A Practical Perspective on Latent Structured Prediction for Coreference Resolution

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

Latent structured prediction theory proposes powerful methods such as Latent Structural SVM (LSSVM), which can potentially be very appealing for coreference resolution (CR). In contrast, only small work is available, mainly targeting the latent structured perceptron (LSP). In this paper, we carried out a practical study comparing for the first time online learning with LSSVM. We analyze the intricacies that may have made initial attempts to use LSSVM fail, i.e., a huge training time and much lower accuracy produced by Kruskal's spanning tree algorithm. In this respect, we also propose a new effective feature selection approach for improving system efficiency. The results show that LSP, if correctly parameterized, produces the same performance as LSSVM, being at the same time much more efficient.

1 Introduction

Recent research on CR has shown effective applications of structured prediction, e.g., the latent structured perceptron (LSP) by Fernandes et al. (2014) obtained the top rank in the CoNLL-2012 Shared Task (Pradhan et al., 2012). There has been an exploration of LSP variants (Chang et al., 2011; Björkelund and Kuhn, 2014; Lassalle and Denis, 2015), and also of SGD-like methods (Chang et al., 2013; Peng et al., 2015; Kummerfeld et al., 2015). Surprisingly, no study was devoted to LSSVM by Yu and Joachims (2009), which offers theoretical guarantees on reducing the error upper-bound. The major advantage of such a theory is the possibility to stop the optimization process, carried out using the Concave-Convex Procedure (CCCP) by Yuille and Rangarajan (2003), when the approximation to the optimum is close as much as we want. In contrast, the gradient descent operated by perceptron-like algorithms does not allow us to estimate how much our solution is far away from the optimum. In other words, we do not know at which epoch our algorithm should stop. Thus, LSSVM holds an important advantage over online methods.

In this paper, we empirically compare LSSVM with two online learning algorithms, LSP and LSPA (a structured passive-aggressive (PA) algorithm (Crammer et al., 2006) that we extended with latent variables) using the exact setting of the CoNLL-2012 dataset. This preserves comparability with the work in CR. For example, we use the latest version of the MELA scorer[1].

It should be noted that implementing a sound comparison was rather complex as it required testing all the algorithms in the same conditions and optimally setting their parameters. In particular, LSSVM and LSP adopt different graph models and use different methods to extract spanning trees from a document graph, namely, Kruskal's (Kruskal, 1956) and Edmonds' (Chu and Liu, 1965; Edmonds, 1967). Although both extract optimal spanning trees, they provide different solutions, which critically impact on accuracy and efficiency. The latter is problematic as LSSVM requires too long time for convergence on the large CoNLL dataset.

To tackle this issue, we applied two kinds of efficiency boost: feature and mention pair selection. Feature selection was rather challenging as the CR feature space is different from a standard text categorization setting. We could not apply a filtering threshold on simple and effective statistics such as document frequency since almost all the features appear in many documents. For solving this problem, we explored the use of efficient binary SVMs for computing feature weights, which we used for

[1] conll.cemantix.org/2012/software.html

our selection. Additionally, we also provided a parallelized version of LSSVM to afford the computation requirement of the full CoNLL dataset.

The results of our study show that LSSVM can be trained on large data and achieve the state of the art of online methods. However, the latter using optimal parameters can even surpass its accuracy and outperform the current state of the art of LSP by 2 points. Finally, our feature selection algorithm is rather efficient and effective.

2 Related Work

The first work of structured prediction for CR is an *SVM^{cluster}* approach by Finley and Joachims (2005), who couple the structural SVM (Tsochantaridis et al., 2004) with approximate clustering inference. They maximize the clustering objective by either (i) a simple greedy approach or (ii) a relaxation of the correlation clustering technique. Both methods resulted computationally very expensive. To overcome such inefficiency, Yu and Joachims (2009) proposed LSSVM performing inference on undirected (latent) graphs built on document mentions using Kruskal's spanning algorithm.

Fernandes et al. (2014) specialized the latent structured perceptron proposed by Sun et al. (2009) for solving CR tasks (LSP). This is based on (i) the Minimum Spanning Tree algorithm on the directed mention graph and (ii) the structured perceptron, updated on a per-document basis.

