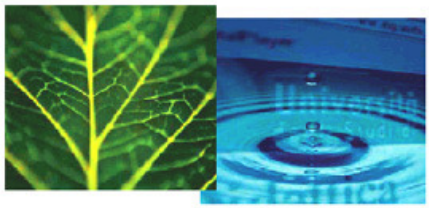
PhD Dissertation



International Doctorate School in Information and Communication Technologies

DISI - University of Trento

ADVANCED MODELS OF SUPERVISED STRUCTURAL CLUSTERING

Iryna Haponchyk

Advisor:

Prof. Alessandro Moschitti

Università degli Studi di Trento

April 2018

Abstract

The strength and power of structured prediction approaches in machine learning originates from a proper recognition and exploitation of inherent structural dependencies within complex objects, which structural models are trained to output. Among the complex tasks that benefited from structured prediction approaches, clustering is of a special interest. Structured output models based on representing clusters by latent graph structures made the task of supervised clustering tractable. While in practice these models proved effective in solving the complex NLP task of coreference resolution, in this thesis, we aim at exploring their capacity to be extended to other tasks and domains, as well as the methods for performing such adaptation and for improvement in general, which, as a result, go beyond clustering and are commonly applicable in structured prediction.

Studying the extensibility of the structural approaches for supervised clustering, we apply them to two different domains in two different ways. First, in the networking domain, we do clustering of network traffic by adapting the model, taking into account the continuity of incoming data. Our experiments demonstrate that the structural clustering approach is not only effective in such a scenario, but also, if changing the perspective, provides a novel potentially useful tool for detecting anomalies. The other part of our work is dedicated to assessing the amenability of the structural clustering model to joint learning with another structural model, for ranking. Our preliminary analysis in the context of the task of answer-passage reranking in question answering reveals a potential benefit of incorporating auxiliary clustering structures.

Due to the intrinsic complexity of the clustering task and, respectively, its evaluation scenarios, it gave us grounds for studying the possibility and the effect from optimizing task-specific complex measures in structured prediction algorithms. It is common for structured prediction approaches to optimize surrogate loss functions, rather than the actual task-specific ones, in order to facilitate inference and preserve efficiency. In this thesis, we, first, study when surrogate losses are sufficient and, second, make a step towards enabling direct optimization of complex structural loss functions. We propose to learn an approximation of a complex loss by a regressor from data. We formulate a general structural framework for learning with a learned loss, which, applied to a particular case of a clustering problem – coreference resolution, i) enables the optimization of a coreference metric, by itself, having high computational complexity, and ii) delivers an improvement over the standard structural models optimizing simple surrogate objectives. We foresee this idea being helpful in many structured prediction applications, also as a means of adaptation to specific evaluation scenarios, and especially when a good loss ap-

proximation is found by a regressor from an induced feature space allowing good factorization over the underlying structure.

Thesis committee

Roberto Basili
University of Rome, Tor Vergata

Andrea Passerini
University of Trento

Ivan Titov
University of Edinbugh

Keywords: structured prediction, structured output methods, supervised clustering, learned loss

Acknowledgements

I was driven by the thirst for interest, when I submitted myself to the PhD program. And the interest, in the broad sense, is the only of my expectations for how this path would be that come true. All the way, something new and interesting appeared to challenge and entertain me. In the rest, I could hardly realize what was awaiting ahead. It turns out, as I write it now, that the system of my expectations, in a sense, was inconsistent, since the fulfilment of the rest of the expectations would have canceled the interest one. By merely knowing the future, you have already lived it, thus no interest. There is something paradoxical about the interest itself here: knowing you will have it does not cancel it at the moment of realization, even if it is to be realized in making predictions as we do in machine learning. Before I get lost in the intricacies of the phenomenon of interest, let me thank those who contributed greatly to its realization during these years.

First of all, I am enormously grateful to my advisor Alessandro Moschitti, for his professional guidance, countless wise and valuable suggestions for doing research, and his help and support. He willingly shares his rich expertize, inspires, starting from the first day, when he revealed to me what startling things math can do for machine learning (at that time, I was new to machine learning), and above all, he always sees and indicates the positive side of the things, which I appreciate very much. I thank my colleagues Olga Uryupina, for her precious advice on coreference resolution and, in general, her encouraging optimism, and Kateryna Tymoshenko, for making possible our work on the ranking task and her willingness to help and share her knowledge, and both, for helping us with the data for the experiments. I thank all the members of our iKernels research group, current and former. It is a great pleasure for me to work side by side with you, and to witness your enthusiasm, ingenuity, and generosity.

The ICT Doctoral School, the DISI department, and the University of Trento, in general, have created a dynamic friendly environment providing a broad range of facilities for studying and doing research. I do not think it is possible to even imagine how many people and how much they have done for its creation. I am very lucky to have a chance to work here, for which I am thankful to all of them. I am particularly grateful to Andrea Passerini for his remarkable lectures on machine learning, among many other dedicated teachers. I thank Katsiaryna Labunets who inroduced me to the school in the beginning of 2012, when everything started.

I would like to thank the members of the thesis defence committee for their keen interest, deep questions, and valuable suggestions. In spite of my anxiety, I enjoyed our discussion, which turned out to be very inspiring. I will keep the memories of this very special day for a long time.

I remember very well the lecture of the course on algorithms by Pavel Skums at the Faculty of Mechanics and Mathematics at the Belarusian State University, where I pursued my degree

in mathematics, when he taught us the algorithms on graphs. I listened in amazement to him showing us Kruskal's spanning algorithm, nothing I knew at that moment of how closely I was going to deal with it. I thank our faculty for giving me that which I can hardly overestimate.

The writing of this manuscript was accompanied by the great music from the already non-existing Finnish band Charon, which I found as I started. I was looking for a service which would suggest me some artists similar to my favourites and the one I found was ultimately precise. I believe it employs a graph-based technology which might be related to the one we study in our work.

During the years of PhD, I met a lot of wonderful people, those who helped me, especially in the very beginning, those with whom we shared a lot of good moments, and those who showed me the real beauty in Man. Friends, you have made this time a treasure. I put this long acknowledgement list in the easiest straightforward way by the following clusters: Belarus, China, Georgia, Iran, Italy, Macedonia, Russia, Spain, Togo, and Ukraine.

Finally, I owe it all to my big family. I know some of you were times more eager than myself for this to happen. Thank you for your constant support all the way. I dedicate this work to my parents. I wish there existed a means to express entirely how much I am grateful to my wonderful mother, for all her care and love, and the freedom she is giving to me. And, I thank immensely that in whose eyes I had often seen that unspeakable, silent love – my dear father. I thank you for our early "mathematical disputes", starting probably from when I was nine-ten years old. I remember you suggesting me to solve a problem using equations, when I had not even known what an equation is. You are always in my heart.

Iryna Haponchyk

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Things are as they are. Looking out into the universe at night, we make no comparisons between right and wrong stars, nor between well and badly arranged constellations.

Alan Watts

1.1 Motivation

The rise of structured prediction methods in machine learning gave way to the development of effective solutions for the complex prediction tasks. In learning scenarios with a presumed or observed dependence between the output variables, structured prediction takes advantage of (imposing and) treating a structure over multiple dependent output variables as a unified output object of learning. It is opposed to solving a bunch of elementary prediction tasks and uniting their independently made predictions by diverse aggregation techniques to impart the underlying structure to the desired output. Passing on to learning structural dependencies solely, in most cases, results in an improvement over unstructured counterpart approaches whereas, for the latter, enriching the input representations seems to be nearly the only way of enhancement.

Powerful structured prediction methods were devised for such standard complex learning tasks as multilabel classification, sequence labelling, ranking, etc. On the discriminative side, generalizations to structured output of all the classic learning algorithms were derived: perceptron by Collins [2002], SVMs by Altun et al. [2003]; Tsochantaridis et al. [2004], online passive-aggressive learning by Crammer et al. [2006]. Discriminant functions f of these structured prediction approaches build on capturing the structural dependencies within the complex output space of structured objects Y. A prediction by the model is done finding a maximizer of the discriminant function f, a structure $\mathbf{y}$, over Y. It is desirable that exploring Y in search for the optimal $\mathbf{y}$ is done by an efficient inference procedure.

1.1.1 Supervised clustering as structured prediction

In this respect, clustering, if viewed as a structured prediction problem, lacks an efficient inference procedure, e.g., k-means clustering is NP-hard [Aloise et al., 2009], correlation clustering is NP-complete [Bansal et al., 2004]. However, structured prediction approaches targeting approximations of clustering inference, e.g., SVM$^{\text{cluster}}$ by Finley and Joachims [2005], supervised k-means by Finley and Joachims [2008], realized tools for doing supervised clustering. Later, Yu and Joachims [2009] proposed a more feasible solution to the supervised clustering formulation enabling exact inference. The trick consisted in transferring learning from the space of clusterings Y to the latent space of spanning graphs H, where inference is efficiently done by a spanning algorithm. Employed in the Latent Structural SVM, the approach of Yu and Joachims found its application in the NLP task of coreference resolution, having intrinsically a clustering nature. A similar approach, again applied to coreference resolution, although with alternative graph structures and a different spanning algorithm wrapped up in the Latent Structured Perceptron learner, by Fernandes et al. [2012, 2014] established the state of the art for its time. Regarding and inferring a clustering $\mathbf{y}$ of some set of elements as a unique structural object appears beneficial in contrast to deciding on the similarity for pairs of elements in detachment from the rest elements of the set.

Clustering is of a general interest and purpose, both as a standalone solution and as a provider of intermediate helpful information. Inspired by the effectiveness of the structured prediction approaches to supervised clustering in coreference resolution, one might think about their extensions to other tasks and domains. Even more promising is the fact that these approaches do not require neither searching for and supplying to the algorithm of an appropriate distance metric nor the beforehand indication of the number of the output clusters, the latter being a thorny issue of unsupervised and semi-supervised clustering approaches.

