Don't understand a measure? Learn it: Structured Prediction for Coreference Resolution optimizing its measures

Iryna Haponchyk* and **Alessandro Moschitti**
*DISI, University of Trento, 38123 Povo (TN), Italy
Qatar Computing Research Institute, HBKU, 34110, Doha, Qatar
{gaponchik.irina,amoschitti}@gmail.com

Abstract

An assential aspect of structured prediction is the evaluation of an output structure against the gold standard. Especially in the loss-augmented setting, the need of finding the max-violating constraint has severely limited the expressivity of effective loss functions. In this paper, we trade off exact computation for enabling the use of more complex loss functions for coreference resolution (CR). Most noteworthily, we show that such functions can be (i) automatically learned also from controversial but commonly accepted CR measures, e.g., MELA, and (ii) successfully used in learning algorithms. The accurate model comparison on the standard CoNLL–2012 setting shows the benefit of more expressive loss for Arabic and English data.

1 Introduction

In recent years, interesting structured prediction methods have been developed for coreference resolution (CR), e.g., (Fernandes et al., 2014; Björkelund and Kuhn, 2014; Martschat and Strube, 2015). These models are supposed to output clusters but, to better control the exponential nature of the problem, the clusters are converted into tree structures. Although this simplifies the problem, optimal solutions are associated with an exponential set of trees, requiring to maximize over such a set. This originated latent models (Yu and Joachims, 2009) optimizing the so-called loss-augmented objective functions.

In this setting, loss functions need to be factorizable together with the feature representations for finding the max-violating constraints. The consequence is that only simple loss functions, basically just counting incorrect edges, were applied in previous work, giving up expressivity for simplicity. This is a critical limitation as domain experts consider more information than just counting edges.

In this paper, we study the use of more expressive loss functions in the structured prediction framework for CR, although some findings are clearly applicable to more general settings. We attempted to optimize the complicated official MELA measure[1] (Pradhan et al., 2012) of CR within the learning algorithm. Unfortunately, MELA is the average of measures, among which CEAF_e has an excessive computational complexity preventing its direct use. To solve this problem, we defined a model for learning MELA from data using a fast linear regressor, which can be then effectively used in structured prediction algorithms. We defined features to learn such a loss function, e.g., different link counts or aggregations such as Precision and Recall. Moreover, we designed methods for generating training data from which our regression loss algorithm (RL) can generalize well and accurately predict MELA values on unseen data.

Since RL is not factorizable[2] over a mention graph, we designed a latent structured perceptron (LSP) that can optimize non-factorizable loss functions on CR graphs. We tested LSP using RL and other traditional loss functions using the same setting of the CoNLL–2012 Shared Task, thus enabling an exact comparison with previous work. The results confirmed that RL can be effectively learned and used in LSP, although the improvement was smaller than expected, considering that our RL provides the algorithm with a more accurate feedback.

Thus, we analyzed the theory behind this pro-

[1] Received most consensus in the NLP community.
[2] We have not found yet a possible factorization.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1018–1028
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1094

cess by also contributing to the definition of the properties of loss optimality. These show that the available loss functions, e.g., by Fernandes et al.; Yu and Joachims, are enough for optimizing MELA on the training set, at least when the data is separable. Thus, in such conditions, we cannot expect a very large improvement from RL.

To confirm such a conjecture, we tested the models in a more difficult setting, in terms of separability. We used different feature sets of a smaller size and found out that in such conditions, RL requires less epochs for converging and produces better results than the other simpler loss functions. The accuracy of RL-based model, using 16 times less features, decreases by just 0.3 points, still improving the state of the art in structured prediction. Accordingly, in the Arabic setting, where the available features are less discriminative, our approach highly improves the standard LSP.

2 Related Work

There is a number of works attempting to directly optimize coreference metrics. The solution proposed by Zhao and Ng (2010) consists in finding an optimal weighting (by beam search) of training instances, which would maximize the target coreference metric. Their models, optimizing MUC and B^3, deliver a significant improvement on the MUC and ACE corpora. Uryupina et al. (2011) benefited from applying genetic algorithms for the selection of features and architecture configuration by multi-objective optimization of MUC and the two CEAF variants. Our approach is different in that the evaluation measure (its approximation) is injected directly into the learning algorithm. Clark and Manning (2016) optimize B^3 directly as well within a mention-ranking model. For the efficiency reasons, they omit optimization of CEAF, which we enable in this work.

$SVM^{cluster}$ – a structured output approach by Finley and Joachims (2005) – enables optimization to any clustering loss function (including non-decomposable ones). The authors experimentally show that optimizing particular loss functions results into a better classification accuracy in terms of the same functions. However, these are in general fast to compute, which is not the MELA case.

While Finley and Joachims are compelled to perform approximate inference to overcome the intractability of finding an optimal clustering, the latent variable structural approaches – SVM of Yu and Joachims (2009) and perceptron of Fernandes et al. (2014) – render exact inference possible by introducing auxiliary graph structures. The modeling of Fernandes et al. (also referred to as the *antecedent tree* approach) is exploited in the works of Björkelund and Kuhn (2014), Martschat and Strube (2015), and Lassalle and Denis (2015). Like us, the first couples such approach with approximate inference but for enabling the use of non-local features. The current state-of-the-art model of Wiseman et al. (2016) also employs a greedy inference procedure as it has global features from an RNN as a non-decomposable term in the inference objective.

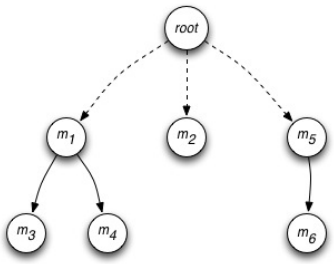

Figure 1: Latent tree used for structural learning

3 Structure Output Learning for CR

We consider online learning algorithms for linking structured input and output patterns. More formally, such algorithms find a linear mapping $f(\mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle$, where $f : X \times Y \rightarrow \mathbb{R}$, $\mathbf{w}$ is a linear model, $\Phi(\mathbf{x}, \mathbf{y})$ is a combined feature vector of input variables X and output variables Y. The predicted structure is derived with the $\operatorname{argmax}_{\mathbf{y} \in Y} f(\mathbf{x}, \mathbf{y})$. In the next sections, we show how to learn $\mathbf{w}$ for CR using structured perceptron. Additionally, we provide a characterization of effective loss functions for separable cases.

