequent drop in recall.

8.3 Candidate Pair Generation

The candidate pair generation step is responsible for creating the candidate pairs graph $G(\boldsymbol{x})$ given the set of detected mentions. Our approach to this subtask is based on the sieves proposed by Lee et al. (2013). The purpose of using sieves in this step is twofold. First, when generating $G(\boldsymbol{x})$, we want to include only the arcs that link mentions that are likely to be coreferent to avoid dealing with a dense graph in the subsequent steps, which involve arc feature generation and solving maximum branching problems on $G(\boldsymbol{x})$. Second, we do not want to generate too many precision errors for the subsequent steps by linking mentions that are not likely to be coreferent. Conversely, the sieves cannot be too restrictive, or they would miss too many intracluster arcs and put coreferent mentions in different connected components. If two coreferent nodes m_1 and m_2 belong to different connected components in $G(\boldsymbol{x})$, the coreference tree predictor is unable to generate a tree on $G(\boldsymbol{x})$ containing the coreferent nodes m_1 and m_2. Therefore, there is a trade-off between the restrictiveness of the sieves and the coreference resolution recall. In this subsection, we evaluate the MUC recall of the candidate arcs in $G(\boldsymbol{x})$, which corresponds to the upper bound for our coreference resolution system when this graph is given as input to the tree predictor.

In Tables 8, 9, and 10, we report the impact of different combinations of sieves on the MUC recall of the candidate arcs for the Arabic, Chinese, and English development sets, respectively. These results help us to understand the contribution of each sieve and the overlap among them. It is important to note that, for each document in the development set, one distinct candidate pairs graph is generated. Hence, in Tables 8–10, we show an overall assessment of the graphs generated for all documents in the respective development sets. The first row of each table corresponds to the case where all graphs contain all possible candidate pairs, which means filtering no arcs out. The recall errors in this case are due to missing mentions, which means that these errors occur in the mention detection step. In addition to recall, for each sieve set configuration, we report the recall percentage (% Recall), which gives the percentage of the maximum recall (first row) achieved by the configuration. The tables also present the number of

Table 8
Performances of the Arabic sieves on the development set.

Sieves	Recall	% Recall	# Arcs	% Arcs
All arcs	81.10	100.0	3,139,572	100.0
(6)	27.17	33.5	60,579	1.9
(1)	41.03	50.6	139,099	4.4
(3)	41.41	51.1	26,487	0.8
(3) and (1)	64.38	79.4	164,017	5.2
(3), (1), and (6)	66.56	82.1	216,252	6.9

Table 9
Performances of the Chinese sieves on the development set.

Sieves	Recall	% Recall	# Arcs	% Arcs
All arcs	84.14	100.0	6,193,390	100.0
(4)	7.45	8.9	4,596	0.1
(2)	18.78	22.3	17,964	0.3
(6)	20.78	24.7	50,432	0.8
(1)	41.62	49.5	260,590	4.2
(3)	65.86	78.3	83,523	1.3
(3) and (1)	75.85	90.1	335,447	5.4
(3), (1), and (6)	77.32	91.9	373,820	6.0
(3), (1), (6), and (2)	77.57	92.2	377,751	6.1
(3), (1), (6), (2), and (4)	77.58	92.2	377,851	6.1

Table 10
Performances of the English sieves on the development set.

Sieves	Recall	% Recall	# Arcs	% Arcs
All arcs	90.22	100.0	7,198,159	100.0
(2)	9.82	10.9	7,628	0.1
(4)	14.30	15.9	26,561	0.4
(5)	41.98	46.5	69,143	1.0
(3)	54.45	60.4	94,571	1.3
(1)	66.01	73.2	615,909	8.6
(1) and (3)	83.01	92.0	696,370	9.7
(1), (3), and (5)	83.93	93.0	757,913	10.5
(1), (3), (5), and (4)	84.12	93.3	763,585	10.6
(1), (3), (5), (4), and (2)	84.54	93.7	764,517	10.6

generated candidate arcs and the percentage of arcs in relation to the maximum number of arcs. Note that for the Chinese language, using only 6.1% of the possible arcs, we achieve 92.2% of the maximum recall. For the English language, we achieve 93.7% of the maximum recall using only 10.6% of the possible arcs. These results demonstrate that our sieve sets for these two languages achieve a good trade-off between the restrictiveness of sieves and the coreference resolution recall. For the Arabic language, there is still considerable room for improvement in the aforementioned trade-off.