1.1.2 Structured prediction: evaluation v.s. optimization

When it turns to applying an approach of machine learning in general and structured output in particular, to a new domain, we inevitably have to adapt to a certain evaluation scenario. Optimization of the empirical risk expressed in terms of a domain- or task-specific evaluation measure is not always possible as the corresponding loss functions do not possess properties sufficient to admit a tractable solution. At the same time, task metric optimization via parametrization is known to have limited potential. Nevertheless, in structured output learning, the expressivity of loss functions is often sacrificed for preserving the feasibility of solutions, e.g., losses with the factorization properties conforming to the factorization of the learned model are preferred for facilitating inference.

In case of clustering, it is very unlikely to bypass the above compromise as clustering eval-

uation metrics are far from being trivial, e.g., clustering accuracy [Zhao and Karypis, 2002], purity and entropy [Fung et al., 2003], just to mention a few of most common ones. In the meantime, the approaches of Yu and Joachims and Fernandes et al. use simple loss functions, counting the number of mistaken edges, in order to perform the max-violating inference using the same spanning graph algorithms. A clustering measure adopted in coreference resolution research – Mention, Entity, and Link Average (MELA) [Pradhan et al., 2012] – has even more superior complexity, and its computation cannot be afforded during training.

1.2 Thesis Work and Structure

In this thesis, we seek to explore the applicability and extensibility of the structured output approaches for supervised clustering. We start by giving a general overview of structured output methods and structured prediction solutions to supervised clustering, in Chapter 2. We revisit the essential components of the structured prediction framework and provide formulations of the algorithms employed throughout this work. Specifically, the structured output approaches with latent variables supplying algorithmic machinery for supervised clustering.

We found supervised clustering useful in the networking domain. In particular, we enable an extension of the graph-based structural model for clustering of the network traffic that provides a viable solution to the intrusion detection task. We address the task of intrusion or, more broadly, anomaly detection framing it as a clustering problem, where new coming instances which do not fall into existing clusters are regarded anomalous. We elaborate the structured prediction approach for supervised clustering to make it operate on the continuous flow of data. This results in an effective anomaly detection method not requiring re-training, which is described in Chapter 4.

The intricacies, highlighted in Section 1.1.2, around optimization of complex metrics arising in applications of structured prediction, inclined us towards research on making structured prediction more 'complex metric' -friendly. We propose to address this kind of problems by approximating a complex measure by learning its representation, or alternatively a representation of the loss function induced by it, from data. Here, we introduce a notion of a *learned loss*. In Chapter 3, we define a general structured output framework for learning with a learned loss, which bases on the idea of approximating a task-specific complex loss with a regressor. This is where MELA enters into play providing an interesting special case of a complex measure, on which we test the whole idea of learning a complex structural loss. The framework is also extended with a specification of the algorithm that performs joint learning of a loss and a model.

First, in Chapter 5, we carry out a practical study comparing structured prediction approaches for supervised clustering applied to the task of coreference resolution: i) online versus batch learning, ii) employing different graph structures and inference algorithms. Subsequently,

we implement the idea of a learned loss for the MELA case on the basis of the best identified structured approach. This way, we enable a direct optimization of an approximation of a complex clustering measure.

Finally, our recent study on using structural clustering as an auxiliary means in the context of another structured prediction task of ranking, as a strategy to boost the accuracy of the latter. It bases on a simple intuition, that in presence of similar items in the ranking list, both relevant and irrelevant, clustering can communicate to a ranking model that certain items are likely to take neighbouring rank positions. Thus, we attempt at combining a structural ranking model with the clustering graph $\mathbf{h}$, providing a procedure for joint inference, and show our preliminary results of such a model "collaboration" in Chapter 6.

1.3 Thesis Contributions

The contributions of the thesis can be summarized as follows:

- Definition of a general structured prediction framework for learning with a learned loss. Joint formulation for learning the loss and the task simultaneously.
- Comparison of structured prediction approaches for supervised clustering applied to the task of coreference resolution.
- Enabling a direct optimization of MELA, a complex clustering measure used in coreference resolution for evaluating the system output, which was previously intractable.
- Anomaly detection model based on supervised clustering.
- Structured prediction model for re-ranking, combined with clustering.

1.4 Publications

Research presented in this thesis resulted into the following publications:

- *Jointly learning models and loss functions from coreference resolution.* Iryna Haponchyk and Alessandro Moschitti. Under submission, 2018.
- *Real-time intrusion detection via structured graph-based learning.* Iryna Haponchyk and Alessandro Moschitti. To be submitted.
- *A Latent Structured Prediction Approach for Passage Re-ranking.* Iryna Haponchyk and Alessandro Moschitti. To be submitted.

- *Don't understand a measure? Learn it: Structured Prediction for Coreference Resolution optimizing its measures.* Iryna Haponchyk and Alessandro Moschitti. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1018-1028, Vancouver, Canada, 2017.
- *A Practical Perspective on Latent Structured Prediction for Coreference Resolution.* Iryna Haponchyk and Alessandro Moschitti. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017), pp. 143–149, Valencia, Spain, 2017.
- *Making Latent SVM^{struct} Practical for Coreference Resolution.* Iryna Haponchyk and Alessandro Moschitti. In Proceedings of the First Italian Conference on Computational Linguistics (CLiC-it 2014), pp. 203–207, Pisa, Italy, 2014.

Chapter 2

Backgrounds on Structured Output Methods and Supervised Clustering

2.1 Structured Output Methods

There is a growing body of approaches in machine learning that are devised to make inference in complex output spaces. Unlike classification, where the output space is $Y = \{1, 2, ..., K\}$, the output space of structured prediction algorithms consists of complex or structured objects like sequences, trees or graphs. In classification, a predictor learns a mapping $f : X \to Y$ from training examples. A structured predictor learns a scoring function $f : X \times Y \to \mathbb{R}$ and makes a prediction by finding

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in Y}{argmax} f(\mathbf{x}, \mathbf{y}). \tag{2.1}$$

Often, as well as throughout this work, function $f(\mathbf{x}, \mathbf{y})$ is assumed linear in a *joint feature vector* $\Phi(\mathbf{x}, \mathbf{y})$ of an input-output example:

$$f(\mathbf{x}, \mathbf{y}) = \mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y}), \tag{2.2}$$

where the form of Φ depends on a specificity of a problem, and $\mathbf{w}$ is a learned weight vector. This modelling allows efficient use of combined features of inputs and outputs.

2.1.1 Algorithms

Structured perceptron [Collins, 2002] is a structural version of the standard perceptron algorithm for classification, providing an online algorithm with established convergence bounds for learning to predict structures. Among large-margin approaches, there are *Structural SVMs* – a generalization of multiclass SVMs to the case of tasks with structured output [Tsochantaridis et al., 2004].

2.1.2 Loss functions

Since incorrect structures $\hat{\mathbf{y}}$ vary in their degree of correctness against the correct (gold) structure $\mathbf{y}$, large-margin structured prediction approaches involve *loss functions*:

$$\Delta(\mathbf{y}, \hat{\mathbf{y}}) : Y \times Y \rightarrow \mathbb{R},$$

which reflect such degrees of correctness. To minimize the risk expressed in terms of a loss $\Delta(\mathbf{y}, \hat{\mathbf{y}})$, Structural SVMs pass to optimizing a piecewise convex upper-bound on the loss

$$\Delta(\mathbf{y}, \hat{\mathbf{y}}) \leq \underset{\hat{\mathbf{y}} \in Y}{max} \left[\Delta(\mathbf{y}, \hat{\mathbf{y}}) + \mathbf{w} \cdot \Phi(\mathbf{x}, \hat{\mathbf{y}})\right] - \mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y}),$$

and solve the following large-margin objective:

$$\min_{\mathbf{w}} \left[\frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{n} \left[\max_{\hat{\mathbf{y}} \in Y} \left[\Delta(\mathbf{y}_i, \hat{\mathbf{y}}) + \mathbf{w} \cdot \Phi(\mathbf{x}_i, \hat{\mathbf{y}})\right] - \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i)\right]\right] \tag{2.3}$$

on training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$ [Tsochantaridis et al., 2004].

2.1.3 Inference

Finding $argmax$ in Equation 2.1 through exhaustive search over the output space Y is not always tractable. For this reason, exploration techniques, efficient, preferably exact but also heuristic, are adopted in different applications, e.g., search for an optimum sequence in sequence tagging could be done by a dynamic programming Viterbi-like decoding algorithm [Altun et al., 2003].

Max-violating inference and loss factorization

The objective in Equation 2.3 involves finding the *max-violating* structure:

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in Y}{argmax} [\Delta(\mathbf{y}_i, \mathbf{y}) + \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y})]. \tag{2.4}$$

For efficient max-violating inference of $\hat{\mathbf{y}}$, it is indispensable that the loss Δ and the score factorize in the same way over the substructures (components) of $\hat{\mathbf{y}}$.

There is a large-margin version of the structured perceptron which adopts inference in Equation 2.4, also called *loss-augmented* inference [Fernandes and Brefeld, 2011]. Another online large-margin algorithm is a structured passive-aggressive (PA) algorithm by Crammer et al. [2006].

