

Latent Structure Perceptron with Feature Induction for Unrestricted Coreference Resolution

Eraldo Rezende Fernandes
Departamento de Informática
PUC-Rio
Rio de Janeiro, Brazil
efernandes@inf.puc-rio.br

Cícero Nogueira dos Santos
Brazilian Research Lab
IBM Research
Rio de Janeiro, Brazil
cicerons@br.ibm.com

Ruy Luiz Milidiú
Departamento de Informática
PUC-Rio
Rio de Janeiro, Brazil
milidiu@inf.puc-rio.br

Abstract

We describe a machine learning system based on large margin structure perceptron for unrestricted coreference resolution that introduces two key modeling techniques: latent coreference trees and entropy guided feature induction. The proposed latent tree modeling turns the learning problem computationally feasible. Additionally, using an automatic feature induction method, we are able to efficiently build nonlinear models and, hence, achieve high performances with a linear learning algorithm. Our system is evaluated on the CoNLL-2012 Shared Task *closed* track, which comprises three languages: Arabic, Chinese and English. We apply the same system to all languages, except for minor adaptations on some language dependent features, like static lists of pronouns. Our system achieves an official score of 58.69, the best one among all the competitors.

1 Introduction

The CoNLL-2012 Shared Task (Pradhan et al., 2012) is dedicated to the modeling of coreference resolution for multiple languages. The participants are provided with corpora for three languages: Arabic, Chinese and English. These corpora are provided by the OntoNotes project and, besides accurate anaphoric coreference information, contain various annotation layers such as part-of-speech (POS) tagging, syntax parsing, named entities (NE) and semantic role labeling (SRL). The shared task consists in the automatic identification of coreferring men-

tions of entities and events, given predicted information on other OntoNotes layers.

We propose a machine learning system for coreference resolution that is based on the large margin structure perceptron algorithm (Collins, 2002; Fernandes and Milidiú, 2012). Our system learns a predictor that takes as input a set of candidate mentions in a document and directly outputs the clusters of coreferring mentions. This predictor comprises an optimization problem whose objective is a function of the clustering features. To embed classic cluster metrics in this objective function is practically infeasible since most of such metrics lead to NP-hard optimization problems. Thus, we introduce *coreference trees* in order to represent a cluster by a directed tree over its mentions. In that way, the prediction problem optimizes over trees instead of clusters, which makes our approach computationally feasible. Since coreference trees are not given in the training data, we assume that these structures are *latent* and use the latent structure perceptron (Fernandes and Brefeld, 2011; Yu and Joachims, 2009) as the learning algorithm.

To provide high predicting power features to our model, we use *entropy guided feature induction* (Fernandes and Milidiú, 2012). By using this technique, we automatically generate several feature templates that capture coreference specific local context knowledge. Furthermore, this feature induction technique extends the structure perceptron framework by providing an efficient general method to build strong nonlinear classifiers.

Our system is evaluated on the CoNLL-2012 Shared Task *closed* track and achieves the scores

54.22, 58.49 and 63.37 on Arabic, Chinese and English test sets, respectively. The official score – the mean over the three languages – is 58.69, which is the best score achieved in the shared task.

The remainder of this paper is organized as follows. In Section 2, we present our machine learning modeling for the unrestricted coreference resolution task. In Section 3, we present the corpus preprocessing steps. The experimental findings are depicted in Section 4 and, in Section 5, we present our final remarks.

2 Task Modeling

Coreference resolution consists in identifying mention clusters in a document. We split this task into two subtasks: mention detection and mention clustering. For the first subtask, we apply the strategy proposed in (dos Santos and Carvalho, 2011). The second subtask requires a complex output. Hence, we use a structure learning approach that has been successfully applied to many similar structure finding NLP tasks (Collins, 2002; Tsochantaridis et al., 2005; McDonald et al., 2006; Fernandes and Brefeld, 2011; Fernandes and Milidiú, 2012).

2.1 Mention Detection

For each text document, we generate a list of candidate mentions using the strategy of (dos Santos and Carvalho, 2011). The basic idea is to use all noun phrases, and, additionally, pronouns and named entities, even if they are inside larger noun phrases. We do not include verbs as mentions.