8.4 EFI

We use entropy-guided feature induction to automatically generate nonlinear features by conjoining the 70 basic features. In this section, we compare our EFI-based system with systems trained using basic features alone. It is important to note that these 70 basic features include several complex features. Some of these features are even conjunctions of simpler basic features, and others provide complex task dependent information, such as head words and agreement on number and gender. These 70 basic features were manually generated by domain experts and encode valuable coreference information.

In Table 11, we report the performance of a system trained with the 70 basic features alone (upper half) and the performance of our EFI-based system (lower half). We observe that the CoNLL score improves impressively, by 8.54 points, when using EFI

Table 11
Impact of EFI on development sets for all languages using all 70 basic features.

EFI	Lang	MUC			B^3			$CEAF_e$			Mean
		R	P	F_1	R	P	F_1	R	P	F_1	
No	Arabic	28.26	48.98	35.84	44.75	74.56	55.93	47.48	26.80	34.26	**42.01**
	Chinese	48.58	73.16	58.39	56.96	83.60	67.76	56.42	34.74	43.00	**56.38**
	English	51.32	73.28	60.37	54.85	78.06	64.43	50.87	30.71	38.30	**54.37**
									CoNLL Score		**50.92**
Yes	Arabic	43.00	47.87	45.30	61.41	70.38	65.59	49.42	44.19	46.66	**52.52**
	Chinese	60.35	70.56	65.05	67.37	79.49	72.93	55.15	44.94	49.52	**62.50**
	English	64.88	74.74	69.46	66.53	78.28	71.93	54.93	43.68	48.66	**63.35**
									CoNLL Score		**59.46**

to generate new features. Moreover, EFI consistently outperforms the baseline on all languages for all metrics.

Informative basic features usually require domain experts to define and select them. This process is expensive and highly language-dependent. To assess EFI's inductive power, we perform another experiment. We randomly remove 30% of the original basic features, retaining only 49 of the 70 basic features. Next, we evaluate the performance impact of this feature reduction on the English development set. Again, we evaluate two systems: one that uses only basic features and another that uses EFI to generate features. We repeat this procedure 20 times and average the results. In Table 12, we present the averaged results for the English development set. The final column of this table presents the standard errors of the three-metric mean. We can see that EFI improves the mean score by almost 6 points.

It is worth noting that, for the EFI results reported in Table 12, we do not merge different template sets, as described in Section 7.5. For this experiment, to simplify the comparison, we use only the template set derived using *all* English sieves.

8.5 Latent Trees

One key modeling aspect of our method consists of representing coreference clusters as latent trees. We already argued in Section 4 that using directed trees to represent coreference clusters is linguistically and computationally plausible. Here, we present experimental evidence that *latent* trees are better than a simple *static* structure in terms of prediction performance. Our findings are aligned with the results reported by Yu and Joachims (2009), who compare undirected latent trees to correlation clustering.

Table 12
Impact of EFI on system performance for the English development set when using 70% of the basic features randomly selected. The reported values are averages of over 20 repetitions of this experiment. The final column indicates the mean standard error.

EFI	MUC			B^3			$CEAF_e$			Mean	Standard
	R	P	F_1	R	P	F_1	R	P	F_1		Error
No	70.89	49.68	58.41	76.77	53.99	63.38	29.90	49.22	37.19	**52.99**	0.25
Yes	70.73	59.17	64.43	75.07	61.94	67.87	39.11	51.57	44.48	**58.93**	0.24

To evaluate the contribution of latent trees to our system performance, we use a simple approach to generate static representations of the coreference clusters in the training set. For a given coreference cluster, we generate a *chain* of mentions by connecting each mention to the previous mention in the same cluster, as in Soon, Ng, and Lim (2001). After generating a chain for each cluster in a document, the first mention of each chain is connected to the artificial root node, resulting in a static document tree. Then, we use these static trees as the golden standard for training. Therefore, the perceptron algorithm learns to predict chains of coreferring mentions instead of general trees. However, for some clusters, due to missing arcs that are filtered out in the candidate pair generation step, it is impossible to generate a chain. In these cases, we connect a mention to the closest previous mention such that the two mentions are in the same cluster, and there is an arc connecting them in the training set. In this experiment, we do not need to use the *Latent* Structured Perceptron training algorithm. Instead, we use the ordinary Structured Perceptron.