The same approach, referred to as *antecedent trees*, is included in the generalized latent structure framework of Martschat and Strube (2015). The authors report that the mention-ranking approach, which uses the LSP inference and mention-based updates[2], produces slightly better results.

It should be noted that the LSP inference is equivalent to the best-left-link inference of Chang et al. (2013), who coupled it with SGD updates on a per-mention basis. Chang et al. (2011, 2012, 2013); Peng et al. (2015) reformulated the best-left-link in terms of Integer Linear Programming inference.

Björkelund and Kuhn (2014) experimented with updates both on a per-mention and document basis to enable inference with non-local features. Lassalle and Denis (2015) experimented with a similar inference procedure by also jointly modeling anaphoricity and mention coreference.

[2]A perceptron update is performed after selecting the best antecedent for a mention.

Model	Parameters
LSSVM^K	$C = 100.0$ $r = 0.5$
LSSVM^E	$C = 100.0$ $r = 1.0$
LSP^K	$C = 1000.0$ $r = 0.1$
LSP^E	$C = 1000.0$ $r = 1.0$

Table 1: Best parameter combinations.

In summary, although many models have been tested, LSSVM has never been trained on a realistic CR dataset. Chang et al. (2013) tested it on the CoNLL-2012 dataset but they could not use CCCP, exactly for efficiency reasons, and thus they applied an SGD approach.

2.1 Algorithm Equivalence

LSSVM, LSP, LSPA can reach the same accuracy subject to different convergence rates and bounds. Indeed, LSSVM solves an optimization problem using a CCCP iteration, the cost of the latter is nearly a cost of one *SVM^{struct}* problem, which in turn is polynomial.

LSP and LSPA require linear times, however, in contrast to LSSVM, they do not have stopping criteria - the number of epochs T has to be set. The CCCP procedure is guaranteed to converge to a local minimum or a saddle point. LSP and LSPA, in essence, perform an update, which is equivalent, up to some constant, to an SGD update of the LSSVM objective, with a gradient taken w.r.t. a document variable.

They can approach the local minimum as close as possible, which is supported by our experiments, reflecting the results compatible among the three algorithms. For LSP and LSPA though, we do not know a priori when to stop training. While, for LSPA, there are error bounds derived by Crammer et al. (2006), there are no bounds for LSP at all.

However, for CR, as it can be seen from our experiments, values of T for LSP and LSPA can be reliably selected on a validation set for a fixed training data size and a choice of features/instances. Since the algorithms optimize a surrogate objective, it is often the case that accurately tuned LSP and LSPA result in higher performance than LSSVM, not mentioning an excessive complexity of the latter.

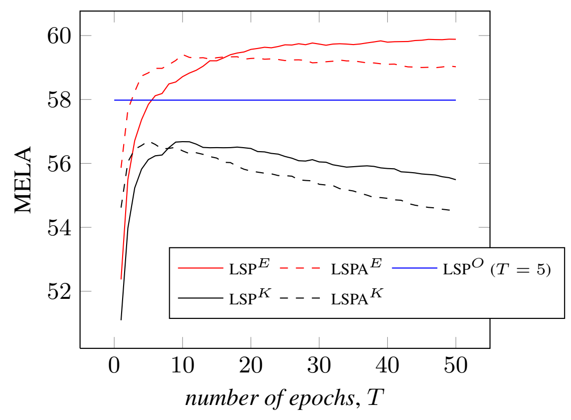

Figure 1: LSP learning curves, with 100 random documents used for training (all the features, all the edges), tested on all the dev. documents.

Model	Dev.	Test	T_{best}	Time, h
LSSVM^K	61.03	59.89	–	1164.09
LSSVM^E	62.91	61.88	–	210.01
LSP^K	61.08	60.00	10	27.77
LSP^E	64.01	**63.04**	43	32.55
LSPA^K	61.15	60.16	6	47.73
LSPA^E	64.14	62.81	8	37.33
LSP^O	62.92	62.00	5	–
*LSP^O	62.31	61.24 [5]	5	–

Table 2: Main results for the systems evaluated on CoNLL-2012 English development and test sets, using all the training documents for training. T_{best} is evaluated on the development set and used on the test set. *LSP^O is the result published in Martschat and Strube (2015).

3 Experiments

3.1 Setup

Data We performed our experiments on the English part of the corpus from CoNLL 2012-Shared Task[3], containing 2,802, 343 and 348 documents for training, development and test sets, respectively.