2.1.4 Latent variables

Auxiliary information, which is not available in the input training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ but which can alleviate infeasible inference or help training in general, can be included as *latent variables* $\mathbf{h}$ [Yu and Joachims, 2009]. The joint feature vector is extended with $\mathbf{h}$ to $\Phi(\mathbf{x}, \mathbf{y}, \mathbf{h})$. Correspondingly, the inference – to

$$(\hat{\mathbf{y}}, \hat{\mathbf{h}}) = \underset{(\mathbf{y},\mathbf{h}) \in Y \times H}{argmax} \ \mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y}, \mathbf{h}). \tag{2.5}$$

Latent Structural SVM

Latent Structural SVM (Latent SVM$^{\text{struct}}$, or LSSVM for short) proposed by Yu and Joachims [2009] introduces latent structures in SVM$^{\text{struct}}$ [Tsochantaridis et al., 2004]. In presence of auxiliary latent structures, the objective function of SVM$^{\text{struct}}$ in Equation 2.3, optimizing an upper bound on the loss function, becomes no longer valid. To solve this problem Yu and Joachims assume that loss functions usually do not depend on the gold latent structures $\mathbf{h}_i$ in correspondence to the gold outputs $\mathbf{y}_i$, and can be computed against the gold $\mathbf{y}_i$ only:

$$\Delta(\mathbf{y}_i, \mathbf{h}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}}) = \Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}}),$$

For such cases, they derived the following LSSVM objective:

$$\min_{\mathbf{w}} \big[\frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{n} \max_{(\hat{\mathbf{y}},\hat{\mathbf{h}}) \in Y \times H} [\Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}}) + \mathbf{w} \cdot \Phi(\mathbf{x}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})] - C \sum_{i=1}^{n} \max_{\mathbf{h} \in H} \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h})\big]. \tag{2.6}$$

Being essentially a difference of two convex functions, it is solved using the Concave-Convex Procedure (CCCP) by Yuille and Rangarajan [2003], which is guaranteed to converge to a local minimum or a saddle point. This alternatively switches between

1. finding an upper bound on the concave part of the objective in Equation 2.6: $g(\mathbf{w}) = -C \sum_{i=1}^{n} \max_{\mathbf{h} \in H} \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h})$, which includes finding the best latent structures $\mathbf{h}_i^*$ explaining gold truths $\mathbf{y}_i$ with respect to the current model weights $\mathbf{w}$:

$$\mathbf{h}_i^* = \underset{\mathbf{h} \in H}{\text{argmax}} \, \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}), \text{and}$$

2. solving a standard SVM$^{\text{struct}}$ problem:

$$\min_{\mathbf{w}} \Big[\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \big[\max_{(\hat{\mathbf{y}},\hat{\mathbf{h}}) \in Y \times H} [\Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}}) + \mathbf{w} \cdot \Phi(\mathbf{x}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})] - \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*)\big]\Big] \tag{2.7}$$

by a one-slack cutting-plane Algorithm 1.

The cost of one CCCP iteration is nearly a cost of one SVM$^{\text{struct}}$, which in turn is polynomial [Tsochantaridis et al., 2004].

Algorithm 1 Latent Structural SVM (LSSVM)

1: Input: $X = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, $\mathbf{w}_0$, ϵ
2: $\mathbf{w} \leftarrow \mathbf{w}_0$; $S \leftarrow \emptyset$
3: **repeat**
4: **for** $i = 1, ..., n$ **do**
5: $H_i(\mathbf{y}, \mathbf{h}) \equiv \Delta(\mathbf{y}_i, \mathbf{y}, \mathbf{h}) +$ $\mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y}, \mathbf{h}) - \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*)$
6: compute $(\hat{\mathbf{y}}_i, \hat{\mathbf{h}}_i) = \underset{(\mathbf{y}, \mathbf{h}) \in Y \times H}{\operatorname{argmax}} H_i(\mathbf{y}, \mathbf{h})$
7: **end for**
8: compute $\xi = \max\{0, \underset{\{(\mathbf{y}_i', \mathbf{h}_i')\}_{i=1}^n \in S}{\max} \sum_{i=1}^n H_i(\mathbf{y}_i', \mathbf{h}_i')\}$
9:
10: **if** $\sum_{i=1}^n H_i(\hat{\mathbf{y}}_i, \hat{\mathbf{h}}_i) > \xi + \epsilon$ **then**
11: $S \leftarrow S \cup \{(\hat{\mathbf{y}}_i, \hat{\mathbf{h}}_i)\}_{i=1}^n$
12: $\mathbf{w} \leftarrow$ optimize primal over S
13: **end if**
14: **until** S has changed during iteration
return $\mathbf{w}$

Algorithm 2 Latent Structured Perceptron (LSP)

1: Input: $X = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, $\mathbf{w}$, C, T
2: $\mathbf{w}_0 \leftarrow \mathbf{w}$; $t \leftarrow 0$
3: **repeat**
4: **for** $i = 1, ..., n$ **do**
5: $\hat{\mathbf{h}} \leftarrow \underset{\mathbf{h} \in H(\mathbf{x}_i)}{\operatorname{argmax}} \mathbf{w}_t \cdot \Phi(\mathbf{x}_i, \mathbf{y}, \mathbf{h}) + C \cdot \Delta(\mathbf{y}_i, \mathbf{y}, \mathbf{h})$
6: **if** $\hat{\mathbf{y}} \not\equiv \mathbf{y}_i$ **then**
7: $\mathbf{h}^* \leftarrow \underset{\mathbf{h} \in H(\mathbf{x}_i, \mathbf{y}_i)}{\operatorname{argmax}} \mathbf{w}_t \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h})$
8: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}^*) - \Phi(\mathbf{x}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})$
9: **end if**
10: $t \leftarrow t + 1$
11: **end for**
12: **until** $t < nT$
13: $\mathbf{w} \leftarrow \frac{1}{t} \sum_{i=1}^t \mathbf{w}_i$
return $\mathbf{w}$

Latent Structured perceptron

Latent Structured Perceptron (LSP) is a version of the structured perceptron with latent variables [Sun et al., 2009]. Algoritm 2 presents the LSP algorithm with loss-augmented inference [Fernandes et al., 2014].

Given a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, initialized model weights $\mathbf{w}_0$ (e.g., to $\vec{0}$ or a random vector), a loss parameter C and the maximum number of epochs T, LSP iterates the following instructions:

- Line 5 seeks for the max-violating latent variable $\hat{\mathbf{h}}$ in $H(\mathbf{x}_i)$, which is the set of all possible latent variables for the current example. Further, we denote by $\hat{\mathbf{y}}$ the output structure corresponding to the max-violating $\hat{\mathbf{h}}$.
- Line 6 tests if the output $\hat{\mathbf{y}}$, induced by the produced max-violating $\hat{\mathbf{h}}$, is incorect.
- Line 7 finds a latent variable $\mathbf{h}^*$ that maximizes the model score $\mathbf{w}_t \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h})$, for the example $(\mathbf{x}_i, \mathbf{y}_i)$. This finds the maximum latent variable corresponding to the ground truth structure $\mathbf{y}_i$ with respect to the current $\mathbf{w}_t$. Finding such max requires an exploration over the set $H(\mathbf{x}_i, \mathbf{y}_i)$, composed of only $\mathbf{h}$ consistent with gold structured label $\mathbf{y}_i$.

- If the test is verified, the model is updated with the vector $\Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}^*) - \Phi(\mathbf{x}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})$.

Online passive-agressive structured output learning

In this work, we will also employ a structured passive-aggressive (PA) algorithm [Crammer et al., 2006] which we extend with latent variables. Following Crammer et al., we perform perceptron-like updates:

$$\mathbf{w} = \min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2,$$

s.t. $\mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*) - \mathbf{w} \cdot \Phi(\mathbf{x}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}}) \geq \Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})$,
where

$$(\hat{\mathbf{y}}, \hat{\mathbf{h}}) = \operatorname*{argmax}_{(\mathbf{y},\mathbf{h}) \in Y \times H} \left[\mathbf{w}_t \cdot \Phi(\mathbf{x}_i, \mathbf{y}, \mathbf{h}) - \mathbf{w}_t \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*) + \Delta(\mathbf{y}_i, \mathbf{y}, \mathbf{h})\right],$$

which in a closed form is

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t(\Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*) - \Phi(\mathbf{x}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})),$$

where $\tau_t = \frac{\mathbf{w}_t \cdot \Phi(\mathbf{x}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}}) - \mathbf{w}_t \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*) + \Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})}{\|\Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*) - \Phi(\mathbf{x}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})\|^2}$.

We refer to this model as LSPA.

2.2 Structured Prediction for Supervised Clustering

2.2.1 Supervised clustering

In this work, we consider the following formulation of the supervised clustering task [Finley and Joachims, 2005]:

Given training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, where

- each $\mathbf{x}_i$ is a set of elements $\mathbf{x}_i = \{x_1, x_2, ..., x_{N_i}\}$ of arbitrary nature (we can assume x_j belong to some set S),
- $\mathbf{y}_i$ is the corresponding gold output clustering of the elements of $\mathbf{x}_i$,

learn to predict clusterings $\mathbf{y}$ of unseen input sets $\mathbf{x} \in 2^S$. Thus, the task is to learn a predictor $h : X \to Y$, where $X = 2^S$ is a set of all possible sets of items from S and Y is a set of all possible clusterings of such sets, $Y = 2^{2^S}$.

Note that the above formulation is different from the view on the task adopted in data mining [Eick et al., 2004], where the input data points x_i are supervised with class membership labels c ($c \in C$, $C = \{c_i\}_{i=1}^N$) and the output clustering is assessed with respect to class purity/impurity of the produced clusters, the number of output clusters being adjusted for the

specific application needs. In contrast, the aim pursued here is to reproduce entirely the intended clustering of the input set, whereas the beforehand indication of the number of clusters k is not required.

2.2.2 Structured output clustering approaches

SVM$^{\text{cluster}}$

Finley and Joachims [2005] formulate the clustering task as a structured prediction. The input-output pair $(\mathbf{x}, \mathbf{y})$ is represented with a following joint feature vector

$$\Phi(\mathbf{x}, \mathbf{y}) = \frac{1}{|\mathbf{x}|^2} \sum_{y \in \mathbf{y}} \sum_{x_i, x_j \in y} \phi(x_i, x_j), \tag{2.8}$$

which sums up feature vectors $\phi(x_i, x_j)$ over all item pairs (x_i, x_j), where x_i and x_j fall into the same clusters y according to $\mathbf{y}$. SVM$^{\text{struct}}$ solver learnes a linear model $\mathbf{w}$, which scores clusterings with $\mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y})$ (Equation 2.2). Finding a clustering $\mathbf{y}$ maximizing such a score, i.e. maximizing the intracluster pairwise similarity scores, is performed by correlation clustering [Bansal et al., 2004]. Inference by correlation clustering is approximate as finding a real maximizer is NP-complete.