3.1 Modeling CR

In this framework, CR is essentially modeled as a clustering problem, where an input-output example is described by a tuple $(\mathbf{x}, \mathbf{y}, \mathbf{h})$, $\mathbf{x}$ is a set of entity mentions contained in a text document, $\mathbf{y}$ is set of the corresponding mention clusters, and $\mathbf{h}$ is a latent variable, i.e., an auxiliary structure that can represent the clusters of $\mathbf{y}$. For example, given the following text:

Although *(she)*$_{m_1}$ was supported by *(President Obama)*$_{m_2}$, *(Mrs. Clinton)*$_{m_3}$ missed *(her)*$_{m_4}$ *(chance)*$_{m_5}$, *(which)*$_{m_6}$ *looked* very good before counting votes.

the clusters of the entity mentions are represented by the latent tree in Figure 1, where its nodes are

Algorithm 1 Latent Structured Perceptron

1: Input: $X = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, $\mathbf{w}_0$, C, T
2: $\mathbf{w} \leftarrow \mathbf{w}_0$; $t \leftarrow 0$
3: **repeat**
4: **for** $i = 1, ..., n$ **do**
5: $\mathbf{h}_i^* \leftarrow \underset{\mathbf{h} \in H(\mathbf{x}_i, \mathbf{y}_i)}{\text{argmax}} \langle \mathbf{w}_t, \Phi(\mathbf{x}_i, \mathbf{h}) \rangle$
6: $\hat{\mathbf{h}}_i \leftarrow \underset{\mathbf{h} \in H(\mathbf{x}_i)}{\text{argmax}} \langle \mathbf{w}_t, \Phi(\mathbf{x}_i, \mathbf{h}) \rangle + C \times \Delta(\mathbf{y}_i, \mathbf{h}_i^*, \mathbf{h})$
7: **if** $\Delta(\mathbf{y}_i, \mathbf{h}^*{}_i, \hat{\mathbf{h}}_i) > 0$ **then**
8: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \Phi(\mathbf{x}_i, \mathbf{h}_i^*) - \Phi(\mathbf{x}_i, \hat{\mathbf{h}}_i)$
9: **end if**
10: **end for**
11: $t \leftarrow t + 1$
12: **until** $t < nT$
13: $\mathbf{w} \leftarrow \frac{1}{t} \sum_{i=1}^{t} \mathbf{w}_i$
return $\mathbf{w}$

mentions and the subtrees connected to the additional root node form distinct clusters. The tree $\mathbf{h}$ is called a latent variable as it is consistent with $\mathbf{y}$, i.e., it contains only links between mention nodes that corefer or fall into the same cluster according to $\mathbf{y}$. Clearly, an exponential set of trees, H, can be associated with one and the same clustering $\mathbf{y}$. Using only one tree to represent a clustering makes the search for optimal mention clusters tractable. In particular, structured prediction algorithms select $\mathbf{h}$ that maximizes the model learned at time t as shown in the next section.

3.2 Latent Structured Perceptron (LSP)

The LSP model proposed by Sun et al. (2009) and specialized for solving CR tasks by Fernandes et al. (2012) is described by Alg. 1.

Given a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, initial $\mathbf{w}_0$[3], a trade off parameter C, and the maximum number of epochs T, LSP iterates the following operations: Line 5 finds a latent tree $\mathbf{h}_i^*$ that maximizes $\langle \mathbf{w}_t, \Phi(\mathbf{x}_i, \mathbf{h}) \rangle$ for the current example $(\mathbf{x}_i, \mathbf{y}_i)$. It basically finds the max ground truth tree with respect to the current $\mathbf{w}_t$. Finding such max requires an exploration over the tree set $H(\mathbf{x}_i, \mathbf{y}_i)$, which only contains arcs between mentions that corefer according to the gold standard clustering $\mathbf{y}_i$. Line 6 seeks for the max-violating tree $\hat{\mathbf{h}}_i$ in $H(\mathbf{x}_i)$, which is the set of all candidate trees using any possible combination of arcs. Line 7 tests if the produced tree $\hat{\mathbf{h}}_i$ has some mistakes with respect to the gold clustering $\mathbf{y}_i$, using loss function $\Delta(\mathbf{y}_i, \mathbf{h}_i^*, \hat{\mathbf{h}}_i)$. Note that some models define a loss exploiting also the current best latent tree $\mathbf{h}_i^*$. If the test is verified, the model is updated with the vector $\Phi(\mathbf{x}_i, \mathbf{h}_i^*) - \Phi(\mathbf{x}_i, \hat{\mathbf{h}}_i)$.

[3]Either 0 or a random vector.

Fernandes et al. (2012) used exactly the directed trees we showed as latent structures and applied Edmonds' spanning tree algorithm (Edmonds, 1967) for finding the max. Their model achieved the best results in the CoNLL–2012 Shared Task, a challenge for CR systems (Pradhan et al., 2012). Their selected loss function also plays an important role as shown in the following.

3.3 Loss functions

When defining a loss, it is very important to preserve the factorization of the model components along the latent tree edges since this leads to efficient maximization algorithms (see Section 5).

Fernandes et al. uses a loss function that (i) compares a predicted tree $\hat{\mathbf{h}}$ against the gold tree $\mathbf{h}^*$ and (ii) factorizes over the edges in the way the model does. Its equation is:

$$\Delta_F(\mathbf{h}^*, \hat{\mathbf{h}}) = \sum_{i=1}^{M} \mathbb{1}_{\hat{\mathbf{h}}(i) \neq \mathbf{h}^*(i)} (1 + r \cdot \mathbb{1}_{\mathbf{h}^*(i)=0}), \quad (1)$$

where $\mathbf{h}^*(i)$ and $\hat{\mathbf{h}}(i)$ output the parent of the mention node i in the gold and predicted tree, respectively, whereas $\mathbb{1}_{\mathbf{h}^*(i) \neq \hat{\mathbf{h}}(i)}$ just checks if the parents are different, and if yes, penalty of 1 (or $1 + r$ if the gold parent is the root) is added.

