10.12871/CLICIT2014139

Making Latent *SVM^{struct}* Practical for Coreference Resolution

Iryna Haponchyk[1] and **Alessandro Moschitti**[2,1]
[1]Department of Information Engineering and Computer Science, University of Trento,
[2]Qatar Computing Research Institute
iryna.haponchyk@unitn.it, amoschitti@gmail.com

Abstract

English. The recent work on coreference resolution has shown a renewed interest in the structured perceptron model, which seems to achieve the state of the art in this field. Interestingly, while SVMs are known to generally provide higher accuracy than a perceptron, according to previous work and theoretical findings, no recent paper currently describes the use of *SVM^{struct}* for coreference resolution. In this paper, we address this question by solving some technical problems at both theoretical and algorithmic level enabling the use of SVMs for coreference resolution and other similar structured output tasks (e.g., based on clustering).

Italiano. *Ricerca recente sulla risoluzione delle coreferenze linguistiche ha mostrato un rinnovato interesse per l'algoritmo del percettrone strutturato, il quale sembra essere lo stato dell'arte per questa disciplina. È interessante notare che, mentre l'esperienza passata e i risultati teorici mostrano che le SVMs sono piú accurate del percettrone, nessun articolo recente descrive l'uso di SVM^{struct} per la risoluzione di coreferenze. In questo articolo, si prova a dare una risposta a tale domanda, risolvendo alcuni problemi tecnici, sia a livello teorico che algoritmico, cosí consentendo l'utilizzo delle SVMs per la risoluzione delle coreferenze e altri problemi che richiedono l'uso di funzioni di output strutturato (e.g., basati su clustering).*

1 Introduction

Coreference resolution (CR) is a complex task, in which document phrases (mentions) are partitioned into equivalence sets. It has recently been approached by applying learning algorithms operating in structured output spaces (Tsochantaridis et al., 2004). Considering the nature of the problem, i.e., the NP-hardness of finding optimal mention clusters, the task has been reformulated as a spanning graph problem.

First, Yu and Joachims (2009) proposed to (i) represent all possible mention clusters with fully connected undirected graphs and (ii) infer document mention cluster sets by applying Kruskal's spanning algorithm (Kruskal, 1956). Since the same clustering can be obtained from multiple spanning forests (there is no one-to-one correspondence), these latter are treated as hidden or latent variables. Therefore, an extension of the structural SVM – *Latent SVM^{struct}* (LSVM) – was designed to include these structures in the learning procedure.

Later, Fernandes et al. (2012) presented their CR system having a resembling architecture. They do inference on a directed candidate graph using the algorithm of Edmonds (1967). This modeling coupled with the latent structured perceptron delivered state-of-the-art results in the CoNLL-2012 Shared Task (Pradhan et al., 2012).

To the best of our knowledge, there is no previous work on a comparison of the two methods, and the LSVM approach of Yu and Joachims has not been applied to the CoNLL data. In our work, we aim, firstly, at evaluating LSVM with respect to the recent benchmark standards (corpus and evaluation metrics defined by the CoNLL-shared task) and, secondly, at understanding the differences and advantages of the two structured learning models. In a closer look at the LSVM implementation[1], we found out that it is restricted to inference on a fully-connected graph. Thus, we provide an extension of the algorithm enabling to op-

[1] http://www.cs.cornell.edu/~cnyu/latentssvm/

erate on an arbitrary graph: this is very important as all the best CR models exploit heuristics to prefilter edges of the CR graph. Therefore our modification of LSVM allows us to use it with powerful heuristics, which greatly contribute to the achievement of the state of the art. Regarding the comparison with the latent perceptron of Fernandes et al. (2012), the results of our experiments provide evidence that the latent trees derived by Edmonds' spanning tree algorithm better capture the nature of CR. Therefore, we speculate that the use of this spanning tree algorithm within LSVM may produce higher results than those of the current perceptron algorithm.