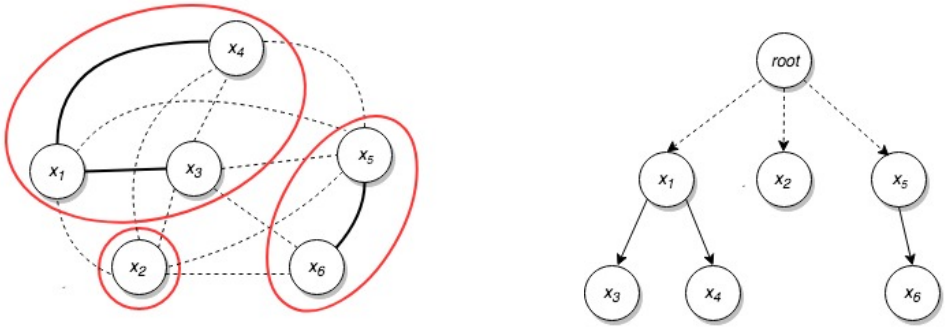
SVM$^{\text{cluster}}$ optimizes the clustering loss functions Δ: i) pairwise loss – the percentage of wrong pairwise decisions ii) F1 loss of the coreference resolution MUC metric [Vilain et al., 1995]. Both the losses are *global* in a sense that they compare entirely the output clustering $\mathbf{y}$ to the gold standard $\mathbf{y}_i$. A greedy procedure and a relaxation of correlation clustering are proposed for the max-violating inference with respect to the mentioned losses.

Supervised k-means

Supervised k-means approach by Finley and Joachims [2008] builds on a similar modelling of SVM$^{\text{cluster}}$. Using the structural SVM formulation, it learns a similarity measure in a structural way but uses k-means as the inference method. Here, however, the number of clusters k should be provided by the user.

Graph-based learning

The above two approaches train a structural predictor optimizing the aggregated pairwise similarities via approximations of clustering methods (correlation clustering, k-means). The intractability of finding an optimal clustering imposes limitations on the accuracy of the inference and scalability with respect to the size of the input sets $\mathbf{x}$.



(a) Latent graph with highlighted clusters (b) Latent tree

Figure 2.1: Graph representations used in structured prediction clustering algorithms

LSSVM$^{\mathrm{K}}$ A workaround proposed by Yu and Joachims [2009] suggests to switch from the output space of clusterings Y to the space of graph structures H. Namely, instead of searching for the optimal clustering $\mathbf{y}$ the search is performed for the spanning forest $\mathbf{h}$ on an undirected graph G, whose nodes are elements x_i of the input $\mathbf{x}$ and edges are all the pairwise links between them (x_i, x_j). In Figure 2.1a, the spanning forest $\mathbf{h}$ composed of the solid edges corresponds to clustering $\mathbf{y}$ highlighted in red. Note, that there is no one to one correspondence between $\mathbf{h}$'s and $\mathbf{y}$'s. Using two other possible alternative ways to span the biggest cluster of 3 in Figure 2.1a give the same $\mathbf{y}$. However, each $\mathbf{h}$ uniquely induces a clustering $\mathbf{y}$.

This way, Yu and Joachims base their approach on the strong pairwise links (those spanning each cluster component, the rest can be inferred), in contrast to the previous approaches taking into account all the pairwise intracluster connections. The spanning forest structures $\mathbf{h}$ are incorporated as latent variables into the structured prediction framework and the joint feature representation decomposes over the edges of $\mathbf{h}$:

$$\Phi(\mathbf{x}, \mathbf{y}, \mathbf{h}) = \sum_{\mathbf{e}=(x_i,x_j)\in\mathbf{h}} \phi(\mathbf{e}) = \sum_{\mathbf{e}=(x_i,x_j)\in\mathbf{h}} \phi(x_i, x_j). \tag{2.9}$$

This facilitates the inference (Equation 2.5) which now can be performed exactly and which reduces to finding a spanning algorithm on the fully connected graph G. Kruskal's spanning algorithm [Kruskal, 1956] is used for inferring $\mathbf{h}$. It involves sorting of all the candidate edges by weight, followed by the construction of a spanning forest, starting from the edge with the highest weight. The threshold on the edge weight to be included into the forest is naturally 0, as including more or less edges decreases the total weight of the spanning forest $\mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y}, \mathbf{h})$.

The authors train the LSSVM algorithm. We will further denote this approach as LSSVM$^{\mathrm{K}}$.

LSP$^{\mathrm{E}}$ An approach similar to LSSVM$^{\mathrm{K}}$, appeared in works of Fernandes et al. [2012, 2014] in application to the NLP task of coreference resolution, adopts another kind of graph structures – directed trees with an additional root node (see Figure 2.1b). In this case, the inference is done on a graph G formed by all the directed arcs (x_i, x_j), where $i < j$, plus directed arcs from the root node to each x_i – $(root, x_i)$. Imposing the node order $root, x_1, x_2, ..., x_n$ and the directionality of links imply the absense of cycles in G. The inference by Edmonds' spanning tree algorithm [Chu and Liu, 1965; Edmonds, 1967] reduces to traversing the nodes $x_1, x_2, ..., x_n$ in their predefined order and choosing for each an incoming arc of the highest weight, which is added to the tree $\mathbf{h}$. In the resulting spanning tree $\mathbf{h}$, the subtrees connected directly to the root node form clusters. In a sense, the inference procedure can be viewed as a iterative process of adding nodes to the tree selecting the best subtree (cluster) for a node to be added to based on the score of the connecting arc. The joint feature vector of the input-output pair decomposes over the arcs of the spanning tree $\mathbf{h}$ in the same way as in Equation 2.9. Fernandes et al. use the LSP algorithm for training to predict such structures. Let us refer to this model as LSP$^{\mathrm{E}}$.

Chapter 3

Genearal Framework for Learning with a Learned Structural Loss

The final goal of ML models is to optimize a task-specific measure. In non-strucutured prediction, there is already a range of methods which optimize measures encompassing the outputs for several (all) training examples, i.e., multivariate measures [Joachims, 2005]. In structured prediction, the importance of reference to the measure is even more amplified. Already at the example level, we have different degrees of correctness of the output structure $\mathbf{y}$ with respect to the task measure μ. It is crucial to communicate to the learner a proper information about such differentiation for it to discriminate correspondingly.

In many cases, direct optimization of a task measure is associated with high complexities. Thus, the models often base on optimizing alternative surrogate objectives. These, for instance, include loss functions which are upper bounds on the real loss Δ associated with the task measure μ. Such loss upper bounds often possess good factorization properties. Or just a simple surrogate loss, e.g., counting mistakes on the subparts of the output structure $\mathbf{y}$.

In this chapter, we abstract from the application specifics and give a general formulation of the framework for optimizing a task-specific structural loss. It is based on learning an approximation of the task loss Δ (or alternatively, of the measure μ) and optimizing such a learned objective within the structured prediction framework. We provide also a joint formulation destined for training the loss model and the general task learner simultaneously.

We, first, explore the conditions when simple surrogate losses are effective in structured prediction.

3.1 On Optimality of Simple Loss Functions

A most common class of surrogate loss functions optimized by the structured prediction algorithms are those having good factorization properties. Suppose the structured output variable $\mathbf{y}$ or the latent structure $\mathbf{h}$, in the latent case, decompose into subparts (substructures). Let us stick with the latent case. If $\mathbf{h}$ decomposes into substructures: $\mathbf{h} = \bigoplus_{j=1}^{n} \mathbf{h}^j$, the straightforward loss can be defined by

$$\Delta_{simple}(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}}) = \sum_{j=1}^{n} \mathbb{1}_{[\hat{\mathbf{h}}^j \notin H^j(\mathbf{y})]}, \tag{3.1}$$

where $\mathbb{1}_{[\cdot]}$ is an indicator function. Δ_{simple} counts the substructure mistakes, i.e., the substructures $\hat{\mathbf{h}}^j$ which do not comply with the gold standard $\mathbf{y}$. Or more generally,

$$\Delta_{simple}(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}}) = \sum_{j=1}^{n} l(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}}^j), \tag{3.2}$$

where $l(\cdot)$ is a substructure loss function. Equation 3.2 allows for assigning different penalties for different mistake types, e.g., through parametrization.

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

Proof. If the data is linearly separable the perceptron converges $\Rightarrow \Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}}) = 0, \forall \mathbf{x}_i$. From Equation 3.2, it follows that $\sum_{j=1}^{n} l(\mathbf{y}_i, \hat{\mathbf{y}}_i, \hat{\mathbf{h}}^j) = 0$. The latter equation and monotonicity imply $l(\mathbf{y}_i, \hat{\mathbf{y}}_i, \hat{\mathbf{h}}^j) = 0, \forall i = \overline{1, n}$, i.e., there are no substructure mistakes, otherwise by fixing such, we would have a smaller Δ, i.e., negative, contradicting the initial positiveness hypothesis. Thus, no substructure mistake in any $\mathbf{x}_i$ implies that $\mu(\mathbf{y}, \hat{\mathbf{h}})$ is maximized on the training set. □

Proposition 1 implies that, for separating the data separable with respect to the target task-specific measure, it is sufficient to optimize a simple factorizable loss whose 0 mistakes maximize the measure and which meets the requirement of delivering monotonicity to the measure. From the proof, it is easy to notice that the conclusion of Proposition 1 holds also under a weaker condition than global monotonicity. It is sufficient that Δ is locally monotone by inclusion in substructure mistakes, i.e., $\forall \hat{\mathbf{h}}_1$, $\forall \hat{\mathbf{h}}_2$, such that $\forall j$, $\hat{\mathbf{h}}^j \in \hat{\mathbf{h}}_1 \setminus (\hat{\mathbf{h}}_1 \cap \hat{\mathbf{h}}_2)$ $\mathbb{1}_{[\hat{\mathbf{h}}^j \notin H^j(\mathbf{y})]} > 0$, $\Delta(\mathbf{y}, \hat{\mathbf{y}}_1, \hat{\mathbf{h}}_1) > \Delta(\mathbf{y}, \hat{\mathbf{y}}_2, \hat{\mathbf{h}}_2)$.

In inseparable cases, however, simple loss functions loose optimality, and the reached "separating" hyperplane might be no longer optimal with respect to the task measure.