Yu and Joachims's loss is based on undirected tree without a root and on the gold clustering $\mathbf{y}$. It is computed as:

$$\Delta_{YJ}(\mathbf{y}, \hat{\mathbf{h}}) = n(\mathbf{y}) - k(\mathbf{y}) + \sum_{\mathbf{e} \in \hat{\mathbf{h}}} l(\mathbf{y}, \mathbf{e}), \quad (2)$$

where $n(\mathbf{y})$ is the number of graph nodes, $k(\mathbf{y})$ is the number of clusters in $\mathbf{y}$, and $l(\mathbf{y}, \mathbf{e})$ assigns -1 to any edge $\mathbf{e}$ that connects nodes from the same cluster in $\mathbf{y}$, and r otherwise.

In our experiments, we adopt both loss functions, however, in contrast to Fernandes et al., we always measure Δ_F against the gold label $\mathbf{y}$ and not against the current $\mathbf{h}^*$, i.e., in the way it is done by Martschat and Strube (2015), who employ an equivalent LSP model in their work.

3.4 On optimality of simple loss functions

The above loss functions are rather simple and mainly based on counting the number of mistaken edges. Below, we show that such simple loss functions achieve training data separation (if it exists) of a general task measure reaching its max on their 0 mistakes. The latter is a desirable characteristic of many measures used in CR and NLP research.

Proposition 1 (Sufficient condition for optimality of loss functions for learning graphs)**.** *Let* $\Delta(\mathbf{y}, \mathbf{h}^*, \hat{\mathbf{h}}) \geq 0$ *be a simple, edge-factorizable loss function, which is also monotone in the number of edge errors, and let* $\mu(\mathbf{y}, \hat{\mathbf{h}})$ *be any graph-based measure maximized by no edge errors. Then, if the training set is linearly separable LSP optimizing* Δ *converges to the* μ *optimum.*

Proof. If the data is linearly separable the perceptron converges $\Rightarrow \Delta(\mathbf{y}_i, \mathbf{h^*}_i, \hat{\mathbf{h}}_i) = 0, \forall \mathbf{x}_i$. The loss is factorizable, i.e.,

$$\Delta(\mathbf{y}_i, \mathbf{h^*}_i, \hat{\mathbf{h}}_i) = \sum_{\mathbf{e} \in \hat{\mathbf{h}}_i} l(\mathbf{y}_i, \mathbf{h^*}_i, \mathbf{e}), \qquad (3)$$

where $l(\cdot)$ is an edge loss function. Thus, $\sum_{\mathbf{e} \in \hat{\mathbf{h}}_i} l(\mathbf{y}_i, \mathbf{h^*}_i, \mathbf{e}) = 0$. The latter equation and monotonicity imply $l(\mathbf{y}_i, \mathbf{h^*}_i, \mathbf{e}) = 0, \forall \mathbf{e} \in \hat{\mathbf{h}}_i$, i.e., there are no edge mistakes, otherwise by fixing such edges, we would have a smaller Δ, i.e., negative, contradicting the initial positiveness hypothesis. Thus, no edge mistake in any $\mathbf{x}_i$ implies that $\mu(\mathbf{y}, \hat{\mathbf{h}})$ is maximized on the training set. □

Corollary 1. $\Delta_F(\mathbf{h}^*, \hat{\mathbf{h}})$ and $\Delta_{YJ}(\mathbf{y}, \hat{\mathbf{h}})$ are both optimal loss functions for graphs.

Proof. Equations 1 and 2 show that both are 0 when applied to a clustering with no mistake on the edges. Additionally, for each edge mistake more, both loss functions increase, implying monotonicity. Thus, they satisfy all the assumptions of Proposition 1. □

The above characteristic suggests that Δ_F and Δ_{YJ} can optimize any measure that reasonably targets no mistakes as its best outcome. Clearly, this property does not guarantee loss functions to be suitable for a given task measure, e.g., the latter may have different max points and behave rather discontinuously. However, a common practice in NLP is to optimize the maximum of a measure, e.g., in case of Precision and Recall, or Accuracy, therefore, loss functions able to at least achieve such an optimum are preferable.

4 Automatically learning a loss function

How to measure a complex task such as CR has generated a long and controversial discussion in the research community. While such a debate is progressing, the most accepted and used measure is the so-called Mention, Entity, and Link Average (MELA) score. As it will be clear from the description below, MELA is not easily interpretable and not robust to the mention identification effect (Moosavi and Strube, 2016). Thus, loss functions showing the optimality property may not be enough to optimize it. Our proposal is to use a version of MELA transformed in a loss function optimized by an LSP algorithm with inexact inference. However, the computational complexity of the measure prevents to carry out an effective learning. Our solution is thus to learn MELA with a fast linear regressor, which also produces a continuos version of the measure.

4.1 Measures for CR

MELA is the unweighted average of MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998) and $CEAF_e$ (CEAF variant with entity-based similarity) (Luo, 2005; Cai and Strube, 2010) scores, having heterogeneous nature.

MUC is based on the number of correctly predicted links between mentions. The number of links required for obtaining the key entity set K is $\sum_{k_i \in K}(|k_i| - 1)$, where k_i are key entities in K (cardinality of each entity minus one). MUC recall computes what fraction of these were predicted, and the predicted were as many as $\sum_{k_i \in K}(|k_i| - |p(k_i)|) = \sum_{k_i \in K}(|k_i| - 1 - (|p(k_i)| - 1))$, where $p(k_i)$ is a partition of the key entity k_i formed by intersecting it with the corresponding response entities $r_j \in R$, s.t., $k_i \cap r_j \neq \emptyset$. This number equals to the number of the key links minus the number of missing links, required to unite the parts of the partition $p(k_i)$ to obtain k_i.