2.2 Mention Clustering

In the mention clustering subtask, a *training* instance (\mathbf{x}, \mathbf{y}) consists of a set of mentions \mathbf{x} from a document and the correct coreferring clusters \mathbf{y} . The structure perceptron algorithm learns a predictor from a given training set $\mathcal{D} = \{(\mathbf{x}, \mathbf{y})\}$ of correct input-output pairs. More specifically, it learns the weight vector \mathbf{w} of the parameterized predictor given by

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

where $\mathcal{Y}(\mathbf{x})$ is the set of clusterings over mentions \mathbf{x} and s is a \mathbf{w} -parameterized scoring function over clusterings.

We use the *large margin* structure perceptron (Fernandes and Milidiú, 2012) that, during training, embeds a loss function in the prediction problem. Hence, it uses a loss-augmented predictor given by

$$F^\ell(\mathbf{x}) = \arg \max_{\mathbf{y}' \in \mathcal{Y}(\mathbf{x})} s(\mathbf{y}'; \mathbf{w}) + \ell(\mathbf{y}, \mathbf{y}'),$$

where ℓ is a non-negative loss function that measures how a candidate clustering \mathbf{y}' differs from the ground truth \mathbf{y} . The training algorithm makes intense use of the predictor, hence the prediction problem must be efficiently solved. Letting s be a classic clustering metric is infeasible, since most of such metrics lead to NP-hard optimization problems.

2.2.1 Coreference Trees

In order to reduce the complexity of the prediction problem, we introduce *coreference trees* to represent clusters of coreferring mentions. A coreference tree is a directed tree whose nodes are the coreferring mentions and arcs represent *some* coreference relation between mentions. In Figure 1, we present a document with seven highlighted mentions comprising two clusters. One plausible coreference tree for the cluster $\{a_1, a_2, a_3, a_4\}$ is presented in Figure 2.

North Korea_{a1} opened its_{a2} doors to the U.S. today, welcoming Secretary of State Madeleine Albright_{b1}. She_{b2} says her_{b3} visit is a good start. The U.S. remains concerned about North Korea's_{a3} missile development program and its_{a4} exports of missiles to Iran.

Figure 1: Exemplary document with seven highlighted mentions comprising two clusters: $\{a_1, a_2, a_3, a_4\}$ and $\{b_1, b_2, b_3\}$. The letter in the mention subscript indicates its cluster and the number uniquely identifies the mention within the cluster.

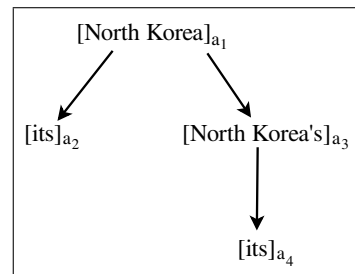


Figure 2: Coreference tree for the cluster a in Figure 1.

We are not concerned about the semantics underlying coreference trees, since they are just auxiliary

structures for the clustering task. However, we argue that this concept is linguistically plausible, since there is a dependency relation between coreferring mentions. Observing the aforementioned example, one may agree that mention a_3 (North Korea's) is indeed more likely to be associated with mention a_1 (North Korea) than with mention a_2 (its), even considering that a_2 is closer than a_1 in the text.

For a given document, we have a forest of coreference trees, one tree for each coreferring cluster. However, 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 text in Figure 1.

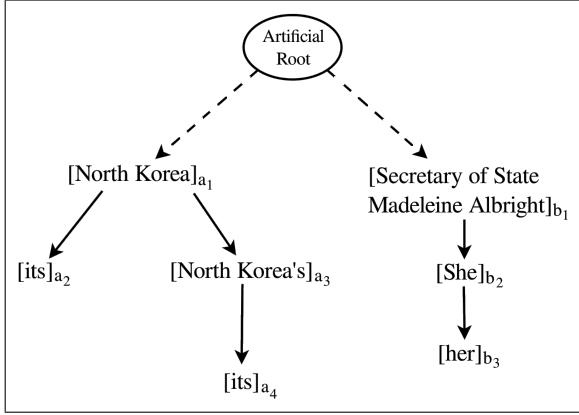


Figure 3: Document tree with two coreference trees for the text in Figure 1. Dashed lines indicate artificial arcs.

2.2.2 Latent Structure Learning

Coreference trees are not given in the training data. Thus, we assume that these structures are *latent* and make use of the latent structure perceptron (Fernandes and Brefeld, 2011; Yu and Joachims, 2009) to train our models. We decompose the original predictor into two predictors, that is

$$F(x) \equiv F_y(F_h(x)),$$

where the *latent predictor* $F_h(x)$ is defined as $\arg \max_{h \in \mathcal{H}(x)} \langle \mathbf{w}, \Phi(x, h) \rangle$, $\mathcal{H}(x)$ is the set of feasible document trees for x and $\Phi(x, h)$ is the joint feature vector representation of mentions x and document tree h . Hence, the latent predictor finds a maximum scoring rooted tree over the given mentions x , where a tree score is given by a linear function over its features. $F_y(h)$ is a straightforward

procedure that creates a cluster for each subtree connected to the artificial root node in the document tree h .