In Table 13, we present the performances of two systems on the CoNLL-2012 development sets: the system trained with static chains to represent coreference clusters and our system based on latent trees. We see that latent trees improve the CoNLL score by more than 1 point. Additionally, improvements are observed on all languages for almost all metrics, except for Arabic MUC. Among the three languages, English presents the largest improvement, more than 2 points, when latent trees are used. In contrast, the impact of latent trees for the Arabic corpus is marginal.

It is also important to note that in addition to these performance improvements, latent structures present another benefit. Latent structures simplify modeling because they avoid the need for a heuristic to derive static structures. Latent structures are automatically derived and evolved during training, guided by the annotated clusters.

8.6 Root Loss Value

Just as some coreference metrics can be more important than others for certain applications, precision and recall have different values for applications. Specifically, for the CoNLL score—which is based on the $F_{\beta=1}$ score—the balance between precision and recall is important. For this reason, we introduced an important parameter in our system: the **root loss value**. This parameter specifies a different loss function value for

Table 13
Latent trees' effect on development set performances.

Coref. Tree	Lang	MUC			B^3			$CEAF_e$			Mean
		R	P	F	R	P	F	R	P	F	
Chain	Arabic	43.17	48.42	45.64	60.79	70.16	65.14	48.99	43.40	46.03	**52.27**
	Chinese	60.45	67.76	63.90	68.36	77.01	72.43	52.64	45.44	48.78	**61.70**
	English	62.12	73.66	67.40	64.97	76.29	70.18	52.91	40.34	45.78	**61.12**
									CoNLL Score		**58.36**
Latent	Arabic	43.00	47.87	45.30	61.41	70.38	65.59	49.42	44.19	46.66	**52.52**
	Chinese	60.35	70.56	65.05	67.37	79.49	72.93	55.15	44.94	49.52	**62.50**
	English	64.88	74.74	69.46	66.53	78.28	71.93	54.93	43.68	48.66	**63.35**
									CoNLL Score		**59.46**

outgoing arcs in the artificial root node. Observe that, in a document tree, each arc from the root node corresponds to a cluster. The effect of a root loss value larger than one is to reduce the creation of new clusters, stimulating larger clusters. Therefore, this parameter can be used to adjust the balance between precision and recall.

In the upper half of Table 14, we present our system performances on the development sets when we set this parameter to one, which is equivalent to not using this parameter at all. We observe that in this case, the differences between the recall and precision values are very high on all metrics and languages, lowering the F-score values. Using the development sets for tuning, we set the root loss value to 6, 2, and 1.5 for Arabic, Chinese, and English, respectively. In the lower half of Table 14, we present the performances using these settings. We observe that this parameter substantially improves the balance between precision and recall, consequently increasing the F-score values. Its effect is accentuated on Arabic and Chinese because the unbalancing issue is worse on these languages. The increase in the CoNLL score is nearly 4 points.

8.7 Large Margin and Averaging

In Figure 7, we show the impact on the MUC F-score provided by the large margin trick using our choice for the loss function and the averaged perceptron. Throughout the training procedure, we report the per-epoch performances of the current model on the English development set during the first 50 epochs. Each epoch corresponds to iterating over all instances in the training data set. First, we examine the large margin averaged perceptron with the parameter C equal to 2,000, which is used to train our best model. Second, we simplify by using the averaged perceptron with *no margin*. Using the margin trick, we observe a consistent improvement in all epochs, and in the final model, we obtain a significant improvement of more than 1%. We also can see that when not using the averaging procedure, the algorithm eventually achieves the optimum performance, as occurs after the 25th epoch. However, without averaging, the performance varies considerably from one epoch to another, which can be very harmful. Very similar behavior is observed for the Arabic and Chinese languages.

Additionally, we see that our method seems to converge before 50 epochs, which is the number of epochs used to train all models used in this work.

Table 14
Root loss value effect on development set performances.