B^3 computes Precision and Recall individually for each mention. For mention m: $Recall_m = \frac{|k_i^m \cap r_j^m|}{|k_i^m|}$, where k_i^m and r_j^m, subscripted with m, denote, correspondingly, the key and response entities into which m falls. The over-document Recall is then an average of these taken with respect to the number of the key mentions. The MUC and B^3 Precision is computed by interchanging the roles of the key and response entities.

$CEAF_e$ computes similarity between key and system entities after finding an optimal alignment between them. Using $\psi(k_i, r_j) = \frac{2|k_i \cap r_j|}{|k_i| + |r_j|}$ as the entity similarity measure, it finds an optimal one-to-one map $g^* : K \rightarrow R$, which maps every key entity to a response entity, maximazing an overall similarity $\Psi(g) = \sum_{k_i \in K} \psi(k_i, g(k_i))$ of the example. This is solved as a bipartite matching problem by the Kuhn-Munkres algorithm. Then Preci-

Algorithm 2 Finding a Max-violating Spanning Tree

1: Input: training example $(\mathbf{x}, \mathbf{y})$; graph $G(\mathbf{x})$ with vertices V denoting mentions; set of the incoming candidate edges, $E(\mathbf{v})$, $\mathbf{v} \in V$; weight vector $\mathbf{w}$
2: $\mathbf{h}^* \leftarrow \emptyset$
3: **for** $\mathbf{v} \in V$ **do**
4: $\quad \mathbf{e}^* = \underset{\mathbf{e} \in E(\mathbf{v})}{\operatorname{argmax}} \langle \mathbf{w}, \mathbf{e} \rangle + C \times l(\mathbf{y}, \mathbf{e})$
5: $\quad \mathbf{h}^* = \mathbf{h}^* \cup \mathbf{e}^*$
6: **end for**
7: **return** max-violating tree $\mathbf{h}^*$
8: (clustering $\mathbf{y}^*$ is induced by the tree $\mathbf{h}^*$)

sion and Recall are $\frac{\Psi(g^*)}{\sum_{r_j \in R} \psi(r_j, r_j)}$ and $\frac{\Psi(g^*)}{\sum_{k_i \in K} \psi(k_i, k_i)}$, respectively.

MELA computation is rather expensive mostly because of CEAF$_e$. Its complexity is bounded by $\mathcal{O}(Ml^2 \log l)$ (Luo, 2005), where M and l are, correspondingly, the maximum and minimum number of entities in $\mathbf{y}$ and $\hat{\mathbf{y}}$. Computing CEAF$_e$ is especially slow for the candidate outputs $\hat{\mathbf{y}}$ with a low quality of prediction, i.e, when l is big, and the coherence with the gold $\mathbf{y}$ is scarse.

Finally, B^3 and CEAF$_e$ are strongly influenced by the mention identification effect (Moosavi and Strube, 2016). Thus, Δ_F and Δ_{YJ} may output identical values for different clusterings that can have a big gap in terms of MELA.

4.2 Features for learning measures

As computational reasons prevent to use MELA in LSP (see our inexact search algorithm in Section 5), we study methods for approximating it with a linear regressor. For this purpose, we define nine features, which count either exact or simplified versions of Precision, Recall and F1 of each of the three metric-components of MELA. Clearly, neither Δ_F nor Δ_{YJ} provide the same values.

Apart from the computational complexity, the difficulty of evaluating the quality of the predicted clustering $\hat{\mathbf{y}}$ during training is also due to the fact that CR is carried out on automatically detected mentions, while it needs to be compared against a gold standard clustering of a gold mention set. However, we can use simple information about automatic mentions and how they relate to gold mentions and gold clusters. In particular, we use four numbers: (i) correctly detected automatic mentions, (ii) links they have in the gold standard, (iii) gold mentions, and (iv) gold links. The last one enables the precise computation of Precision, Recall and F1-measure values of MUC; the required partitions $p(k_i)$ of key entities are also available at training time as they contain only automatic mentions. These are the first three features that we design. Likewise for B^3, the feature values can be derived using (ii) and (iii).

For computing CEAF$_e$ heuristics, we do not perform cluster alignment to find an optimal $\Psi(g^*)$. Instead of $\Psi(g^*)$, which can be rewritten as $\sum_{m \in K \cap R} \frac{2}{|k_i^m| + |g^*(k_i^m)|}$ if summing up over the mentions not the entities, we simply use $\tilde{\Psi} = \sum_{m \in K \cap R} \frac{2}{|k_i^m| + |r_j^m|}$, pretending that for each m its key k_i^m and response r_j^m entities are aligned. $\sum_{r_j \in R} \psi(r_j, r_j)$ and $\sum_{k_i \in K} \psi(k_i, k_i)$ in the denominators of the Precision and Recall are the number of predicted and gold clusters, correspondingly. The imprecision of the CEAF$_e$ related features is expected to be leveraged when put together with the exact B^3 and MUC values into the regression learning using the exact MELA values (implicitly exact CEAF$_e$ values as well).

4.3 Generating training and test data

The features described above can be used to characterize the clustering variables $\hat{\mathbf{y}}$. For generating training data, we collected all the max-violating $\hat{\mathbf{y}}$ produced during LSP$_F$ (using Δ_F) learning and associate them with their correct MELA scores from the scorer. This way, we can have both training and test data for our regressor. In our experiments, for the generation purpose, we decided to run LSP$_F$ on each document separately to obtain more variability in $\hat{\mathbf{y}}$'s. We use a simple linear SVM to learn a model $\mathbf{w}_\rho$. Considering that MELA$(\mathbf{y}, \hat{\mathbf{y}})$ score lies in the interval $[100, 0]$, a simple approximation of the loss could be:

$$\Delta_\rho(\mathbf{y}, \hat{\mathbf{y}}) = 100 - \mathbf{w}_\rho \cdot \phi(\mathbf{y}, \hat{\mathbf{y}}). \quad (4)$$

Below, we show its improved version and an LSP for learning with it based on inexact search.