In Figure 4, we depict the proposed latent structure perceptron algorithm for the mention clustering task. Like its univariate counterpart (Rosenblatt,

```

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

```

Figure 4: Latent structure perceptron algorithm.

1957), the structure perceptron is an online algorithm that iterates through the training set. For each training instance, it performs two major steps: (i) a prediction for the given input using the current model; and (ii) a model update based on the difference between the predicted and the ground truth outputs. The latent structure perceptron performs an additional step to predict the latent ground truth $\tilde{\mathbf{h}}$ using a specialization of the latent predictor and the current model. This algorithm learns to predict document trees that help to solve the clustering task. Thereafter, for an unseen document x , the predictor $F_h(x)$ and the learned model \mathbf{w} are employed to produce a predicted document tree \mathbf{h} which, in turn, is fed to $F_y(\mathbf{h})$ to give the predicted clusters.

Golden coreference trees are not available. However, during training, for a given input x , we have the golden clustering y . Thus, we predict the *constrained document tree* $\tilde{\mathbf{h}}$ for the training instance (x, y) using a specialization of the latent predictor – the *constrained latent predictor* – that makes use of y . The constrained predictor finds the maximum scoring document tree among all rooted trees of x that follow the correct clustering y , that is, rooted trees that only include arcs between mentions that are coreferent according to y , plus one arc from the artificial node to each cluster. In that way, the constrained predictor optimizes over a subset $\mathcal{H}(x, y)$ contained in $\mathcal{H}(x)$ and, moreover, it guarantees that

$F_y(\tilde{h}) = y$, for any w . The constrained tree is used as the ground truth on each iteration. Therefore, the model update is determined by the difference between the constrained document tree and the document tree predicted by the ordinary predictor.

The loss function measures the impurity in the predicted document tree. In our modeling, we use a simple loss function that just counts how many predicted edges are not present in the constrained document tree. For the arcs from the artificial root node, we use a different loss value. We set that through the parameter r , which we call the *root loss value*.

We decompose the joint feature vector $\Phi(x, h)$ along tree edges, that is, pairs of candidate corefering mentions. This approach is similar to previous structure learning modelings for dependency parsing (McDonald et al., 2005; Fernandes and Milidiú, 2012). Thus, the prediction problem reduces to a maximum branching problem, which is efficiently solved by the *Chu-Liu-Edmonds algorithm* (Chu and Liu, 1965; Edmonds, 1967). We also use the *averaged* structure perceptron as suggested by (Collins, 2002), since it provides a more robust model.

3 Data Preparation

It is necessary to perform some corpus processing steps in order to prepare training and test data. In this section, we detail the methodology we use to generate coreference arcs and the features that describe them.

3.1 Coreference Arcs Generation

The input for the prediction problem is a graph whose nodes are the mentions in a document. Ideally, we could consider the complete graph for each document, thus every mention pair would be an option for building the document tree. However, since the total number of mentions is huge and a big portion of arcs can be easily identified as incorrect, we filter the arcs and, thus, include only candidate mention pairs that are more likely to be coreferent.

We filter arcs by simply adapting the sieves method proposed in (Lee et al., 2011). However, in our filtering strategy, precision is not a concern and the application order of filters is not important. The objective here is to build a small set of candidate arcs that shows good recall.

Given a mention pair (m_i, m_j) , where m_i appears before m_j in the text, we create a directed arc from m_i to m_j if at least one of the following conditions holds: (1) the number of mentions between m_i and m_j is not greater than a given parameter; (2) m_j is an alias of m_i ; (3) there is a match of both mentions strings up to their head words; (4) the head word of m_i matches the head word of m_j ; (5) test shallow discourse attributes match for both mentions; (6) m_j is a pronoun and m_i has the same gender, number, speaker and animacy of m_j ; (7) m_j is a pronoun and m_i is a compatible pronoun or proper name.

Sieves 2 to 7 are obtained from (Lee et al., 2011). We only introduce sieve 1 to lift recall without using other strongly language-dependent sieves.

3.2 Basic Features

We use a set of 70 basic features to describe each pair of mentions (m_i, m_j) . The feature set includes lexical, syntactic, semantic, and positional information. Our feature set is very similar to the one used by (dos Santos and Carvalho, 2011). However, here we do not use the semantic features derived from WordNet. In the following, we briefly describe some of these basic features.