Evaluation measure We report our coreference results in terms of the MELA score (Pradhan et al., 2012) computed using the version 8 of the official CoNLL scorer.

Models and software As baselines, we used (i) the original implementation of the Latent *SVM^{struct}* [4] (denoted as LSSVM^K) performing inference on undirected graphs using Kruskal's spanning algorithm, (ii) LSP^E – our implementation of the LSP algorithm with a tree modeling of Fernandes et al. (2014) and Edmonds' spanning tree algorithm, (iii) *cort* – coreference toolkit by Martschat and Strube (2015), precisely its antecedent tree approach, encoding, as well as LSP^E, the modeling of Fernandes et al. (denoted as LSP^O, where "O" stands for Original).

In LSP^E, the candidate graph, by construction, does not contain cycles, and the inference by Edmonds' algorithm is reduced to selecting for each node an incoming edge with a maximum weight, in other words, the best antecedent or no antecedent for each mention. Thus, the difference between our LSP^E and cort is only due to a different implementation.

Along with the baselines, we consider the following models: (i) LSSVM^E, i.e., LSSVM with the latent trees and Edmonds', (ii) LSP^K, i.e., LSP using Kruskal's on undirected graphs, and (iii) two structured versions of the PA online learning algorithms, LSPA^E and LSPA^K.

We employed the cort toolkit both to preprocess the CoNLL data and to extract candidate mentions and features (the basic cort feature set).

As emphasized by Fernandes et al., averaging the perceptron weights renders the learning curve rather smooth. We applied weight averaging in all the LSP and LSPA variants.

Parametrization All the models require tuning of a regularization parameter C and of a specific loss parameter r. In LSSVM^K and LSP^K, r is a penalty for adding an incorrect edge; in LSSVM^E and LSP^E, r is a penalty for selecting an incorrect root arc. We selected the parameters on the entire development set by training on 100 random documents from the training set. We picked a C from $\{1.0, 100.0, 1000.0, 2000.0\}$, the r values for LSSVM^K and LSP^K from $\{0.05, 0.1, 0.5\}$, and the r values for LSSVM^E and LSP^E from the interval $[0.5, 2.5]$ with step 0.5. The values reported in Table 1 were used for all our experiments.

3.2 Selecting the epoch number

A standard previous work setting for the number of epochs T of the online learning algorithms is 5 (Martschat and Strube, 2015). Fernandes et al.

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

[4]www.cs.cornell.edu/~cnyu/latentssvm/

[5]This result is obtained using a concatenation of the training and the development set.

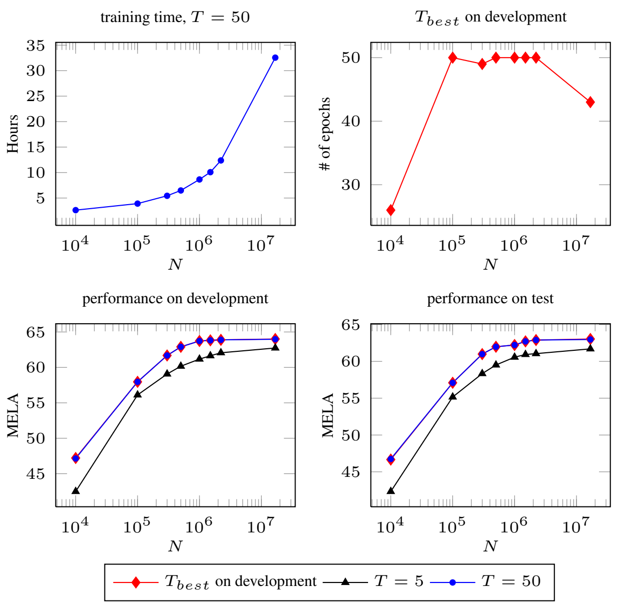

Figure 2: LSP^E training time and accuracy with respect to the number of features N, selected according to the binary classifier weights.