Root Loss Value	Lang	MUC			B^3			$CEAF_e$			Mean
		R	P	F	R	P	F	R	P	F	
Off	Arabic	34.18	58.85	43.25	50.61	82.13	62.63	57.37	33.75	42.49	**49.45**
	Chinese	41.94	85.23	56.22	51.25	93.70	66.26	63.10	29.80	40.48	**54.32**
	English	62.75	77.41	69.31	63.88	81.34	71.56	57.46	41.08	47.91	**62.92**
									CoNLL Score		**55.56**
On	Arabic	43.00	47.87	45.30	61.41	70.38	65.59	49.42	44.19	46.66	**52.52**
	Chinese	60.35	70.56	65.05	67.37	79.49	72.93	55.15	44.94	49.52	**62.50**
	English	64.88	74.74	69.46	66.53	78.28	71.93	54.93	43.68	48.66	**63.35**
									CoNLL Score		**59.46**

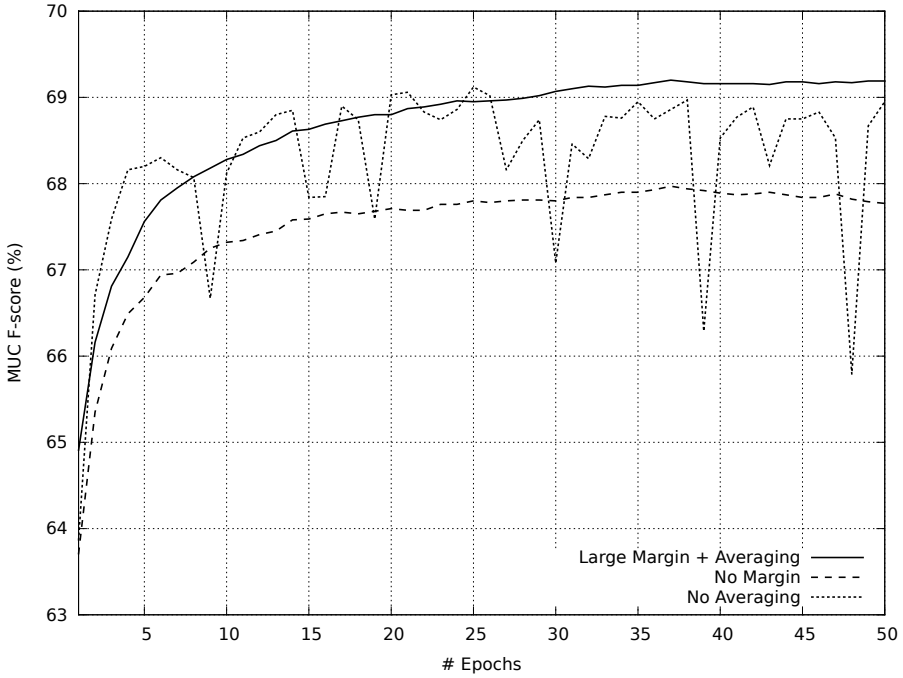

Figure 7
Effect of large margin and averaging on structured perceptron performance.

8.8 Chinese Nested Mentions

Nested noun phrases such as *((the happy boy) from Brazil)* are very common. Considering either all nested noun phrases as coreferring mentions or only the outer noun phrases as mentions is an annotation design choice. In the CoNLL-2012 data sets, for both English and Arabic, only the outer noun phrases are considered as mentions. However, in the Chinese newswire documents, nested mentions are annotated as coreferring (Chen and Ng 2012). Thus, in this work, we evaluate the effect of using arcs linking nested mentions for the Chinese language.

First, we observe that the percentage of candidate arcs linking two nested mentions is 0.41% in the training, 0.34% in the development, and 0.45% in the test subsets of the Chinese data set. These percentages are calculated over the candidate arcs only (i.e., the arcs selected by the Chinese sieves). Thus, the computational impact of adding these arcs is negligible.

Next, we compute two upper bounds that give us insight into the potential improvement provided by nested mentions. For that purpose, we select all *correct* arcs from the candidate ones and compute the MUC recall of these arcs. When nested mentions are not included in the candidate arcs, we obtain a MUC recall of 71.29%. However, when nested mentions are included, we obtain a MUC recall of 77.58%, which is a significant increase in the performance upper bound. These values are computed using the development set.

In Table 15, we present in detail the actual impact of nested mentions on our system performance. The first row shows the results when arcs linking nested mentions are included, and the second row shows the results when they are ignored.

Table 15
Impact of nested mentions on the system performance for the Chinese development set.