3.2 Learning a Loss Function

Consider a more difficult setting, whether in terms of separability or when the specifics of the real task loss cannot be straightforwardly spanned onto the defined structure (onto substructure variables). What if such implicit loss specifics could be captured within some approximation by a learned model over substructure variables or their aggregations? In the simplest linear case, suppose we have learned a function $\tilde{\Delta} : \mathbb{R}^m \rightarrow \mathbb{R}$:

$$\tilde{\Delta}(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}}) = \mathbf{v} \cdot \Psi(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}}), \tag{3.3}$$

over some feature space induced by some mapping $\Psi : Y \times Y \times H \rightarrow \mathbb{R}^m$. $\tilde{\Delta}$ is such that it approximates the task-specific loss Δ

$$\tilde{\Delta}(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}}) \approx \Delta(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}}), \tag{3.4}$$

with sufficient accuracy.

We need data for training and validation of the loss model $\mathbf{v}$, i.e., examples of the form:

$$x^{\rho} = [\Delta(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}}), \Psi(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}})],$$

here, ρ stands for "regression". x^{ρ} is simply a tuple containing a target value, which is the actual value of the task-specific loss Δ on the output $(\hat{\mathbf{y}}, \hat{\mathbf{h}})$ for some example $(\mathbf{x}, \mathbf{y})$, and a feature vector Ψ describing such output against the gold standard $\mathbf{y}$. For generating a dataset $X^{\rho} = \{x^{\rho}\}$ of such examples, we propose to collect max-violating structured outputs $(\hat{\mathbf{y}}, \hat{\mathbf{h}})$ produced for training examples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$ during structural learning with a surrogate loss, e.g., LSP learning using Δ_{simple}. Generally, there is no restriction on the amount of structures we can generate, although it can be subject to how many of them we can afford to label with target values Δ.

In Section 5.3, we learn a linear regression model that approximates the complex coreference resolution measure and optimize it using LSP. Optimizing $\tilde{\Delta}$ by LSP (injecting in Line 5 of Algorithm 2) requires defining a customized procedure for finding the max-violating constraint, if $\tilde{\Delta}$ does not factorize or factorize differently than the model $\mathbf{w}$. Let us denote the instance of LSP optimizing the learned loss as *LSP*$_{\rho}$.

3.3 Joint Learning of a Model and a Loss Function

In this section, we define an algorithm for learning the loss function along with the model, where the training data for the former is generated during the training of the latter. At the same time, the model uses the loss function, which becomes more and more accurate throughout training. The definition of the joint algorithm is based on LSP, for the sake of simplicity, and since we apply it later in this form, however, it is of general purpose and not bound to LSP.

We propose two methods of a joint learning strategy. The first method trains $\tilde{\Delta}$ model in a batch style at each iteration of LSP$_\rho$ using training data, which become more and more available. And the second approach uses an online regressor to learn $\tilde{\Delta}$. This means that the updates of the LSP$_\rho$ and the loss regressor models are interleaved.

Algorithm 3 Latent Structured Perceptron with Joint $\tilde{\Delta}$ Learning (LSP$^*_\rho$)

Input: $X = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, $\mathbf{w}$, C, T
$\mathbf{w}_0 \leftarrow \mathbf{w}$; $t \leftarrow 0$; $X^\rho = \emptyset$; $\tilde{\Delta} \leftarrow \Delta_{simple}$
repeat
  for $i = 1, ..., n$ **do**
    $\hat{\mathbf{h}} \leftarrow \underset{\mathbf{h} \in H(\mathbf{x}_i)}{\operatorname{argmax}} \mathbf{w}_t \cdot \Phi(\mathbf{x}_i, \mathbf{y}, \mathbf{h}) + C \cdot \tilde{\Delta}(\mathbf{y}_i, \mathbf{y}, \mathbf{h})$
    $x^\rho = [\Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}}), \Psi(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})]$
    $X^\rho = X^\rho \cup x^\rho$
    if $\hat{\mathbf{y}} \not\equiv \mathbf{y}_i$ **then**
      $\mathbf{h}^* \leftarrow \underset{\mathbf{h} \in H(\mathbf{x}_i, \mathbf{y}_i)}{\operatorname{argmax}} \mathbf{w}_t \cdot \Phi(\mathbf{x}_i, \mathbf{y}, \mathbf{h})$
      $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}^*) - \Phi(\mathbf{x}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})$
    end if
    $t \leftarrow t + 1$
  end for
  Train $\tilde{\Delta}$ on X^ρ
until $t < nT$
$\mathbf{w} \leftarrow \frac{1}{t} \sum_{i=1}^{t} \mathbf{w}_i$
return $\mathbf{w}$

Algorithm 4 Latent Structured Perceptron with Joint Online $\tilde{\Delta}$ Learning (LSP$^{*_{online}}_\rho$)

Input: $X = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, $\mathbf{w}$, C, T
$\mathbf{w}_0 \leftarrow \mathbf{w}$; $\mathbf{v} \leftarrow \vec{0}$; $t \leftarrow 0$; $\tilde{\Delta} \leftarrow \Delta_{simple}$; $flag \leftarrow$ `true`
repeat
  $err = 0$
  for $i = 1, ..., n$ **do**
    $\hat{\mathbf{h}} \leftarrow \underset{\mathbf{h} \in H(\mathbf{x}_i)}{\operatorname{argmax}} \mathbf{w}_t \cdot \Phi(\mathbf{x}_i, \mathbf{y}, \mathbf{h}) + C \cdot \tilde{\Delta}(\mathbf{y}_i, \mathbf{y}, \mathbf{h})$
    if $\hat{\mathbf{y}} \not\equiv \mathbf{y}_i$ **then**
      $\mathbf{h}^* \leftarrow \underset{\mathbf{h} \in H(\mathbf{x}_i, \mathbf{y}_i)}{\operatorname{argmax}} \mathbf{w}_t \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h})$
      $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}^*) - \Phi(\mathbf{x}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})$
    end if
    $t \leftarrow t + 1$
    if $flag$ **then**
      $\nu = \Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}}) - \mathbf{v} \cdot \Psi(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})$
      $\mathbf{v} \leftarrow \mathbf{v} + \nu \Psi(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})$
      $err = err + |\nu|$
    end if
  end for
  if $err > prev\_err$ **then**
    $flag \leftarrow$ **false**
  else
    $prev\_err = err$
  end if
until $t < nT$
$\mathbf{w} \leftarrow \frac{1}{t} \sum_{i=1}^{t} \mathbf{w}_i$
return $\mathbf{w}$

3.3.1 Joint learning approach (LSP^*_ρ)

Our basic approach with an automatically learned loss involves (i) learning standard models, e.g., LSP, (ii) generating examples from such models, (iii) learning $\tilde{\Delta}$ on the examples, and then (iv) using the learned loss in LSP_ρ.

In order to reduce the computation time of the approach, we verify if it is possible to combine the above steps in one joint algorithm. For this purpose, we designed Algorithm 3, which adds the following instructions to Algorithm 2: first, the starting loss $\tilde{\Delta}$ is set to Δ_{simple} in Line 2. Second, Line 6 composes a regression example x^ρ, for training $\tilde{\Delta}$, from the current max-violating structure $\hat{\mathbf{h}}$ for the example $(\mathbf{x}_i, \mathbf{y}_i)$, which contributes to creating the training set X^ρ. Finally, in Line 14, $\tilde{\Delta}$ is trained on X^ρ, and is used for all the training documents at the next epoch. In our experiments, we used a linear SVM regression solver for training $\tilde{\Delta}$ model.

3.3.2 Online algorithm for learning of a model and a loss function

One advantage of joint learning is to provide the loss learner with examples specific to the current learning task. However, although it is very fast, the loss regressor has to be retrained on each iteration. Thus, to further decrease the computational complexity of the joint learning algorithm, we use simple perceptron updates instead of SVM for the learning $\tilde{\Delta}$. In this variant of our model depicted in Algorithm 4, $\mathbf{v}$, the model used to obtain $\tilde{\Delta}$, is gradually updated in Lines 13–15 on each processed example, until the overall epoch regression error no longer decreases.

As will be seen from the experiments, the joint models show that it is enough to utilize the examples from the general LSP_ρ training. It is not needed to generate them separately on preprocessing as it is done for LSP_ρ. This even appears favourable as the examples come from the regions of the search space of the general learner.

Chapter 4

Structural Clustering in Network Environments

Supervised clustering can aid in solving tasks arising in network environments, ranging from network traffic classification to anomaly detection. In particular, the advantage can be taken of the fact that certain network transmissions are detected to fall within the same category (cluster), since they share similar behaviors and/or characteristics. In this chapter, we show how clustering of the network traffic can be converted into an effective tool for detecting anomalies or intrusions.

The robustness of intrusion detection systems is highly conditioned by their capacity to adapt to the changes in network environments, on one hand, and to receiving previously unseen types of traffic, on the other hand. The state-of-the-art learning methods, however, do not show such flexibility: they would need to be continuously retraining, e.g., using online learning, on new coming data, which might considerably increase time and resource costs.

As it will be revealed in this chapter, the supervised clustering approach is able to adapt to the current network state without requiring retraining. The structured prediction formulation effectively captures inter-instance similarity patterns. This way, the approach learns to predict clusters of the traffic instances of the same type and classify the new incoming instances based on such clusters. Its application to a network scenario of the NSL-KDD dataset shows impressive improvements over the classification approaches.

4.1 Overview

Intrusion detection component is crucial for any computer system or network. The task of detecting intrusions (or anomalies) is classically viewed as a classification problem in which a model is trained to distinguish between good and bad traffic behavior. A variety of classification

approaches, not restricted to SVM [Raman et al., 2017], Random forest [Farnaaz and Jabbar, 2016], and neural networks [Subba et al., 2016], has been proposed in the literature. The current top classification models report the accuracy of more than 95%. However, such high results are unrealistic as they were obtained in problematic evaluation settings: (i) tuning of the model parameters on the test set [Subba et al., 2016] or (ii) cross-validation [Raman et al., 2017; Farnaaz and Jabbar, 2016], which is rather inappropriate as the distribution of the instances is supposed to be different between training and test set to simulate a real-world scenario.

In particular, model evaluation using cross-validation, which is valid for a general classification scenario, might not give an adequate assessment of the accuracy of an intrusion detection system. This is because it is rather prone, if only the folds are not formed adequately, to violate the requirement of the task, according to which the test data is not supposed to contain traffic with anomaly types that were seen on training. Indeed, the benchmark intrusion detection KDD Cup 1999 dataset [Archive, 1999] carefully respects the above property. In contrast, including the test data into the training loop makes the system diverge from the realistic setting.