5 Learning with learned loss functions

Our experiments will demonstrate that Δ_ρ can be accurately learned from data. However, the features we used for this are not factorizable over the edges of the latent trees. Thus, we design a new LSP algorithm that can use our learned loss in an approximated max search.

5.1 A general inexact algorithm for CR

If the loss function can be factorized over tree edges (see Equation 3) the max-violating constraint in Line 6 of Alg. 1 can be efficiently found by exact decoding, e.g., using Edmonds' algorithm as in Fernandes et al. (2014) or Kruskal's as

Algorithm 3 Inexact Inference of a Max-violating Spanning Tree with a Global Loss

1: Input: training example $(\mathbf{x}, \mathbf{y})$; graph $G(\mathbf{x})$ with vertices V denoting mentions; set of the incoming candidate edges, $E(\mathbf{v})$, $\mathbf{v} \in V$; $\mathbf{w}$, ground truth tree $\mathbf{h}^*$
2: $\hat{\mathbf{h}} \leftarrow \emptyset$
3: $score \leftarrow 0$
4: **repeat**
5: $\quad prev\_score = score$
6: $\quad score = 0$
7: $\quad$ **for** $\mathbf{v} \in V$ **do**
8: $\quad\quad \mathbf{h} = \hat{\mathbf{h}} \setminus \mathbf{e}(\mathbf{v})$
9: $\quad\quad \hat{\mathbf{e}} = \underset{\mathbf{e} \in E(\mathbf{v})}{\text{argmax}} \langle \mathbf{w}, \mathbf{e} \rangle + C \times \Delta(\mathbf{y}, \mathbf{h}^*, \mathbf{h} \cup \mathbf{e})$
10: $\quad\quad \hat{\mathbf{h}} = \mathbf{h} \cup \hat{\mathbf{e}}$
11: $\quad\quad score = score + \langle \mathbf{w}, \hat{\mathbf{e}} \rangle$
12: $\quad$ **end for**
13: $\quad score = score + \Delta(\mathbf{y}, \mathbf{h}^*, \hat{\mathbf{h}})$
14: **until** $score = prev\_score$
15: **return** max-violating tree $\hat{\mathbf{h}}$

in Yu and Joachims (2009). The candidate graph, by construction, does not contain cycles, and the inference by Edmonds' algorithm does technically the same as the "best-left-link" inference algorithm by Chang et al. (2012). This can be schematically represented in Alg. 2.

When we deal with Δ_ρ, Alg. 2 cannot be longer applied as our new loss function is non-factorizable. Thus, we designed a greedy solution, Alg. 3, which still uses the spanning tree algorithm, though, it is not guaranteed to deliver the max-violating constraint. However, finding even a suboptimal solution optimizing a more accurate loss function may achieve better performance both in terms of speed and accuracy.

We reformulate Step 4 of Alg. 2, where a max-violating incoming edge $\hat{\mathbf{e}}$ is identified for a vertex $\mathbf{v}$. The new max-violating inference objective contains now a global loss measured on the partial structure $\hat{\mathbf{h}}$ built up to now plus a candidate edge $\mathbf{e}$ for a vertex $\mathbf{v}$ in consideration (Line 10 of Alg. 3). On a high level, this resembles the inference procedure of Wiseman et al. (2016), who use it for optimizing global features coming from an RNN. Differently though, after processing all the vertices, we repeat the procedure until the score of $\hat{\mathbf{h}}$ no longer improves.

Note that Björkelund and Kuhn (2014) perform inexact search on the same latent tree structures to extend the model to non-local features. In contrast to our approach, they use beam search and accumulate the early updates.

In addition to the design of an algorithm enabling the use of our Δ_ρ, there are other intricacies caused by the lack of factorization that need to be taken into account (see the next section).

Samples		# examples	MSE	SCC
Train	Test			
S_1	S_2	$6,011$	2.650	99.68
S_2	S_1	$5,496$	2.483	99.70

Table 1: Accuracy of the loss regressor on two different sets of examples generated from different documents samples.

5.2 Approaching factorization properties

The Δ_ρ defined by Equation 4 approximately falls into the interval $[0, 100]$. However, the simple optimal loss functions, Δ_F and Δ_{YJ}, output a value dependent on the size of the input training document in terms of edges (as they factorize in terms of edges). Since this property cannot be learned from MELA by our regression algorithm, we calibrate our loss with respect to the number of correctly predicted mentions, c, in that document, obtaining $\Delta'_\rho = \frac{c}{100}\Delta_\rho$.

Finally, another important issue is connected to the fact that on the way as we incrementally construct a max-violating tree according to Alg. 3, Δ_ρ decreases (and MELA grows), as we add more mentions to the output, traversing the tree nodes $\mathbf{v}$. Thus, to equalize the contribution of the loss among the candidate edges of different nodes, we also scale the loss of the candidate edges of the node $\mathbf{v}$ having order i in the document, according to the formula $\Delta''_\rho = \frac{i}{|V|}\Delta'_\rho$. This can be interpreted as giving more weight to the hard-to-classify instances – an important issue alleviated by Zhao and Ng (2010). Towards the end of the document, the probability of correctly predicting an incoming edge for a node generally decreases, as increases the number of hypotheses.

6 Experiments

In our experiments, we first show that our regressor for learning MELA approximates it rather accurately. Then, we examine the impact of our Δ_ρ on state-of-the-art systems in comparison with other loss functions. Finally, we show that the impact of our model is amplified when learning in smaller feature spaces.