Lexical: *head word* of $m_{i/j}$; *String matching* of (head word of) m_i and m_j (y/n); *Both are pronouns* and their strings match (y/n); *Previous/Next two words* of $m_{i/j}$; *Length* of $m_{i/j}$; *Edit distance* of head words; $m_{i/j}$ is a definitive NP (y/n); $m_{i/j}$ is a demonstrative NP (y/n); *Both are proper names* and their strings match (y/n).

Syntactic: *POS tag* of the $m_{i/j}$ head word; *Previous/Next two POS tags* of $m_{i/j}$; m_i and m_j are *both pronouns / proper names* (y/n); *Previous/Next predicate* of $m_{i/j}$; *Compatible pronouns*, which checks whether two pronouns agree in number, gender and person (y/n); *NP embedding level*; *Number of embedded NPs* in $m_{i/j}$.

Semantic: the result of a *baseline system*; *sense* of the $m_{i/j}$ head word; *Named entity type* of $m_{i/j}$; m_i and m_j have the *same named entity*; *Semantic role* of $m_{i/j}$ for the prev/next predicate; *Concatenation of semantic roles* of m_i and m_j for the same predicate (if they are in the same sentence); *Same speaker* (y/n); m_j is an *alias* of m_i (y/n).

Distance and Position: Distance between m_i and m_j in sentences; Distance in number of mentions;

Distance in number of person names (applies only for the cases where m_i and m_j are both pronouns or one of them is a person name); One mention is in apposition to the other (y/n).

3.3 Language Specifics

Our system can be easily adapted to different languages. In our experiments, only small changes are needed in order to train and apply the system to three different languages. The adaptations are due to: lack of input features for some languages; different POS tagsets are used in the corpora; and creation of static list of language specific pronouns.

Some input features, that are available for the English corpus, are not available in Arabic and Chinese corpora. Namely, the Arabic corpus does not contain NE, SRL and speaker features. Therefore, for this language we do not derive basic features that make use of these input features. For Chinese, we do not use features derived from NE data, since this data is not provided. Additionally, the Chinese corpus uses a different POS tagset. Hence, some few mappings are needed during the basic feature derivation stage.

The lack of input features for Arabic and Chinese also impact the sieve-based arcs generation. For Chinese, we do not use sieve 6, and, for Arabic, we only use sieves 1, 3, 4 and 7. Sieve 7 is not used for the English corpus, since it is a specialization of sieve 6. The first sieve parameter is 4 for Arabic and Chinese, and 8 for English.

In the arcs generation and basic feature derivation steps, our system makes use of static lists of language specific pronouns. In our experiments, we use the POS tagging information and the golden coreference chains to automatically extract these pronoun lists from training corpora.

3.4 Entropy Guided Feature Induction

In order to improve the predictive power of our system, we add complex features that are combinations of the basic features described in the previous section. We use feature templates to generate such complex features. However, we automatically generate templates using the entropy guided feature induction approach (Fernandes and Milidiú, 2012; Milidiú et al., 2008). These automatically generated templates capture complex contextual information and are difficult to be handcrafted by humans. Furthermore,

this feature induction mechanism extends the structure perceptron framework by providing an efficient general method to build strong nonlinear predictors.

We experiment with different template sets for each language. The main difference between these sets is basically the training data used to induce them. We obtain better results when merging different template sets. For the English language, it is better to use a template set of 196 templates, which merges two different sets: (a) a set induced using training data that contains mention pairs produced by filters 2 to 6; and (b) another set induced using training data that contains mention pairs produced by all filters. For Chinese and Arabic, it is better to use template sets induced specifically for these languages merged with the template set (a) generated for the English language. The final set for the Chinese language has 197 templates, while the final set for Arabic has 223.

4 Empirical Results

We train our system on the corpora provided in the CoNLL-2012 Shared Task. There are corpora available on three languages: Arabic, Chinese and English. For each language, results are reported using three metrics: MUC, B^3 and CEAF_e. We also report the mean of the F-scores on these three metrics, which gives a unique score for each language. Additionally, the official score on the CoNLL-2012 shared task is reported, that is the mean of the scores obtained on the three languages.