In our work, we follow the guidelines of the KDD Cup 1999 competition in the evaluation of the results, i.e., we train the models using only training data, and evaluate them on test data. In our attempt to apply cross-validation, the classification accuracy of previous methods increased by more than 20% compared to the standard setting. This confirms that 95% of accuracy is an unrealistic number, and consequently, new models are needed for operational scenarios.

The above experiment also suggests that classification methods essentially tend to model the training data distribution while the situation in a network is subject to continuous changes, which imposes onto a system a requirement of adaptability. The supervised clustering approach we explore in this work addresses this issue. Its main difference from the classification view of the problem is that it is based on capturing the similarity patterns between traffic instances, thus is able to adapt to changes in the network environment.

Our operational hypothesis is that the categorization of the preceding traffic can be available with some delay. For example, after a certain time if no alarm has been received by users, system administrators, or lower-level automatic systems, we can consider that the given traffic is not affected by anomalies. Then, we can rely on such data when classifying the new instances. Our model dynamically exploiting the newly coming data improve the accuracy of the basic systems by more than 12% points. The strong side of the proposed approach is its ability to adapt to the network situation without the need to be retrained on more recent data.

In the following sections, we give a detailed description of the proposed modelling and provide the results of the experimental analysis in Section 4.5.

4.2 Structural Representation of Network Data

We use LSP^E approach of Fernandes et al. [2012, 2014] described in Section 2.2.2 for supervised clustering of the network data. This choice is motivated by the fact that, in a network scenario, we deal with a continuous data flow with a naturally defined order.

To train an algorithm, we need a labeled set: a collection of points pre-assigned to some clusters. Applied to a network scenario, we form directed graphs out of the pairwise "links" between transmission data points[1] (Figure 4.1). In these graphs, each node $t_i, i \geq 1$ corresponds to a network transmission, t_0 denotes the artificial root node. For illustration purposes, we flatten here the graph structures presented in Figure 2.1b to emphasize the continuity of the data flow, moreover, the graph in Figure 4.1 can be extended to the right continuously. We connect each transmission node t_i to each of the preceding transmission nodes $t_0, t_1, ..., t_{i-1}$. Each edge corresponds to a pair of transmission points (t_i, t_j), where $i < j$; it is associated with pairwise features and labeled 0 or 1 (in the training phase), depending on whether t_i and t_i fall into the same cluster or not. We assume that all transmissions of the same class type belong to the same cluster, e.g., in NSL-KDD, we consider all the normal traffic points to fall into one cluster, as well as all the points from each of the fine-grained attack types (back, buffer_overflow, ftp_write, guess_passwd, etc.) to be grouped together as well.

4.3 Features from Raw Network Sensor Data

Each transmission data point t is associated with a number of transmission parameters or attributes $\psi(t)$. We extract the pairwise feature representation $\phi(t_i, t_j)$ as follows: i) from the pair of the continuous-valued attributes $\psi_f(t_i)$ and $\psi_f(t_j)$, we form a pairwise feature with value:

$$\phi_f(t_i, t_j) = max_f - min_f + |\psi_f(t_i) - \psi_f(t_j)|,$$

[1]We use interchangeably throughout the paper the terms 'traffic instance', 'transmission', 'transmission data point', 'connection', 'connection record', 'connection data point'.

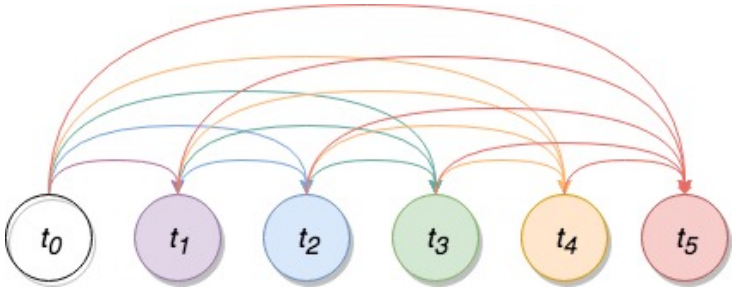


Figure 4.1: Graph of pairwise links between transmission nodes.

where max_f and min_f are the max and the min values, correspondingly, of the ψ_f over all the data points in the training set; and ii) from each binary attribute ψ_f, we make 3 binary features: $\mathbb{1}_{[\psi_f(t_j)=1 \& \psi(t_i)=1]}$, $\mathbb{1}_{[\psi_f(t_j)=1 \veebar \psi(t_i)=1]}$, and $\mathbb{1}_{[\psi_f(t_j)=0 \& \psi(t_i)=0]}$, where $\mathbb{1}_{[\cdot]}$ is the indicator function, same applied to each instantiation of each symbolic attribute.

The introduced pairwise features $\phi(t_i, t_j)$ are destined to express the similarity between the points of a pair. This way, the approach is different from the classification-based methods, as it enables the model to learn, based on these features, to predict how similar transmissions are and to group together also transmissions of the type crossing the data plane for the first time based on such similarities.

4.4 Inference

At the prediction stage, the algorithm computes weights for each edge using the learned model. Then we do the inference, which consists in finding the maximum spanning trees (Figure 4.2) on the sample graphs. Following the LSP$^{\text{E}}$ inference described in Section 2.2.2, we find such trees by iterating over the transmission nodes t_j and selecting for each of them the in-coming edge with the highest score. This way, each transmission node t_j is connected to another preceding node t_i with which they have the highest similarity according to the model. This means it is connected to the cluster to which t_i belongs. The tree in Figure 4.2 is one of the two possible ways to represent two clusters $\{t_1, t_4, t_5\}$ and $\{t_2, t_3\}$ (alternatively, t_5 can be connected to t_4). An advantage of such a graph-based model is its incremental nature, which enables a new coming data point to be clustered at a cheap cost within the current clustering without the need to reconsider the previous clustering decisions.

If t_j is the first occurrence of a particular class type in the data flow, the link from it to the artificial node (t_j, t_0) is considered correct (having label 1). At the prediction stage, if the model decides to link a new coming point t_j to t_0, it is probable that t_j belongs to an unseen traffic class type, therefore a potential anomaly. This way, in our experiments with NSL-KDD, we declare anomalous i) points that are predicted to be connected to the clusters of the known (previously seen) anomalous type, ii) points connected to t_0, and iii) points connected to ii), or,

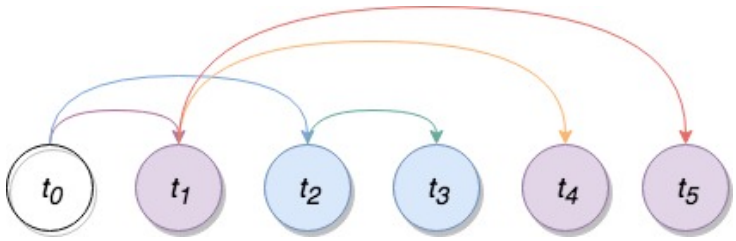


Figure 4.2: Spanning tree.

inversely, everything that is not connected to the normal traffic type cluster.

4.5 Experimental Analysis

4.5.1 Data description

We use an intrusion detection NSL-KDD[2] dataset [Tavallaee et al., 2009] which is a version of the most widely used benchmark KDD Cup 1999 dataset [Archive, 1999] solving some of its deficiencies, such as redundancy and duplicate records. The dataset is composed of a variety of connection records from the U.S. Air-force military network. The connection records attributed to normal traffic are annotated as one class type. The attack connections are labelled with 38 attack types, among which, 14 appear only in test data. Overall, there are approximately $126K$ training and $22.5K$ test single connection vectors with 41 attribute each.

4.5.2 Experimental setup

LSP model

The proposed LSP model is implemented on the basis of the Latent *SVM*^{struct} Yu and Joachims [2009] implementation[3].

Traning data sampling In theory, the structural model should be fed with a series of structured examples of the form $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ are sets of transmission nodes, and $\mathbf{y}$ – clusterings of such sets, from which we construct candidate graphs as described in Section 4.2. Instead, in the current scenario, such kind of transmission sets are not given, neither we intend to split the training data to obtain them since otherwise we risk to break the eventual continuity of the network traffic. What we do instead is considering the whole training set of transmissions as a continuous $\mathbf{x} = \{t_0, t_1, ..., t_N\}$, where N equals to the original training set size. Due to the huge size of the training set, for each node t_i, we limit the number of candidate arcs to 100 from the preceding nodes $t_{i-100}, ..., t_{i-1}$, which will later refer to as referent points, plus one link from the root (t_0, t_i). LSP updates (Line 8 of Algorithm 2) are now interleaved with inference: we update on tree segments encompassing nodes $t_j, ..., t_k$ with their in-coming arcs, where t_k is the first transmission node on which a mistake occurs at max-violating inference after the previous update, ended with t_{j-1}.

[2] `http://nsl.cs.unb.ca/NSL-KDD/`

[3] `www.cs.cornell.edu/~cnyu/latentssvm/`

Baselines

We compare the LSP approach against two classification baselines: SVM and perceptron. For the SVM baseline, we employ LIBSVM version 3.20 package [Chang and Lin, 2011]. We use our implementation of the binary perceptron.

Features

In all the three cases, at the preprocessing phase, we linearly scale the continuous-valued attributes to lie in the interval of $[0, 1]$ using LIBSVM. The rest, binary and symbolic-valued, attributes are converted into binary features as described in Section 4.3.

Evaluation

All the reported models are trained on the full training set and evaluated on the test set, no parameter tuning applied. We report the performance of the models in terms of the overall accuracy, detection rate, and false positive alarm rate:

$$Overall\ accuracy = \frac{TP + TN}{TP + TN + FP + FN},$$

$$Detection\ rate = \frac{TP}{TP + FN},$$

$$FP\ rate = \frac{FP}{FP + TN},$$

where TP is the number of correctly identified anomalies, FP is the number of normal traffic instances incorrectly identified as anomalous, TN - correctly identified normal traffic, and FN - incorrectly identified as normal.