Nested	MUC			B^3			$CEAF_e$			Mean
Mentions	R	P	F_1	R	P	F_1	R	P	F_1	
Yes	60.35	70.56	65.05	67.37	79.49	72.93	55.15	44.94	49.52	**62.50**
No	54.40	68.19	60.52	64.17	78.84	70.76	51.42	38.96	44.33	**58.54**

Considering these arcs increases our system score on the Chinese language by almost 4 points.

8.9 Error Analysis

In this section, we provide statistics regarding the most common errors within the outputs of our structure predictor for the development set of each language. Three steps in our predictor are direct sources of errors: mention detection, candidate pair generation, and mention clustering. The errors in each step propagate to the next steps.

In Table 16, we report the MUC recall, precision, and F-score after each of these three key steps. Remember that during mention detection and candidate pair generation, recall is much more important than precision. Hence, we can observe that recall is indeed kept much higher than precision during these steps for all languages. For the Arabic and Chinese languages, the drops in recall are larger than are the drops for English in both preliminary steps. This behavior is because of missing features and sieves for these two languages and also because most of the features and sieves were originally proposed for the English language. In particular, the Arabic candidate pair generation step generates too many recall errors because it is based on only three sieves.

As observed in Section 8.2, the inclusion of singleton mentions is responsible for most of the precision errors during the mention detection step. Our system does not handle coreferring mentions of events, and it thus does not consider verbs when creating candidate mentions, which is responsible for many recall errors in this step. Additionally, there are recall and precision errors due to errors in the automatically

Table 16
Performances on the development sets per system step.

Language	Step	Recall	Precision	F-score
Arabic	Mention Detection	81.10	12.29	21.35
	Candidate Pairs Generation	66.56	12.45	20.98
	Mention Clustering	43.00	47.87	45.30
Chinese	Mention Detection	84.14	22.42	35.41
	Candidate Pairs Generation	77.57	22.45	34.82
	Mention Clustering	54.40	68.19	60.52
English	Mention Detection	90.22	23.99	37.90
	Candidate Pairs Generation	84.54	23.87	37.23
	Mention Clustering	64.88	74.74	69.46

generated parse trees. This type of error is more frequent in the Arabic and Chinese corpora.

Although the candidate pair generation step does not significantly impact the precision, this step is able to discard a large portion of the candidate arcs, as depicted in Section 8.3.

In our system, the mention clustering subtask is almost completely solved by the coreference tree predictor. Now, let us examine the errors generated by this predictor. For this purpose, we compare each predicted document tree $\hat{\boldsymbol{h}}$ with the corresponding constrained document tree $\tilde{\boldsymbol{h}}$. Recall that the latter is the best scoring tree that uses only intracluster arcs and some additional artificial root arcs. For each incorrectly predicted arc $(i, j) \in \hat{\boldsymbol{h}}$ that links non-coreferent mentions i and j, let (i^*, j) be its corresponding correct arc in $\tilde{\boldsymbol{h}}$. For all documents in the development set, we compute the frequency of the head POS tags of mentions j, i, and i^*. Observe that a constrained document tree is computed on the graph generated in the candidate pair generation step, which already includes recall errors. Thus, the frequencies reported here are computed over the remaining errors.

In Table 17, we present the top 12 errors for each language, where NNP and NR stand for singular proper noun, NN for singular noun, NNS for plural noun, NT for temporal noun, PRP for personal pronoun, PRP$ for possessive pronoun, PN for pronoun, and *Root* for the artificial root node, which means that the corresponding node is the root of its coreference subtree.

A highly frequent error of our predictor is to connect a singleton mention to a coreference tree. This type of error corresponds to 45% of all errors for the Arabic language, 37.7% for Chinese, and 31.9% for English. The pleonastic *it* is a particularly problematic singleton in the English language. Indeed, for English, such cases roughly correspond to 14.2% of all singleton errors. In Table 18, we present the most frequent head POS tags for these singleton errors in each language development set. For Arabic and English, the order among these POS tags is very similar. The most frequent errors are pronouns, followed by proper nouns. However, in the English language, the proportion of pronouns is much higher than the proportion of proper nouns, most likely due

Table 17
Most frequent errors whenever an incorrect arc (i, j) is predicted instead of the correct arc (i^*, j).