We report our system results on development and test sets. The development results are obtained with systems trained only on the training sets. However, test set results are obtained by training on a larger dataset – the one obtained by concatenating training and development sets. During training, we use the *gold standard* input features, which produce better performing models than using the provided automatic values. That is usually the case on NLP tasks, since golden values eliminate the additional noise introduced by automatic features. On the other hand, during evaluation, we use the automatic values provided in the CoNLL shared task corpora.

In Table 1, we present our system performances on the CoNLL-2012 development sets for the three languages. Given the size of the Arabic training cor-

Language	MUC			B ³			CEAF _e			Mean
	R	P	F ₁	R	P	F ₁	R	P	F ₁	
Arabic	43.00	47.87	45.30	61.41	70.38	65.59	49.42	44.19	46.66	52.52
Chinese	54.40	68.19	60.52	64.17	78.84	70.76	51.42	38.96	44.33	58.54
English	64.88	74.74	69.46	66.53	78.28	71.93	54.93	43.68	48.66	63.35
Official Score										58.14

Table 1: Results on the development sets.

Language	MUC			B ³			CEAF _e			Mean
	R	P	F ₁	R	P	F ₁	R	P	F ₁	
Arabic	34.18	58.85	43.25	50.61	82.13	62.63	57.37	33.75	42.49	49.45
Chinese	49.17	76.03	59.72	58.16	86.33	69.50	57.56	34.38	43.05	57.42
English	62.75	77.41	69.31	63.88	81.34	71.56	57.46	41.08	47.91	62.92
Official Score										56.59

Table 2: Results on the development sets *without* root loss value.

Language	MUC			B ³			CEAF _e			Mean
	R	P	F ₁	R	P	F ₁	R	P	F ₁	
Arabic	43.63	49.69	46.46	62.70	72.19	67.11	52.49	46.09	49.08	54.22
Chinese	52.69	70.58	60.34	62.99	80.57	70.70	53.75	37.88	44.44	58.49
English	65.83	75.91	70.51	65.79	77.69	71.24	55.00	43.17	48.37	63.37
Official Score										58.69

Table 3: Official results on the test sets.

Language	Parse / Mentions	MUC			B ³			CEAF _e			Mean
		R	P	F ₁	R	P	F ₁	R	P	F ₁	
Arabic	Auto / GB	45.18	47.39	46.26	64.56	69.44	66.91	49.73	47.39	48.53	53.90
	Auto / GM	57.25	76.48	65.48	60.27	79.81	68.68	72.61	46.00	56.32	63.49
	Golden / Auto	46.38	51.78	48.93	63.53	72.37	67.66	52.57	46.88	49.56	55.38
	Golden / GB	46.38	51.78	48.93	63.53	72.37	67.66	52.57	46.88	49.56	55.38
	Golden / GM	56.89	76.27	65.17	60.07	80.02	68.62	72.24	45.58	55.90	63.23
Chinese	Auto / GB	58.76	71.46	64.49	66.62	79.88	72.65	54.09	42.02	47.29	61.48
	Auto / GM	61.64	90.81	73.43	63.55	89.43	74.30	72.78	39.68	51.36	66.36
	Golden / Auto	59.35	74.49	66.07	66.31	81.43	73.10	55.97	41.50	47.66	62.28
	Golden / GB	59.35	74.49	66.07	66.31	81.43	73.10	55.97	41.50	47.66	62.28
	Golden / GM	61.70	91.45	73.69	63.57	89.76	74.43	72.84	39.49	51.21	66.44
English	Auto / GB	64.92	77.53	70.67	64.25	78.95	70.85	56.48	41.69	47.97	63.16
	Auto / GM	70.69	91.21	79.65	65.46	85.61	74.19	74.71	42.55	54.22	69.35
	Golden / Auto	67.73	77.25	72.18	66.42	78.01	71.75	56.16	44.51	49.66	64.53
	Golden / GB	65.65	78.26	71.40	64.36	79.09	70.97	57.36	42.23	48.65	63.67
	Golden / GM	71.18	91.24	79.97	65.81	85.51	74.38	74.93	43.09	54.72	69.69

Table 4: Supplementary results on the test sets alternating parse quality and mention candidates. Parse quality can be automatic or golden; and mention candidates can be automatically identified (Auto), golden mention boundaries (GB) or golden mentions (GM).

pus and the feature limitations for Arabic and Chinese, the performance variations among the three languages are no more than expected. One important parameter that we introduce in this work is the root loss value, a different loss function value on arcs from the artificial root node. The effect of this parameter is to diminish the creation of clusters, thus

stimulating bigger clusters and adjusting the balance between precision and recall. Using the development sets for tuning, we set the value of the root loss value parameter to 6, 2 and 1.5 for Arabic, Chinese and English, respectively. In Table 2, we present our system performances on the development sets when we set this parameter to 1 for all languages, that is

equivalent to not use this parameter at all. We can observe, by comparing these results with the ones in Table 1, that this parameter really causes a better balancing between precision and recall, and consequently increases the F_1 scores. Its effect is accentuated on Arabic and Chinese, since the unbalancing issue is worse on these languages.