4.5.3 Results

For making a decision regarding each test connection point, the LSP model relies onto a set of candidate referent points. In a realistic scenario, the points from the network history for which the type is already known could be used for reference. In this regard, two relevant issues arise: i) how to select the referent points, and ii) how many.

Standard classification scenario

In the evaluation setting of NSL-KDD, as a standard classification scenario, the correct labels are ”available” only for training instances. Therefore, we first evaluate LSP performance using

training instances as candidate references for each of the test instances. In Table 4.1, this corresponds to LSP TR model. In this experiment, we randomly select a subset of 280 training points containing representatives of all the training classes proportionally to the class sizes. Let us call this subset TR. In the mentioned model, TR points are used as referent for drawing predictions for each of the test points. LSP stays on a par with the baseline SVM models, with linear and polynomial kernels of degree 3. As mentioned in Section 4.1, there is a big gap between these numbers and 10-fold cross-validation results in Table 4.2 which we obtained for SVM.

Model			**Overall accuracy**	**Detection rate**	**FP rate**
LSP	TR		75.82	59.35	2.42
	TR + $\text{TE}_{pred.}$	$d = 100$	74.30	56.86	2.65
	TR + TE_{gold}	$d = 100$	89.18	86.61	7.85
	TR + TE_{gold}	$d = 200$	89.24	86.82	7.56
	TR + TE_{gold}	$d = 1000$	88.55	85.23	7.06
	TR + TE_{gold}	$d = 2000$	88.26	84.44	6.69
SVM	linear		74.96	61.58	7.34
	polynomial, degree 3		76.83	60.50	1.59
Perceptron			76.28	66.09	8.37
	online update on d test items	$d = 100$	77.46	67.44	8.20

Table 4.1: Comparison of anomaly detection models on the NSL-KDD test set.

Using recent traffic for reference

In the following scenarios, we aim to check the impact on the LSP performance of adding other test points to the test point's reference list, those from its close or relatively close history. For classifying each point, we compose a set, TE, of 100 consequent test points, preceding it and placed at distance d from it, e.g., $d = 100$ meaning that for point t_j, TE includes points t_i satisfying $d < j - i \leq d + 100$. Note that by doing so we exclude d immediate preceding points, allowing some delay.

In such first experiment, if a point is predicted to be linked to some of the TE points, it receives the predicted label of that point, (result subscripted *pred.* in Table 4.1). The results slightly worsen compared to using only TR for reference. Analysing the test output in this evaluation scenario reveals that the link predictions were reasonably good, however, a small number of prediction errors in the labels of TE points, propagated down through the test set, causes the observed drop in the classification accuracy. If instead the model is aware of the real labels of the TE points, the accuracy becomes considerably better (result subscripted *gold*). This reflects the main property of the LSP model of being able to learn from some connection point

kernel	**linear**	**polynomial, degree** 3
Overall accuracy	97.41	95.57

Table 4.2: SVM accuracy on the NSL-KDD training set in 10-fold cross-validation.

pairs and be successfully applicable to some rather different ones, since based on capturing the point similarity rather than the training data distribution. In a real network environment, this property is particularly valuable, as such a model is less dependent on the changes in the network traffic over time.

To exclude eventual homogeneity in the distribution of the class types of neighbouring test data points, we explore larger values of d. Increasing d preserves relatively high accuracy. It should be noted that some part of the observed small drop is due to the fact that, in the current evaluation scenario, for the first $d + 100$ test points, $TE = \emptyset$.

Comparison to online learning

To make a fair comparison to LSP using distant gold labels, we introduce another baseline – online binary perceptron with weight averaging. In our experiment, this model converges fast on the training set reaching decent overall accuracy of 97.19%, detection rate of 96.90%, and false positive rate of 2.48%. However, as it is often the case, it does not generalize that well on the unseen test data, as seen from the bottom section of Table 4.1. The perceptron reaches the generalization levels of SVM and LSP being slightly better in accuracy and detection rate and slightly worse in false positive alarm rate.

An appropriate model variant for a network environment could be the one allowing efficient real-time retraining. Perceptron can provide such functionality by performing online updates on the records of new traffic instances coming to the network plane. The last line in Table 4.1 shows the performance of such a model. It corresponds to the previous standard perceptron which we continue to update on the test data examples after being trained on the training set. Before drawing a prediction for the test point t_j, we make an update on the test point t_{j-d}. This way, the model sees all the traffic except for the preceding d connections. The accuracy gain due to such online updates on test, not exceeding 1.4% points, is much lower compared to that of LSP relying on the "recent" traffic. Moreover, perceptron requires a constant control as it risks to be skewed towards eventual predominating traffic and not to be able to adapt fast to the change of the network situation. At the same time, LSP has an advantage that it does not even require any retraining and adaptation.

4.6 Summary

The supervised clustering approach proved beneficial for intrusion detection in network traffic environments. The method has an incremental nature, which is able to effectively use recent network information in a structural way. It is based on the similarity patterns between the traffic instances arising in a network rather than on the properties and characteristics of individual instances. This enables the model to adapt efficiently to the eventual changes in the traffic situation without the need to be retrained. Such essential properties deliver a valuable accuracy gain with respect to the basic approaches, making it a good candidate for employment in a realistic network environment.

In this study, we focused primarily on the comparison of machine learning approaches to intrusion detection, without their in-depth enhancement using different data mining techniques such as feature engineering, and instance and feature selection. Regarding this, the proposed LSP approach can be improved in a number of ways.

The following several directions of enhancing the LSP model can be explored: i) finding the best method for selecting the referent candidate instances (or representatives of the preceding clusters) for a given new coming instance and in general, ii) exploiting the information about the coarse-grained anomaly categories (DOS, R2L, U2R, probing) of connection instances, iii) choosing effective feature representations considering also those of higher orders, spanning more that two instances, iv) testing the model's ability to be optimized to different network metrics most crucial in particular environments, either by means of tuning or directly, by injecting as a loss function in the learning algorithm.

Chapter 5

Structured Prediction for Coreference Resolution

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 regarding the use of LSSVM. Mainly, the structured approaches target the latent structured perceptron (LSP). In Section 5.2, we present a practical study comparing online learning with LSSVM. We analyze the intricacies that may make initial attempts to use LSSVM fail, i.e., huge training time and the much lower accuracy produced by the Kruskal's spanning tree algorithm. In this respect, we also propose a new 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.

An essential 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 respect, CR offers an interesting setting because it uses complex measures, e.g., MELA, which has been designed by domain experts based on semantic considerations. Consequently, defining an optimal loss function for CR is rather challenging as it requires a deep knowledge of this specific domain, added to, already complex, inherent clustering nature of the task. At the same time, the structured prediction approaches, LSSVM and LSP, optimize simple loss functions preserving the factorization of the objective onto the underlying structure.

In Section 5.3, we show that complex loss functions can be (i) automatically learned also from the controversial but commonly accepted CR measure – MELA, and (ii) successfully used in learning algorithms. We study the advantages of jointly learning loss functions and models, in Section 5.4. The accurate model comparison on the well-known CR benchmark dataset from CoNLL–2012 Shared task demonstrates the benefit of more expressive losses, although we have

to trade off the exact inference for enabling their use in the learning algorithm.

5.1 Coreference Resolution Task

We start by recalling the definition of coreference resolution and its benchmark evaluation metric MELA – Mention, Entity, and Link Average score (MELA) [Pradhan et al., 2012].

5.1.1 Definition

Coreference Resolution (CR) aims at linking together all the mentions referring to the same entities entities of a target text. CR is typically modeled as a clustering problem. Given $\mathbf{x} = \{m_i\}_{i=1}^{M}$, a set of entity mentions contained in the input text, the desired output $\mathbf{y}$ is a set of the corresponding mention clusters. For example, the text in Figure 5.1 contains three entities, *Tona*, *Australian Government*, and *China*, and several mentions of them, $m_1, m_2, ..., m_7$. The target $\mathbf{y}$ is then composed of three clusters $\{m_1, m_6\}$, $\{m_2, m_3, m_5, m_7\}$ and $\{m_4\}$.

5.1.2 Task Evaluation

Evaluation of the output of coreference systems is based on comparing the gold mention clusters or *key entity set* $K = \bigsqcup_i k_i$ with the *response entity set* $R = \bigsqcup_j r_j$, produced by a system. K is a disjoint union of sets k_i, each of which contains mentions referring to the same entity, equivalently for R. Note that real-life coreference systems are trained to predict clusters on mentions M detected by the system. Mention set M does not always coincide with the mentions comprised in the key entity set K. Thus, the ground truth clustering $\mathbf{y}$ supplied to the structured prediction learner is a contraction of K onto M. Let us denote the gold standard clustering corresponding to K as $\tilde{\mathbf{y}}$. R is the output $\hat{\mathbf{y}}$.

(Tona, the Australian foreign affairs minister)$_{m_1}$, recently said in Sydney that *(the Australian government)*$_{m_2}$ will maintain a one - China policy, and that the relationship between *(Australia)*$_{m_3}$ and *(China)*$_{m_4}$ is a central point of *(Australia's)*$_{m_5}$ foreign policy. *(Tona)*$_{m_6}$ made this announcement while explaining *(Australia's)*$_{m_7}$ Asian policy structure to foreign reporters.

Figure 5.1: Text fragment with seven highlighted mentions from the English part of the corpus from CoNLL 2012-Shared Task[1].

5.1.3 MUC

MUC coreference score [Vilain et al., 1995] is based on the number of correctly predicted links between mentions. The number of links required for obtaining the key entity set K is $\sum_{k_i \in K}(|k_i| - 1)$, i.e., 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. More precisely, the formula for MUC recall is

$$Recall = \frac{\sum\limits_{k_i \in K}(|k_i| - |p(k_i)|)}{\sum\limits_{k_i \in K}(|k_i| - 1)}. \tag{5.1}$$

The MUC Precision is computed by interchanging the roles of the key and response entities, i.e.,

$$Precision = \frac{\sum\limits_{r_j \in R}(|r_j| - |p(r_j)|)}{\sum\limits_{r_j \in R}(|r_j| - 1)}. \tag{5.2}$$

5.1.4 B^3

B^3 is a mention-based metric [Bagga and Baldwin, 1998]. 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:

$$Recall = \frac{\sum\limits_{k_i \in K}\sum\limits_{m \in k_i} Recall_m}{\sum\limits_{k_i \in K}|k_i|} = \frac{\sum\limits_{k_i \in K}\sum\limits_{r_j \in R}\frac{|k_i \cap r_j|^2}{|k_i|}}{\sum\limits_{k_i \in K}|k_i|} \tag{5.3}$$

B^3 Precision is computed by switching as well the roles of the key and response entities.