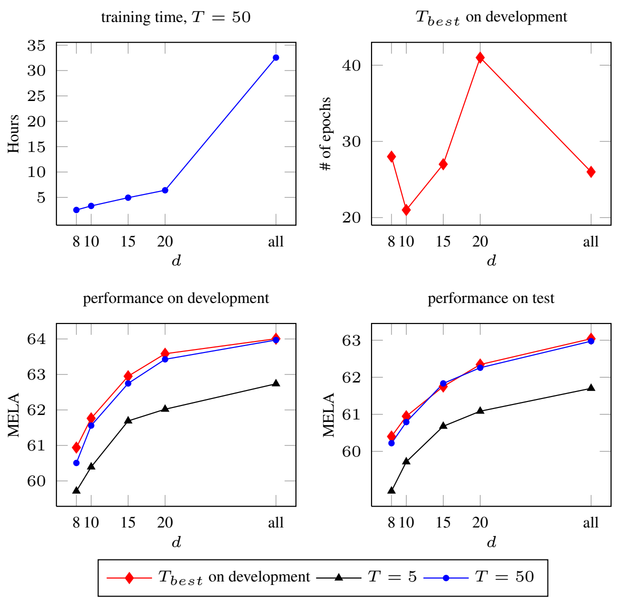

Figure 3: LSP^E training time and accuracy with respect to d (max number of candidate antecedent edges for each mention).

(2014) noted that $T = 50$ was sufficient for convergence. Figure 1 shows that setting T is crucial for achieving a high accuracy. We also note that the dataset size and the selected sets of features and/or instances highly affect the best epoch number, thus, for each particular experiment, we selected the best T from 1 to 50 on the dev. set.

3.3 Model Comparison

Table 2 reports the results of the models trained on the entire training set, and the numbers of epochs T_{best} for LSP and LSPA, tuned on the development set. LSP^O denotes the result of our run of the original cort software. We note that (i) LSP and LSPA perform on a par in both the settings; (ii) the latent trees used with Edmonds' algorithm outperform the undirected graphs used with Kruskal's; (iii) LSSVM^E is around one point less than LSP^E and LSPA^E; (iv) the training time of LSSVM^E is one order of magnitude longer than that of LSP^E; and (v) LSSVM^K took more than 1.5 months to converge.

3.4 Feature Selection

The number of distinct features extracted from cort and used for training in the above experiments is around 16.8 millions. Training systems with such a large model size is nearly prohibitive, this especially concerns SVMs, which may require a substantial number of iterations for convergence.

We tried to filter out less relevant features removing those that appear in a fewer number of documents but these were too few, e.g., less than 1% of all features have document frequency ≤ 3.

Thus, we proposed a feature selection technique consisting in (i) training a binary classification model, $\vec{w}$, on all mention-pair feature vectors and (ii) removing features with lower absolute weights in $\vec{w}$. Figure 2 plots the accuracy of CR models, using different numbers of features selected as described above. Interestingly, only retaining 5% of the features ($N = 10^6$) results in a small loss.

3.5 Candidate edge selection

Using all the candidate edges in the CR graph is another cause of computational burden, which is overcome by the best CR systems by exploiting heuristic linguistic filters.

In cort, filtering is not implemented and all the candidate edges are used for training. We simply adopted one of the filters, the so-called sieves, of Fernandes et al. (2014) to reduce the number of candidate links. Such a sieve retains links between two mentions only if their distance is lower than or equal to d, i.e., we consider only links (m_i, m_j) with $|j - i| \leq d$. Fernandes et al. use $d = 8$.

Figure 3 shows that, although the training time is reduced considerably, the accuracy suffers. In our experiments, we used $d = 20$, which causes a loss smaller than 0.5 in MELA. It should be noted that we also had to enable the LSSVM implementation to operate on non-complete candidate graphs as it was originally designed for making inference on fully-connected graphs only (Haponchyk and Moschitti, 2014).

3.6 Results on Filtered Data

Table 3 reports the results using filtering corresponding to the setting $N = 10^6$, $d = 20$. We note that (i) the training time is reduced by more than 10 times; (ii) LSSVM^K is outperformed by LSP^K (2 points) and performs worse than LSSVM^E; (iii) LSPA^K seems to generalize better on filtered data than LSP^K; and (iv) w.r.t. no filtering, LSSVM^E faces a lower drop in performance than LSP^E does, approaching nearer to the latter.