Arabic				Chinese				English			
j	i	i^*	%	j	i	i^*	%	j	i	i^*	%
NNP	NNP	*Root*	15.1	NN	*Root*	NN	21.4	PRP	PRP	*Root*	9.2
NN	*Root*	NN	9.3	NN	NN	*Root*	20.9	NNP	NNP	*Root*	9.2
PRP	NN	*Root*	8.0	PN	PN	*Root*	9.3	NN	*Root*	NN	8.3
NN	NN	*Root*	5.9	NR	NR	*Root*	8.0	NN	NN	*Root*	6.0
PRP$	NN	*Root*	4.4	NN	NN	NN	4.7	NNS	*Root*	NNS	3.9
PRP	NNP	*Root*	3.4	PN	PN	PN	3.4	PRP	PRP	PRP	3.2
PRP	PRP	*Root*	2.9	PN	NN	*Root*	3.4	NNP	*Root*	NNP	3.2
NNP	*Root*	NNP	2.9	PN	*Root*	PN	3.0	PRP	NN	*Root*	3.0
PRP	PRP$	*Root*	2.7	NT	NT	*Root*	2.2	PRP	*Root*	PRP	2.7
NNP	NNP	NNP	2.5	NR	NR	NR	1.9	NNP	NNP	NNP	2.5
NN	*Root*	NNP	2.2	NN	*Root*	NR	1.8	NN	*Root*	NNP	2.5
PRP$	NN	NN	2.0	NR	*Root*	NR	1.8	PRP	NNP	*Root*	2.4
	Others		38.7		Others		18.1		Others		43.8

Table 18
Most frequent singleton errors per head POS tag and language.

Arabic		Chinese		English	
POS	**%**	**POS**	**%**	**POS**	**%**
PRP	35.6	NN	51.1	PRP	43.2
NNP	30.5	PN	25.2	NNP	25.5
PRP$	19.6	NR	19.1	NN	17.3
NN	12.1	NT	4.7	PRP$	8.8
NNS	2.2			*Others*	5.1

to the pleonastic *it* cases. Surprisingly, in the Chinese language, most singleton errors are nouns.

9. Conclusion

In this article, we describe a new machine learning system for multilingual unrestricted coreference resolution, based on two key modeling techniques: latent coreference trees and entropy-guided feature induction. We use the large margin structured perceptron as training algorithm. According to our experiments, latent trees are powerful enough to model the complexity of coreference structures in a document while rendering the learning problem computationally feasible. Our empirical findings also show that EFI enables our system to learn effective nonlinear classifiers while using a linear training algorithm.

We evaluate our system on the CoNLL-2012 Shared Task data sets, which comprise three languages: Arabic, Chinese, and English. To cope with this multilingual task, our system needs only minor adaptations due to certain language-dependent aspects of the used data sets. As far as we know, our system presents the best known performance on the three languages.

We also provide detailed experimental results that highlight the contribution of individual parts of our system, providing important insights to researchers interested in coreference resolution. These results also highlight interesting aspects of our approach that can be explored for further improvements.

One limitation of the proposed modeling approach is the arc-based features. Such local information is clearly not sufficient to model all dependencies involved in coreference resolution. It is necessary to consider more complex contextual features. Hence, in future work, we plan to include higher-order features and cluster-sensitive features. Higher-order tree-based features have been successfully applied to dependency parsers (McDonald and Pereira 2006; Koo et al. 2010) and can be used to extend our coreference system. Cluster-sensitive features can further extend our modeling by considering features that are more strongly adherent to the coreference task. However, as far as we know, there is as yet no structure learning system that considers such features. Thus, we must design new prediction algorithms to cope with this complex context.

Acknowledgments

This work has been partially funded by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro, and Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico through grants 557.128/2009-9, E-26/170028/2008, and 0011-00147.01.00/09, respectively. The first author was also

supported by a CNPq doctoral fellowship and a doctoral internship from Coordenação de Aperfeiçoamento de Pessoal de Nível Superior.

References

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

Bansal, Mohit and Dan Klein. 2012. Coreference semantics from web Features. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 389–398, Jeju Island.

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

Bergsma, Shane and Dekang Lin. 2006. Bootstrapping path-based pronoun resolution. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 33–40, Sydney.