2 Structured Perceptron vs. *SVM^{struct}*

In this section, we briefly describe the basics of the widely known structured prediction framework. Structured learning algorithms aim at discovering patterns that relate input to complex (thus generally structured) output. Formally, they seek for a mapping $f : X \times Y \rightarrow \mathbb{R}$ over a combined feature space of input variables X and output variables Y, where predictions are derived by finding the $\underset{\mathbf{y}\in Y}{\operatorname{argmax}} f(\mathbf{x}, \mathbf{y})$. The function $f(\mathbf{x}, \mathbf{y})$ is often assumed to be linear with respect to $\Phi(\mathbf{x}, \mathbf{y})$, which is a *joint feature vector* representing an input example together with its associated output. In other words, we have a linear function of the type: $f(\mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y})$. The structured perceptron learning consists in iterating over the entire training set $\{(\mathbf{x_i}, \mathbf{y_i})\}_{i=1,..,l}$ of the following operations: (i) find the optimal output:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}\in Y}{\operatorname{argmax}} f(\mathbf{x_i}, \mathbf{y})$$

(given the current weight $\mathbf{w}$) and (ii) update $\mathbf{w}$ as follows: $\mathbf{w} \leftarrow \mathbf{w} + \Phi(\mathbf{x_i}, \mathbf{y_i}) - \Phi(\mathbf{x_i}, \hat{\mathbf{y}})$ when prediction errors occur, i.e., $\hat{\mathbf{y}} \neq \tilde{\mathbf{y}}$, where $\tilde{\mathbf{y}}$ is the gold standard output. The structured perceptron algorithm dates back to the early work of Collins (2002), who provided its theoretical guarantees and proof of convergence.

SVMs outperform perceptron in terms of generalization accuracy. They were extended by Tsochantaridis et al. (2004) to deal with structured output spaces. In a standard form, the optimization problem is formulated as

$$\min_{\mathbf{w}} \tfrac{1}{2}\|\mathbf{w}\|^2$$

s.t.$\forall \mathbf{i}, \forall \mathbf{y} \in Y \setminus \mathbf{y_i}$:$\langle \mathbf{w}, \Phi(\mathbf{x_i}, \mathbf{y_i}) - \Phi(\mathbf{x_i}, \mathbf{y})\rangle \geq 1$. The set of margin constraints in the above formulation may be exponentially large or even infinite when Y, for example, is the space of subtrees in syntactic parsing or the space of strings in the sequence labeling task. However, it was shown that using the sparseness of Y and structure and dependencies in Φ, one can drastically reduce the number of constraints to be examined, which makes the optimization feasible. A general SVM algorithm for predicting structured outputs, as well as its instantiations for several complex prediction tasks, was implemented in *SVM^{struct}* and made publicly available[2].

CR is essentially modelled as a clustering problem. Considering a clustering of a document mention set a desired output of a predictor, one can approach the task with a learning algorithm operating in the output space Y of all possible clusterings. Further, we describe two structured learning methods, applied to CR, that were able to overcome the intractability of search for an optimal clustering in Y.

3 Corereference resolution with SVMs

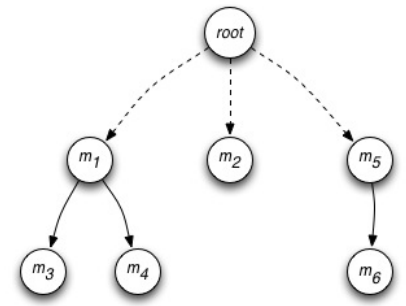
Latent SVM^{struct} was introduced by Yu and Joachims (2009), who construct an undirected graph for each document (Figure 1b). The authors reformulate the *structural SVM* of Tsochantaridis et al. (2004) introducing *latent* variables into a learning procedure. In the LSVM formulation, an input-output example is, thus, described by a tuple $(\mathbf{x}, \mathbf{y}, \mathbf{h})$, where $\mathbf{x}$ is a document mention set, $\mathbf{y}$ is a corresponding clustering and $\mathbf{h}$ is a latent variable. $\mathbf{h}$ is consistent with $\mathbf{y}$ in a way that for training examples, $\mathbf{h}$ contains only links between mention nodes that are coreferent according to $\mathbf{y}$. For test examples a clustering $\mathbf{y}$ is, instead, imposed by an $\mathbf{h}$ automatically generated by the classification algorithm. The joint feature vector decomposes along the edges of $\mathbf{h}$:

$$\Phi(\mathbf{x}, \mathbf{y}, \mathbf{h}) = \sum_{e\in\mathbf{h}} \phi(e).$$

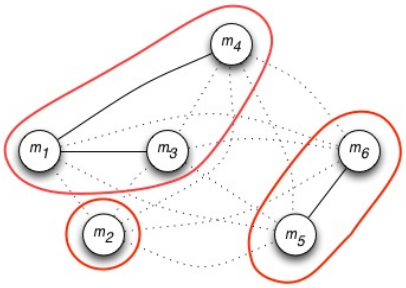
The learning procedure involves running Kruskal's algorithm for finding a maximum spanning forest of a graph containing all possible links between mentions. The resulting spanning forest, in which each connected component corresponds to a separate cluster (in Figure 1b clusters are circled), is a desired $\mathbf{h}$.