6.1 Setup

Data We conducted our experiments on English and Arabic parts of the corpus from CoNLL 2012-Shared Task[4]. The English data contains 2,802, 343, and 348 documents in the training,

[4] conll.cemantix.org/2012/data.html

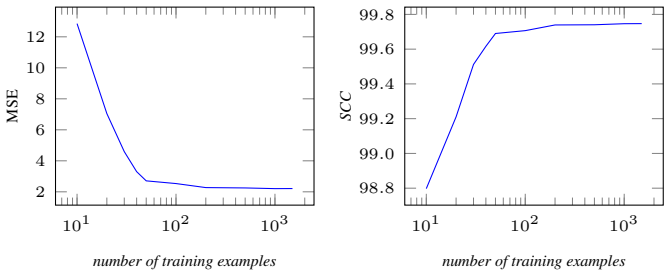

Figure 2: Regressor Learning curves.

dev. and test parts, respectively. The Arabic data includes 359, 44, and 44 documents for training, dev. and test sets, respectively.

Models We implement our version of LSP, where LSP_F, LSP_{YJ}, and LSP_ρ use the loss functions, Δ_F, Δ_{YJ}, and Δ_ρ, defined in Section 3.3 and 5.2, respectively. We used cort[5] – coreference toolkit by Martschat and Strube (2015) both to preprocess the English data and to extract candidate mentions and features (the basic set). For Arabic, we used mentions and features from BART[6] (Uryupina et al., 2012). We extended the initial feature set for Arabic with the feature combinations proposed by Durrett and Klein (2013), those permitted by the available initial features.

Parametrization All the perceptron models require tuning of a regularization parameter C. LSP_F and LSP_{YJ} – also tuning of a specific loss parameter r. We select the parameters on the entire dev. set by training on 100 random documents from the training set. We pick up $C \in \{1.0, 100.0, 1000.0, 2000.0\}$, the r values for LSP_F from the interval $[0.5, 2.5]$ with step 0.5, and the r values for LSP_{YJ} – from $\{0.05, 0.1, 0.5\}$. Ultimately, for English, we used $C = 1000.0$ in all the models; $r = 1.0$ in LSP_F and $r = 0.1$ in LSP_{YJ}. And wider ranges of parameter values were considered for Arabic, due to the lower mention detection rate: $C = 1000.0$, $r = 6.0$ for LSP_F, $C = 1000.0$, $r = 0.01$ for LSP_{YJ}, and $C = 5000.0$ – for LSP_ρ. A standard previous work setting for the number of epochs T of LSP is 5 (Martschat and Strube, 2015). Fernandes et al. (2014) noted that $T = 50$ was sufficient for convergence. We selected the best T from 1 to 50 on the dev. set.

Evaluation measure We used MUC, B^3, CEAF_e and their average MELA for evaluation, computed by the version 8 of the official CoNLL scorer.

[5] http://smartschat.de/software

[6] http://www.bart-coref.org/

Model	**Selected** ($N = 1M$)			**All** ($N \sim 16.8M$)		
	Dev.	Test	T_{best}	Dev.	Test	T_{best}
LSP_F	63.72	62.19	49	64.05	63.05	41
LSP_{YJ}	63.72	62.44	29	64.32	62.76	13
LSP_ρ	64.12	63.09	27	64.30	63.37	18
M&S AT	–	–	–	62.31	61.24	5
M&S MR	–	–	–	63.52	62.47	5
B&K	–	–	–	62.52	61.63	–
Fer	–	–	–	60.57	60.65	–

Table 2: Results of our and previous work models evaluated on the dev. and test sets following the exact CoNLL-2012 English setting, using all training documents with All and $1M$ features. T_{best} is evaluated on the dev. set.

6.2 Learning loss functions

For learning MELA, we generated training and test examples from LSP_F according to the procedure described in Section 4.3. In the first experiment, we trained the $\mathbf{w}_\rho$ model on a set of examples S_1, generated from a sample of 100 English documents and tested on a set of examples S_2, generated from another sample of the same size, and vice versa. The results in Table 1 show that with just $5,000/6,000$, the Mean Squared Error (MSE) is roughly between $\sim 2.4 - 2.7$: these are rather small numbers considering that the regression output values in the interval $[0, 100]$. Squared Correlation Coefficient (SCC) reaches a correlation of about 99.7%, demonstrating that our regression approach is effective in estimating MELA.

Additionally, Figure 2 shows the regression learning curves evaluated with MSE and SCC. The former rapidly decreases and, with about $1,000$ examples, reaches a plateau of around 2.3. The latter shows a similar behaviour, approaching a correlation of about 99.8% with real MELA.

6.3 State of the art and model comparison

We first experimented with the standard CoNLL setting to compare the LSP accuracy in terms of MELA using the three different loss functions, i.e., LSP_F, LSP_{YJ} and LSP_ρ. In particular, we used all the documents of the training set and all $N \sim 16.8$M features from cort, and tested on the both dev. and test sets. The results are reported in Columns All of Table 2.

We note first that our Δ_ρ is effective as it stays on a par with Δ_F and Δ_{YJ} on the dev. set. This is interesting as Corollary 1 shows that such functions can optimize MELA, the reported values refer to the optimal epoch numbers. Also, LSP_ρ improves the other models on the test set by 0.3 percent points (statistical significant at the 93% level of confidence).