Björkelund, Anders and Richárd Farkas. 2012. Data-driven multilingual coreference resolution using resolver stacking. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning: Shared Task*, pages 49–55, Jeju Island.

Björkelund, Anders and Pierre Nugues. 2011. Exploring lexicalized features for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 45–50, Portland, OR.

Cai, Jie, Eva Mujdricza-Maydt, and Michael Strube. 2011. Unrestricted coreference resolution via global hypergraph partitioning. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 56–60, Portland, OR.

Cai, Jie and Michael Strube. 2010. End-to-end coreference resolution via hypergraph partitioning. In *Proceedings of the International Conference on Computational Linguistics*, pages 143–151, Stroudsburg, PA.

Chang, Kai-Wei, Rajhans Samdani, Alla Rozovskaya, Nick Rizzolo, Mark Sammons, and Dan Roth. 2011. Inference protocols for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 40–44, Portland, OR.

Chen, Chen and Vincent Ng. 2012. Combining the best of two worlds: A hybrid approach to multilingual coreference resolution. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning: Shared Task*, pages 56–63, Jeju Island.

Chu, Y. J. and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1,396–1,400.

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

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

della Pietra, Stephen, Vincent della Pietra, and John Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

Denis, Pascal and Jason Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 236–243, Rochester, NY.

Denis, Pascal and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 660–669, Honolulu, HI.

Doddington, G., A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. 2004. The automatic content extraction (ACE) program—Tasks, data, and evaluation. In *Proceedings of the International Conference on LREC*, pages 837–840, Lisbon.

dos Santos, Cícero Nogueira and Ruy L. Milidiú. 2009. Entropy guided transformation learning. In A.-E. Hassanien et al., editors, *Foundations of Computational Intelligence (1)*. Springer, pages 159–184.

dos Santos, Cícero Nogueira and Davi L. Carvalho. 2011. Rule and tree ensembles for unrestricted coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 51–55, Portland, OR.

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

Fernandes, Eraldo R., Cícero Nogueira dos Santos, and Ruy L. Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning: Shared Task*, pages 41–48, Jeju Island.

Fernandes, Eraldo R. and Ruy L. Milidiú. 2012. Entropy-guided feature generation for structured learning of Portuguese dependency parsing. In H. Caseli et al., editors, *Proceedings of the Conference on Computational Processing of the Portuguese Language*, volume 7243 of *Lecture Notes in Computer Science*. Springer, Berlin/Heidelberg, pages 146–156.

Fernandes, Eraldo R. and Ulf Brefeld. 2011. Learning from partially annotated sequences. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6911 of *Lecture Notes in Computer Science*. Springer, Berlin/Heidelberg, pages 407–422.

Finkel, Jenny Rose and Christopher D. Manning. 2008. Enforcing transitivity in coreference resolution. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers*, pages 45–48, Columbus, OH.

Finley, Thomas and Thorsten Joachims. 2005. Supervised clustering with support vector machines. In *Proceedings of the International Conference on Machine Learning*, pages 217–224, New York, NY.

Haghighi, Aria and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1,152–1,161, Singapore.

Haghighi, Aria and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 385–393, Los Angeles, CA.

Klenner, Manfred. 2007. Enforcing consistency on coreference sets. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pages 323–328, Borovets.

Koo, Terry, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1,288–1,298, Cambridge, MA.

Lee, Heeyoung, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

Lee, Heeyoung, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34, Portland, OR.

Luo, Xiaoqiang. 2005. On coreference resolution performance metrics. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 25–32, Vancouver.

Luo, Xiaoqiang, 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 Annual Meeting of the Association for Computational Linguistics*, ACL'04, pages 135–142, Stroudsburg, PA.

Martschat, Sebastian, Jie Cai, Samuel Broscheit, Éva Mújdricza-Maydt, and Michael Strube. 2012. A multigraph model for coreference resolution. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning: Shared Task*, pages 100–106, Jeju Island.

McCallum, Andrew. 2003. Efficiently inducing features of conditional random fields. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, UAI'03, pages 403–410, San Francisco, CA.

McCallum, Andrew and Ben Wellner. 2005. Conditional models of identity uncertainty with application to noun coreference. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Proceedings of the Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, pages 905–912.

McCarthy, Joseph F. and Wendy G. Lehnert. 1995. Using decision trees for coreference resolution. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1,050–1,055, Montreal.