The official results on the test sets are depicted in Table 3. For Chinese and English, these performances are virtually identical to the performances on the development sets. On the other hand, the official performance for the Arabic language is significantly higher than the development set performance. This difference is likely due to the fact that the Arabic training set is much smaller than the Chinese and English counterparts. Thus, by including the development set in the training of the final Arabic system, we significantly improve the official performance.

We report in Table 4 the *supplementary* results provided by the shared task organizers on the test sets. These additional experiments investigate two key aspects of any coreference resolution system: the parse feature and the mention candidates that are given to the clustering procedure. We alternate the parse feature between the official *automatic* parse and the *golden* parse from OntoNotes. Regarding mention candidates, we use three different strategies: automatic mentions (Auto, in Table 4), golden mention boundaries (GB) and golden mentions (GM). Automatic mentions are completely detected by our system, as described in Section 2.1. Golden mention boundaries comprise all noun phrases in the *golden* parse tree, even when the automatic parse is used as input feature. Golden mentions are all non-singleton mentions, i.e., all mentions that take part in some entity cluster. It is important to notice that golden mention information is much stronger than golden boundaries.

By observing Table 4, it is clear that the most beneficial information is golden mentions (compare the Auto/GM results in Table 4 with the results in Table 3). The mean F-score over all languages when using golden mentions is almost 8 points higher than the official score. These results are not surprising since to identify non-singleton mentions greatly reduces the final task complexity. Golden mention boundaries (Auto/GB) increase the mean F-score for Chinese by almost 3 points. Conversely, for the

other two languages, the results are decreased when this information is given. This is probably due to parameter tuning, since any additional information potentially changes the learning problem and, nevertheless, we use exactly the same three models – one per language – to produce all the results on Tables 3 and 4. One can observe, for instance, that the recall/precision balance greatly varies among the different configurations in these experiments. The golden parse feature (Golden/Auto) causes big improvements on the mean F-scores for all languages, specially for Chinese.

5 Conclusion

In this paper, we describe a machine learning system based on large margin latent structure perceptron for unrestricted coreference resolution. We introduce two modeling approaches that have direct impact on the final system performance: latent coreference trees and entropy guided feature induction.

According to our experiments, latent coreference trees are powerful enough to model the complexity of coreference structures in a document, while turning the learning problem computationally feasible. Our empirical findings also show that entropy guided feature induction enables learning of effective nonlinear classifiers.

Our system is evaluated on the CoNLL-2012 Shared Task *closed* track, which consists on modeling coreference resolution for three languages: Arabic, Chinese and English. In order to cope with this multi-language task, our system needs only minor adaptations on some language dependent features.

As future work, we plan to include second order features and cluster sensitive features.

Acknowledgments

This work was 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 by the Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

References

- Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.
- Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, pages 1–8.
- Cicero Nogueira dos Santos and Davi Lopes 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, Oregon, USA, June. Association for Computational Linguistics.
- J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.
- Eraldo R. Fernandes and Ulf Brefeld. 2011. Learning from partially annotated sequences. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, Athens, Greece.
- Eraldo R. Fernandes and Ruy L. Milidiú. 2012. Entropy-guided feature generation for structured learning of Portuguese dependency parsing. In *Proceedings of the Conference on Computational Processing of the Portuguese Language (PROPOR)*, volume 7243 of *Lecture Notes in Computer Science*, pages 146–156. Springer Berlin / Heidelberg.
- Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford’s multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL Shared Task 2011, pages 28–34, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL’05*, pages 91–98.
- Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 216–220.
- Ruy L. Milidiú, Cicero N. dos Santos, and Julio C. Duarte. 2008. Phrase chunking using entropy guided transformation learning. In *Proceedings of ACL2008*, Columbus, Ohio.
- Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning (CoNLL 2012)*, Jeju, Korea.
- Frank Rosenblatt. 1957. The Perceptron – a perceiving and recognizing automaton. Technical report, Cornell Aeronautical Laboratory. Report 85-460-1.
- I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.
- Chun-Nam Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *Proceedings of the International Conference on Machine Learning (ICML)*.