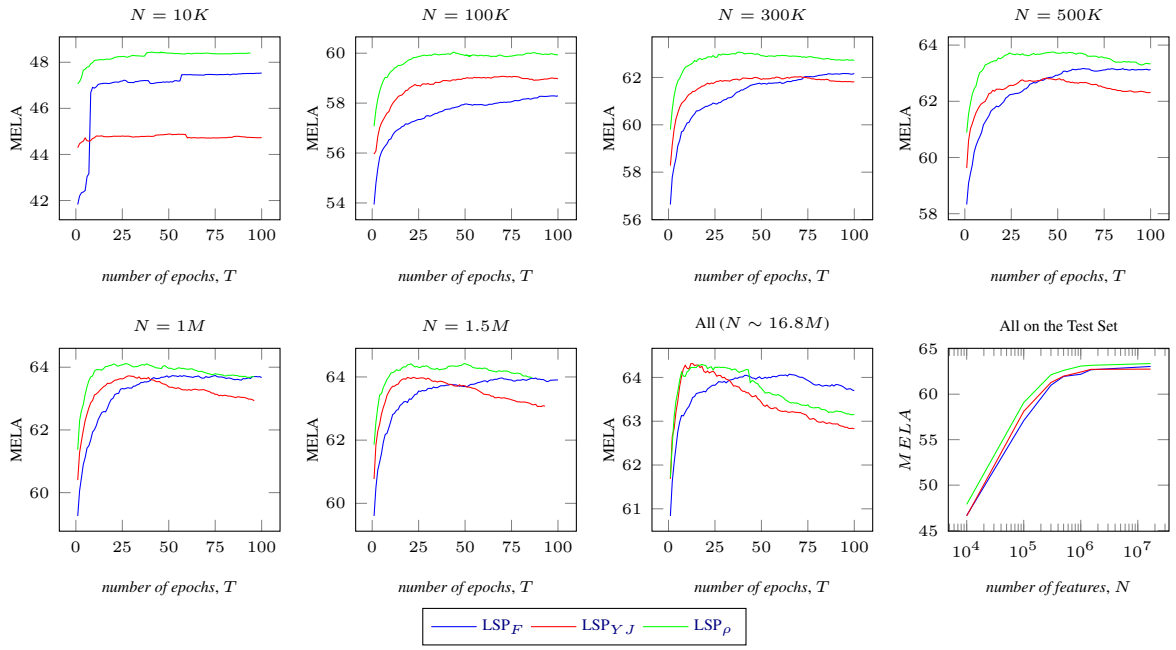

Figure 3: Results of LSP models on the dev. set using different number of features, N. The last plot reports MELA score on the test set of the models using the optimal number of epochs tuned on the dev. set.

#Feat.	Model	Test Set			
		MUC	B^3	$CEAF_e$	$MELA$
All	LSP$_F$	72.66	59.94	56.54	63.05
	LSP$_{YJ}$	72.18	59.31	55.82	62.76
	LSP$_\rho$	72.34	60.36	57.40	63.37
1M	LSP$_F$	71.95	59.03	55.59	62.19
	LSP$_{YJ}$	72.35	59.54	56.38	62.44
	LSP$_\rho$	72.09	60.11	57.07	63.09

Table 3: Results on the test set using the same setting of Table 2 and the measures composing MELA.

Secondly, all the three models improve the state of the art on CR using LSP, i.e., by Martschat and Strube (2015) using antecedent trees (M&S AT) or mention ranking (M&S MR), Björkelund and Kuhn (2014) using a global feature model (B&K) and Fernandes et al. (2014) (Fer). Noted that all the LSP models were trained on the training set only, without retraining on the training and dev. sets together, thus our scores can be improved.

Thirdly, Table 3 shows the breakdown of the MELA results in terms of its components on the test set. Interestingly, LSP$_\rho$ is noticeably better in terms of B^3 and CEAF$_e$, while LSP with simple losses, as expected, deliver higher MUC score.

Finally, the overall improvement of Δ_ρ is not impressive. This mainly depends on the optimality of the competing loss functions, which in a setting of $\sim$ 16.8M features, satisfy the separability condition of Proposition 1.

6.4 Learning in more challenging conditions

In these experiments, we verify the hypothesis that when the optimality property is partially or totally missing Δ_ρ is more visibly superior to Δ_F and Δ_{YJ}. As we do not want to degrade their effectiveness, the only condition dependent on the setting is the data inseparability or at least harder to be separated. These conditions can be obtained by reducing the size of the feature space. However, since we aim at testing conditions, where Δ_ρ is practically useful, we filter out less important features, preserving the model accuracy (at least when the selection is not extremely harsh). For this purpose, we use a feature selection approach using a basic binary classifier trained to discriminate between correct and incorrect mention pairs. It is typically used in non structured CR methods and has a nice property of using the same features of LSP (we do not use global features in our study). We carried out a selection using the absolute values of the model weights of the classifier for ranking features and then selecting those having higher rank (Haponchyk and Moschitti, 2017).

The MELA produced by our models using all the training data is presented in Figure 3. The first 7 plots show learning curves in terms of LSP epochs for different feature sets with increasing size N, evaluated on the dev. set. We note that: firstly, the fewer features are available, the better LSP$_\rho$ curves are than those of LSP$_F$ and LSP$_{YJ}$ in terms of accuracy and convergence speed. The intuition is that finding a separation of the training set (generalizing well) becomes more challenging (e.g., with 10k features, the data is not linearly sep-

arable) thus a loss function which is closer to the real measure provides some advantages.

Secondly, when using all features, LSP$_\rho$ is still overall better than the other models but clearly the latter can achieve the same MELA on the dev. set.

Thirdly, the last plot shows the MELA produced by LSP models on the test set, when trained with the best epoch derived from the dev. set (previous plots). We observe that LSP$_\rho$ is constantly better than the other models, though decreasing its effect as the feature number increases.

Next, in Column 1 (Selected) of Table 2, we report the model MELA using 1 million features. We note that LSP$_\rho$ improves the other models by at least 0.6 percent points, achieving the same accuracy as the best of its competitors, i.e., LSP$_F$, using all the features.

Finally, Δ_ρ does not satisfy Proposition 1, therefore, generally, we do not know if it can optimize any μ-type measure over graphs. However, being learned to optimize MELA, it clearly separates data maximizing such a measure. We empirically verified this by checking the MELA score obtained on the training set: we found that LSP$_\rho$ always optimizes MELA, iterating for fewer epochs than the other loss functions.

6.5 Generalization to other languages

Here, we test the effectiveness of the proposed method on Arabic using all available data and features. The results in Table 4 reveal an indisputable superiority of LSP$_\rho$ over the counterparts optimizing simple loss functions. They support the results of the previous section as we had to deal with the insufficiency of the expert-based features for Arabic. In such an uneasy case, LSP$_\rho$ was able to improve over LSP$_F$ by more than 4.7 points.