McDonald, Ryan, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 91–98, Ann Arbor, MI.

McDonald, Ryan and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11st Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, Trento.

Milidiú, Ruy Luiz, Cícero Nogueira dos Santos, and Julio C. Duarte. 2008. Phrase chunking using entropy guided transformation learning. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 647–655, Columbus, OH.

Ng, Vincent. 2009. Graph-cut-based anaphoricity determination for coreference resolution. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 575–583,Boulder, CO.

Ng, Vincent. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1,396–1,411, Uppsala.

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

Ponzetto, Simone Paolo and Michael Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 192–199, New York, NY.

Poon, Hoifung and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov logic. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 650–659, Honolulu, HI.

Pradhan, Sameer, 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*, CoNLL'12, pages 1–40, Jeju Island.

Pradhan, Sameer, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. CoNLL-2011 Shared Task: Modeling unrestricted coreference in OntoNotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27, Portland.

Quinlan, J. Ross. 1992. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, 1st edition.

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

Rosenblatt, Frank. 1957. The Perceptron—A perceiving and recognizing automaton. Technical report 85-460-1, Cornell Aeronautical Laboratory.

Sapena, Emili, Lluís Padró, and Jordi Turmo. 2010. Relaxcor: A global relaxation labeling approach to coreference resolution. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 88–91, Los Angeles, CA.

Sapena, Emili, Lluís Padró, and Jordi Turmo. 2011. Relaxcor participation in CoNLL Shared Task on coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 35–39, Portland, OR.

Sapena, Emili, Lluís Padró, and Jordi Turmo. 2013. A constraint-based hypergraph partitioning approach to coreference resolution. *Computational Linguistics*, 39(4):847–884.

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

Stoyanov, Veselin, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. 2010. Coreference resolution with reconcile. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Short Papers*, pages 156–161, Uppsala.

Stoyanov, Veselin, 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 Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 656–664, Suntec.

Su, Jiang and Harry Zhang. 2006. A fast decision tree learning algorithm. In *Proceedings of the 21st National Conference on Artificial intelligence*, pages 500–505, Boston, MA.

Sun, Xu, Takuya Matsuzaki, Daisuke Okanohara, and Jun'ichi Tsujii. 2009. Latent variable perceptron algorithm for structured classification. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 1,236–1,242, Pasadena, CA.

Sundheim, Beth and Ralph Grishman. 1995. Appendix D: Coreference task definition (v2.3). In *Proceedings of the 6th Conference on Message Understanding*, pages 333–344, Columbia, MD.

Tsochantaridis, I., Thorsten Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

Uryupina, Olga, Alessandro Moschitti, and Massimo Poesio. 2012. BART goes multilingual: The UniTN/Essex submission to the CoNLL-2012 Shared Task. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning: Shared Task*, pages 122–128, Jeju Island.

Uryupina, Olga, Sriparna Saha, Asif Ekbal, and Massimo Poesio. 2011. Multi-metric optimization for coreference: The UniTN/IITP/Essex submission to the 2011 CoNLL Shared Task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 61–65, Portland, OR.

Vilain, Marc, 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, Columbia, MD.

Weischedel, Ralph, Eduard Hovy, Mitchell Marcus, Martha Palmer, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. 2011. OntoNotes: A large training corpus for enhanced processing. In Joseph Olive, Caitlin Christianson, and John McCary, editors, *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*. Springer.

Yang, Xiaofeng, 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 the Annual Meeting of the Association for Computational Linguistics*, pages 843–851, Columbus, OH.

Yang, Xiaofeng, Jian Su, and Chew Lim Tan. 2008. A twin-candidate model for learning-based anaphora resolution. *Computational Linguistics*, 34(3):327–356.

Yang, Xiaofeng, Guodong Zhou, Jian Su, and Chew Lim Tan. 2003. Coreference resolution using competition learning approach. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 176–183, Sapporo.

Yu, Chun-Nam John and Thorsten Joachims. 2009. Learning structural SVMS with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML'09, pages 1,169–1,176, New York, NY.

Yuan, Bo, Qingcai Chen, Yang Xiang, Xiaolong Wang, Liping Ge, Zengjian Liu, Meng Liao, and Xianbo Si. 2012. A mixed deterministic model for coreference resolution. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning: Shared Task*, pages 76–82, Jeju Island.