The LSVM implementation provided by the authors follows the *SVM^{struct}* API paradigm. In our

[2]It is a software package for implementing structural SVMs available at `http://svmlight.joachims.org/svm_struct.html`



(a)



(b)

Figure 1: Graphical models employed in structured learning algorithms

experiments, we analysed one of the LSVM specializations that is designed for CR. The inference by Kruskal's spanning algorithm is done on a fully-connected graph of mention pair relations. However, a large portion of mention pair links apparently do not convey significant information because they connect non-coreferring mentions, e.g., very distant mentions are very improbable to corefer. Thus, it has been a common practice in coreference research to adopt a preliminary strategy for mention pair filtering, e.g., Fernandes et al. (2012) preprocess the data by applying a set of linguistic filters (the so called sieves). This issue becomes crucial in the LSVM setting as Kruskal's algorithm includes sorting all document edges (this number is exponential in the number of mentions) by their weight. In our work, we intended to enable the LSVM implementation to operate on non-complete candidate graphs, i.e., whose edges have been filtered by some strategies.

4 Coreference Resolution with Latent Perceptron

The latent perceptron of Fernandes et al. (2012) is related to the earlier work of Yu and Joachims (2009) but they model CR as a spanning tree problem. They introduce document trees (Figure 1a), in which nodes represent mentions and edges – relations between them, plus an additional root node. The subtrees directly connected to the root node of such a tree form clusters. To obtain this tree, Edmonds' algorithm is run on a directed candidate graph of document mention relations. Such trees are implicit in data and hence are called latent. This modeling is incorporated into a latent perceptron framework in its loss-augmented formulation. It achieved the best results in the CoNLL 2012-Shared Task (Pradhan et al., 2012).

Edmonds' algorithm iterates over the tree nodes and chooses the best incoming edge (edge of maximum weight). By that means, the best antecedent is chosen for each mention (or no antecedent if the chosen edge starts in the root node). This strategy thereby fits the nature of the CR task very well.

5 Adapting *Latent SVM^{struct}* to filtered data

As mentioned before, we intend to enable the use of LSVM on filtered graphs, i.e., when some candidate edges between mention nodes are missing. The theoretical description of the algorithm does not impose any limitation on the use of partial graphs. However, the provided implementation requires fully-connected graphs. Indeed, a bare execution of LSVM on the partial data results into a low performance score (see Table 1).

In the implementation, each mention is assigned with an ID of the cluster it belongs to, which is chosen according to the rule $clusterID(m_i) = min_i\{m_i \cup \{m_j : \exists$ a positive edge between m_i and $m_j\}\}$, where m are the IDs of the mentions. Let us suppose that we have a cluster with 4 mentions $K = \{m_1, m_2, m_3, m_4\}$ (mentions receive an ID, corresponding to the order of their appearance in the document). If we are provided with all the edges then we surely obtain $\forall i = 1..4, clusterID(m_i) = m_1$. However, if an edge, e.g., (m_1, m_3), is missing, $clusterID(m_3) = m_2$ and it would differ from the cluster ID of the other coreferring mentions. Thus, we made the necessary modifications to the LSVM program code, which resolve the above problem by activating the following rule: $clusterID(m_i) = min\{m_i \cup \{m_j : \exists$ a positive route connecting m_i and $m_j\}\}$.

Another program issue requiring an adjustment is the construction of a gold spanning forest for the first iteration. In the original version of software, this is done by connecting consecutively the cluster edges. For the aforementioned cluster K, chain $\{(m_1, m_2), (m_2, m_3), (m_3, m_4)\}$ would

	All edges		Filtered edges	
Scorer Version	v4	v7	v4	v7
Original LSVM	60.22	56.56	53.15	46.67
Modified LSVM	60.22	56.56	60.31	57.18

(a) development set

	All edges		Filtered edges	
Scorer Version	v4	v7	v4	v7
Original LSVM	59.61	55.19	52.85	46.03
Modified LSVM	59.61	55.19	59.71	56.09

(b) test set

Table 1: Performance of the LSVM implementations on the English part of the CoNLL-2012 dataset.

be output. However, this is not a valid manner when instead of the entire graphs, some edges are filtered. Our modification therefore connects each mention m_i to $min\{m_j : m_j > m_i, \exists$ a positive edge between m_i and $m_j\}$.