We also tested the loss model $\mathbf{w}_\rho$ trained for the experiments on the English data (resp. setting All of Section 6.3) in LSP$_\rho$ on Arabic. This corresponds to LSP$_\rho^{EN}$ model. Notably, it performs even better, 1.5 points more, than LSP$_\rho$ using a loss learned from Arabic examples. This suggests a nice property of data invariance of Δ_ρ. The improvement delivered by the "English" $\mathbf{w}_\rho$ is due to the fact that it was trained on the data which is richer: (i) quantitatively, since coming from almost 8 times more training documents in comparison to Arabic and (ii) qualitatively, in a sense of diversity with respect to the RL target value. Indeed, the Arabic data is much less separable than the English data and this prevents to have examples where MELA values are higher.

Model	**All** ($N \sim 395K$)		
	Dev.	Test	T_{best}
LSP$_F$	31.20	33.19	10
LSP$_{YJ}$	27.70	28.51	13
LSP$_\rho$	36.91	37.91	6
LSP$_\rho^{EN}$	**38.47**	**39.56**	12
Uryupina et al., 2012	–	37.54	–
B&K	46.67	48.72	–
Fer	–	45.18	–

Table 4: Results of our and baseline models evaluated on the dev. and test sets following the exact CoNLL-2012 Arabic setting, using all training documents. T_{best} is evaluated on the dev. set.

7 Conclusions

In this paper, we studied the use of complex loss functions in structured prediction for CR. Given the scale of our investigation, we limited our study to LSP, which is anyway considered state of the art. We derived several findings: (i) for the first time, up to our knowledge, we showed that a complex measure, such as MELA, can be learned by a linear regressor (RL) with high accuracy and effective generalization. (ii) The latter was essential for designing a new LSP based on inexact search and RL. (iii) We showed that an automatically learned loss can be optimized and provides state-of-the-art performance in a real setting, including thousands of documents and millions of features, such as CoNLL–2012 Shared Task. (iv) We defined a property of optimal loss functions for CR, which shows that in separable cases, such losses are enough to get the state of the art. However, as soon as separability becomes more complex simple loss functions lose optimality and RL becomes more accurate and faster. (v) Our MELA approximation provides a loss that is data invariant which, once learned, can be optimized in LSP on different datasets and in different languages.

Our study opens several future directions, ranging from defining algorithms based on automatically learned loss functions to learning more effective measures from expert examples.

Acknowledgements

We would like to thank Olga Uryupina for providing us with the preprocessed data from BART for Arabic. This work has been supported by the EC project CogNet, 671625 (H2020-ICT-2014-2, Research and Innovation action). Many thanks to the anonymous reviewers for their valuable suggestions.

References

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

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 47–57. http://www.aclweb.org/anthology/P/P14/P14-1005.

Jie Cai and Michael Strube. 2010. Evaluation metrics for end-to-end coreference resolution systems. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Stroudsburg, PA, USA, SIGDIAL '10, pages 28–36. http://dl.acm.org/citation.cfm?id=1944506.1944511.

Kai-Wei Chang, Rajhans Samdani, Alla Rozovskaya, Mark Sammons, and Dan Roth. 2012. Illinois-coref: The ui system in the conll-2012 shared task. In *Joint Conference on EMNLP and CoNLL - Shared Task*. Association for Computational Linguistics, Jeju Island, Korea, pages 113–117. http://www.aclweb.org/anthology/W12-4513.

Kevin Clark and Christopher D. Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 643–653. http://www.aclweb.org/anthology/P16-1061.

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

Jack Edmonds. 1967. Optimum branchings. *Journal of research of National Bureau of standards* pages 233–240.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Joint Conference on EMNLP and CoNLL - Shared Task*. Association for Computational Linguistics, Jeju Island, Korea, pages 41–48. http://www.aclweb.org/anthology/W12-4502.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2014. Latent trees for coreference resolution. *Computational Linguistics* 40(4):801–835.

Thomas Finley and Thorsten Joachims. 2005. Supervised clustering with support vector machines. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*. ACM, New York, NY, USA, pages 217–224. https://doi.org/10.1145/1102351.1102379.

Iryna Haponchyk and Alessandro Moschitti. 2017. A practical perspective on latent structured prediction for coreference resolution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 143–149. http://www.aclweb.org/anthology/E17-2023.

Emmanuel Lassalle and Pascal Denis. 2015. Joint anaphoricity detection and coreference resolution with constrained latent structures. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'15, pages 2274–2280. http://dl.acm.org/citation.cfm?id=2886521.2886637.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '05, pages 25–32. https://doi.org/10.3115/1220575.1220579.

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

Nafise Sadat Moosavi and Michael Strube. 2016. Which coreference evaluation metric do you trust? a proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 632–642. http://www.aclweb.org/anthology/P16-1060.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*. Association for Computational Linguistics, Jeju Island, Korea, page 1–40. http://www.aclweb.org/anthology/W12-4501.

Xu Sun, Takuya Matsuzaki, Daisuke Okanohara, and Jun'ichi Tsujii. 2009. Latent variable perceptron algorithm for structured classification. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI'09, pages 1236–1242. http://dl.acm.org/citation.cfm?id=1661445.1661643.

Olga Uryupina, Alessandro Moschitti, and Massimo Poesio. 2012. Bart goes multilingual:

The unitn/essex submission to the conll-2012 shared task. In *Joint Conference on EMNLP and CoNLL - Shared Task*. Association for Computational Linguistics, Stroudsburg, PA, USA, CoNLL '12, pages 122–128. http://dl.acm.org/citation.cfm?id=2391181.2391198.

Olga Uryupina, 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*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL Shared Task '11, pages 61–65. http://dl.acm.org/citation.cfm?id=2132936.2132944.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Message Understanding Conference*. pages 45–52.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 994–1004. http://aclweb.org/anthology/N/N16/N16-1114.pdf.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, New York, NY, USA, ICML '09, pages 1169–1176. https://doi.org/10.1145/1553374.1553523.

Shanheng Zhao and Hwee Tou Ng. 2010. Maximum metric score training for coreference resolution. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, Beijing, China, pages 1308–1316. http://www.aclweb.org/anthology/C10-1147.