3.7 Discussion

The results of our study are the following:

(i) for the first time, we show that LSSVM can be applied to a realistic CR dataset and achieve the same state of the art of the online methods;

(ii) although the optimum found by CCCP produces better results than online learning algorithms, the latter, when parameterized, provide similar accuracy, while at the same time being much more efficient;

(iii) in this respect, we studied the optimal model parameterization and found that LSP can be highly improved, almost 2 points (63.04 vs. 61.24) over the previous best LSP result, by accurately selecting the number of epochs on a validation set;

(iv) the results of all the approaches using an undirected graph model coupled with Kruskal's are $3 - 7$ absolute percent points lower than their results obtained with a directed tree model coupled with Edmonds'. Our outcome is supported by Chang et al. (2013) who employed a fast SGD approach with the best-left-link inference, which is equivalent to Edmonds' algorithm applied to the directed latent trees. They compared the previous inference approach with the spanning graph algorithm by Kruskal on undirected graphs. They explain that the better accuracy of the first method is due to the fact that the latent tree structure considers the order of the mentions in the document. Apart from that, by using an artificial root, it implicitly models the cluster initial elements (i.e., discourse-new mentions).

(v) The use of direct trees in Edmonds' method delivers comparable results among all the algorithms; and

(vi) our new approach to feature selection based on binary SVMs turned out to be efficient and effective and, together with mention pair instance filtering, sped up training by 88% only losing 0.15 of a point in accuracy.

Model	Dev.	Test	T_{best}	Time, h
LSSVM^K	56.16	54.50	–	23.06
LSSVM^E	62.82	61.75	–	24.09
LSP^K	57.98	56.81	6	1.82
LSP^E	63.11	61.98	49	1.62
LSPA^K	58.69	57.38	3	3.50
LSPA^E	63.28	62.11	6	1.98

Table 3: Main results for the systems evaluated on CoNLL-2012 English development and test sets, using all training documents with filtered features (N=10^6) and edges (d=20).

4 Conclusions

This work provides a comparative analysis of online and batch methods for structured prediction in CR. Although LSSVM can reliably select a stopping point of its learning, LSP and LSPA, when well parameterized, can achieve the same accuracy. This empirically demonstrates that all these methods, inherently optimizing the same objective, are able to achieve the same optimum.

Additionally, we show a very positive impact of our new feature selection method for CR, based on a pairwise classifier, which we can efficiently train thanks to linear SVMs.

Finally, we also demonstrate that a noticeable benefit to all online methods comes from accurately parameterizing the epoch number. The latter is rather stable between development and test sets but must be parametrized when using different training data, feature or instance sets.

Acknowledgements

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

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, pages 47–57.

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

Kai-Wei Chang, Rajhans Samdani, Alla Rozovskaya, Nick Rizzolo, Mark Sammons, and Dan Roth. 2011. *Inference Protocols for Coreference Resolution*, Association for Computational Linguistics, chapter Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task, pages 40–44.

Kai-Wei Chang, Rajhans Samdani, Alla Rozovskaya, Mark Sammons, and Dan Roth. 2012. *Joint Conference on EMNLP and CoNLL - Shared Task*, Association for Computational Linguistics, chapter Illinois-Coref: The UI System in the CoNLL-2012 Shared Task, pages 113–117.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica* 14:1396–1400.

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

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

Rezende Eraldo Fernandes, Nogueira Cícero dos Santos, and Luiz Ruy 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.

Iryna Haponchyk and Alessandro Moschitti. 2014. Making Latent SVM^{struct} practical for coreference resolution. In *Proceedings of the First Italian Conference on Computational Linguistics (CLiC-it 2014) & the Fourth International Workshop EVALITA 2014*. Pisa, Italy, pages 203–207.

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

K. Jonathan Kummerfeld, Taylor Berg-Kirkpatrick, and Dan Klein. 2015. An empirical analysis of optimization for max-margin nlp. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 273–279.

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.

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

Haoruo Peng, Kai-Wei Chang, and Dan Roth. 2015. A joint framework for coreference resolution and mention head detection. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 12–21.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. *Joint Conference on EMNLP and CoNLL - Shared Task*, Association for Computational Linguistics, chapter CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes, pages 1–40.

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.

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. ACM, New York, NY, USA, ICML '04, pages 104–.

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.

Alan Yuille and Anand Rangarajan. 2003. The concave-convex procedure (CCCP). *Neural Computation* 15:915–936.