Beside the other insignificant changes to the program code, our adjustments enabled us to train the LSVM on thoroughly filtered data while reaching basically the same performance as in the fully-connected case.

6 Experiments

In all our experiments, we used the English part of the corpus from the CoNLL 2012-Shared Task[3], which comprises 2,802, 343 and 348 documents for training, development and testing, respectively. We report our results in terms of the MELA score (Pradhan et al., 2012) computed using the versions 4 and 7 of the official CoNLL scorer. Our feature set is composed of *BART*[4](Versley et al., 2008) and some Fernandes et al. features.

Table 1(a) reports our experiments on the development set, using the original LSVM (Row 1) and our modified version enabling the use of filters (Row 2). The first column regards the use of a fully-connected coreference graph. The numbers confirm that we do not introduce any errors to the implementation since we obtain equal performance as with the original algorithm (v4 and v7 are different scorers). The results in the rightmost column are more interesting as they show that the original LSVM loses up to 10 absolute percent points whereas the modified version obtains practically the same results as when using unfiltered graphs. It should be noted that we use here only 3.94% of edges: this corresponds to a substantial speed-up of the learning and classification phases. Table 1(b) illustrates the same trend on the test set.

[3] http://conll.cemantix.org/2012/data.html

[4] http://bart-anaphora.org

	All edges		Filtered edges	
Scorer Version	v4	v7	v4	v7
Development	61.68	58.25	61.78	58.89
Test	61.21	57.64	61.23	57.90

Table 2: Accuracy of our implementation of the Latent Perceptron of Fernandes et al. (2012) on the English part of the CoNLL-2012 dataset.

In Table 2, we report the performance of our implementation of the modelling of Fernandes et al., showing that the perceptron model unexpectedly outperforms LSVM in all the settings. The main difference of the methods is that LSVM finds a global optimum[5], whereas perceptron simply finds a solution. We thus would expect higher accuracy from LSVM. However, LSVM uses graphs instead of trees along with a different spanning tree algorithm, i.e., Kruskal's vs. Edmond's used by the Latent Perceptron.

To shed some light on this question, we implemented the latent perceptron with the graph model and Kruskal's spanning algorithm as it is done in LSVM. Due to the time constraints, we could train this perceptron implementation only on a part of the filtered training set, constituted by 363 out of all $2,802$ documents. We obtained 58.79(v4) and 55.43(v7) on the development set. These results are lower than what we obtained with LSVM on the same data, i.e., 59.51(v4), 56.22(v7). Additionally, the same perceptron but using latent trees and Edmonds' algorithm scored 61.37(v4) and 58.33(v7). This suggests that Edmonds' spanning tree algorithm is superior to Kruskal's for CR and LSVM using it may outperform the latent perceptron.

7 Conclusions

We have performed a comparative analysis of the structured prediction frameworks for coreference resolution. Our experiments reveal that the graph modelling of Fernandes et al. and Edmonds' spanning algorithm seem to tackle the task more specifically. As a short-term future work, we intend to verify if LSVM benefits from using Edmonds' algorithm. We have also enabled the LSVM implementation to operate on partial graphs, which allows the framework to be combined with different filtering strategies and facilitates its comparison with other systems.

[5] Although, in latent methods, this is often not true as the data is not separable.

Acknowledgments

The research described in this paper has been partially supported by the EU FP7 grant #288024: LIMOSINE – Linguistically Motivated Semantic aggregation engiNes.

References

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 - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

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

Joseph B. Kruskal. 1956. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. In *Proceedings of the American Mathematical Society, 7*.

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*, page 1–40, Jeju Island, Korea, July. Association for Computational Linguistics.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 104–, New York, NY, USA. ACM.

Yannick Versley, Simone Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang, and Alessandro Moschitti. 2008. Bart: A modular toolkit for coreference resolution.

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*, ICML '09, pages 1169–1176, New York, NY, USA. ACM.