5.1.5 CEAF

$CEAF_e$ 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 $g^* : K \to R$, 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 Precision 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.

Mention, Entity, and Link Average (MELA)

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. MELA computation is rather expensive mostly because of $CEAF_e$. Its complexity is bounded by $\mathcal{O}(Ll^2 \log l)$ [Luo, 2005], where L 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.

5.2 Comparison of Structured Prediction Methods for Coreference Resolution

CR research has shown effective applications of structured prediction, e.g., the latent structured perceptron (LSP) by Fernandes et al. [2012, 2014] obtained the top rank in the CoNLL-2012 Shared Task [Pradhan et al., 2012]. There has been an ample 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; Samdani et al., 2014; Peng et al., 2015; Kummerfeld et al., 2015]. However, LSSVM by Yu and Joachims [2009], which offers theoretical guarantees on reducing the error upper-bound, is scarcely studied. The major advantage of such 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.

We empirically compare LSSVM with two online learning algorithms, LSP and LSPA (a structured passive-aggressive (PA) algorithm [Crammer et al., 2006] that we extend with latent variables) using the exact setting of the CoNLL-2012 dataset. This preserves comparability with the work in CR. We use the latest version of the MELA scorer[2].

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

[2] `conll.cemantix.org/2012/software.html`

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 apply two kinds of efficiency boost: feature and mention pair selection. Feature selection is rather challenging as the CR feature space is different from a standard text categorization setting. We cannot 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 explore the use of efficient binary SVMs for computing feature weights, which we used for 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.

5.2.1 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 Structural SVM problem, which in turn is polynomial [Tsochantaridis et al., 2004].

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, with a gradient taken with respect to 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 not 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.

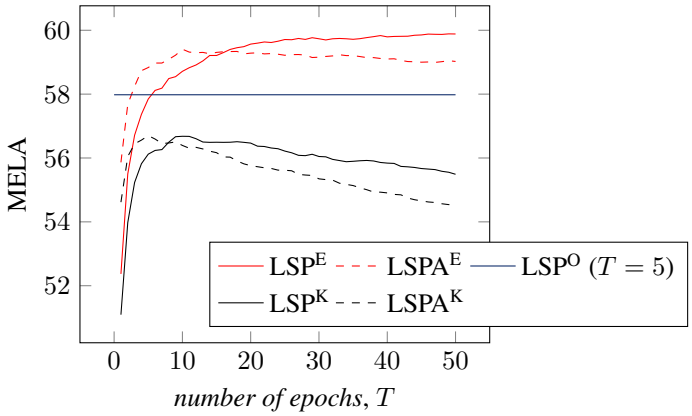


Figure 5.2: LSP learning curves, with 100 random documents used for training (all the features, all the edges), tested on all the development documents.

5.2.2 Experimental study

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. [2012, 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}.

[3] `conll.cemantix.org/2012/data.html`
[4] `www.cs.cornell.edu/~cnyu/latentssvm/`

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

Table 5.1: Best parameter combinations for structural approaches selected on CoNLL-2012 English development set.

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$^{\text{K}}$ and LSP$^{\text{K}}$, r is a penalty for adding an incorrect edge; in LSSVM$^{\text{E}}$ and LSP$^{\text{E}}$, r is a penalty for selecting an incorrect root arc. We select the parameters on the entire development set by training on 100 random documents from the training set. We pick up C from $\{1.0, 100.0, 1000.0, 2000.0\}$, the r values for LSSVM$^{\text{K}}$ and LSP$^{\text{K}}$ from $\{0.05, 0.1, 0.5\}$, and the r values for LSSVM$^{\text{E}}$ and LSP$^{\text{E}}$ from the interval $[0.5, 2.5]$ with step 0.5. The values reported in Table 5.1 were used for all our experiments.

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. [2012, 2014] noted that $T = 50$ was sufficient for convergence. Figure 5.2 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.

Model comparison

Table 5.5 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$^{\text{O}}$ 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 the Edmonds' algorithm outperform the undirected graphs used with Kruskal's; (iii) LSSVM$^{\text{E}}$ is around one point less than LSP$^{\text{E}}$ and LSPA$^{\text{E}}$; (iv) the training time of LSSVM$^{\text{E}}$ is one order of magnitude longer than that of LSP$^{\text{E}}$; and (v) LSSVM$^{\text{K}}$ took more than 1.5 months to converge.

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
LSPA^{K}	61.15	60.16	6	47.73
LSP^{E}	64.01	**63.04**	43	32.55
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 5.2: System results on CoNLL-2012 English development and test sets, using all the training documents for training. T_{best} is evaluated on the development set. *LSP^{O} is the result published in Martschat and Strube [2015].

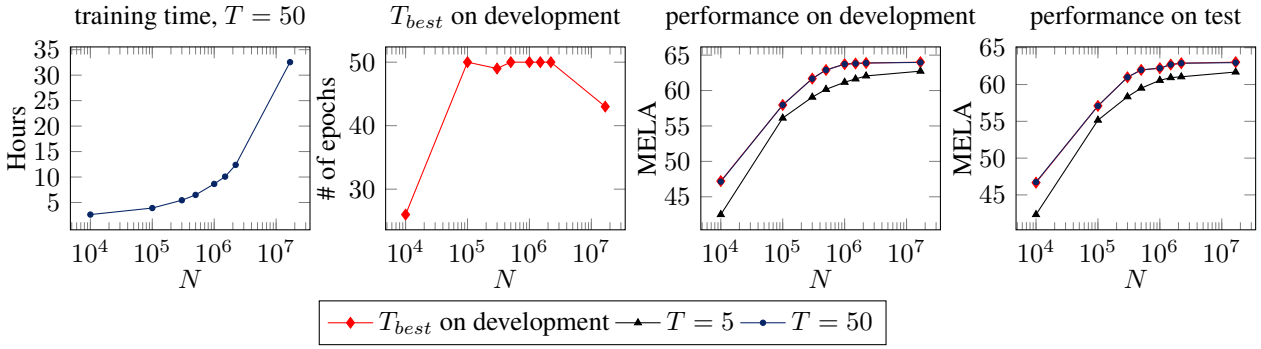


Figure 5.3: LSP^{E} training time and accuracy with respect to the number of features N, selected according to the binary classifier weights.

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 the 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 5.3 plots the accuracy of CR models, using different numbers of features selected as above. Interestingly, only retaining 5% of the features ($N = 10^6$) results in a small loss.

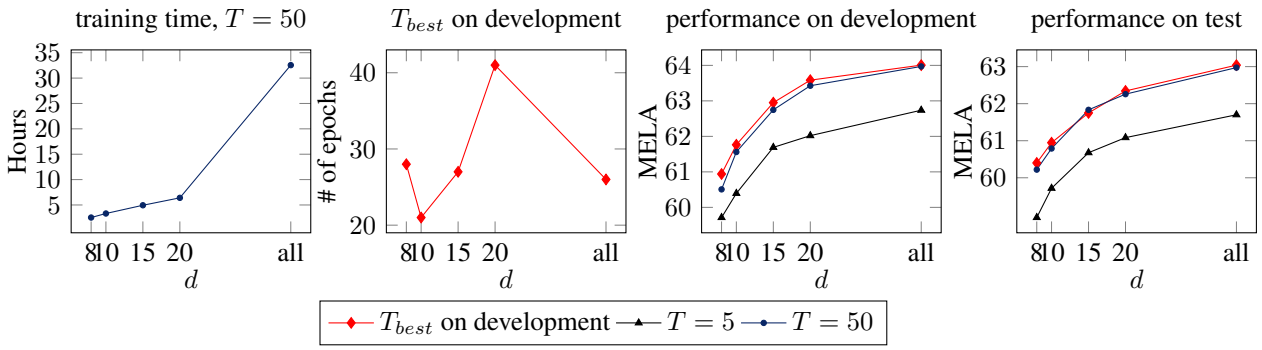


Figure 5.4: LSP$^{\text{E}}$ training time and accuracy with respect to d (max number of candidate antecedent edges for each mention).

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. [2012, 2014] to reduce the number of candidate links. Such sieve retains links between two mentions only if their distance is lower 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 5.4 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].

Results on filtered data

Table 5.3 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$^{\text{K}}$ is outperformed by LSP$^{\text{K}}$ (2 points) and performs worse than LSSVM$^{\text{E}}$; (iii) LSPA$^{\text{K}}$ seems to generalize better on filtered data than LSP$^{\text{K}}$; and (iv) w.r.t. no filtering, LSSVM$^{\text{E}}$ faces a lower drop in performance than LSP$^{\text{E}}$ does, approaching nearer to the latter.

5.2.3 Discussion

The results of our study are the following:

(i) we show that LSSVM can be applied to a realistic CR dataset and achieve the same state

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

Table 5.3: System results on CoNLL-2012 English development and test sets, using all training documents with filtered features (N=10^6) and edges (d=20).

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 the Edmonds' algorithm applied to the directed latent trees. They compared the previous inference approach with the spanning graph algorithm by Kruskal's 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.

5.3 Learning and Optimizing a Complex Clustering Metric

CR provides an interesting application scenario of structured output methods in NLP where complex measures defined by domain experts cannot be optimized by simple loss functions.

As seen from Section 5.2, recent methods based on structured perceptron, e.g., Fernandes et al. [2012, 2014]; Björkelund and Kuhn [2014]; Martschat and Strube [2015], provide efficient solutions representing $\mathbf{h}$ as trees. The efficiency and effectiveness of these models come from the optimization of simple loss functions, essentially based on the number of errors in detecting graph edges. These functions may be suitable for optimizing a measure of a general clustering task, where clusters do not have any specific semantic meaning, which is not the case for CR. For instance, let us consider the example in Figure 5.1 and suppose that System 1 outputs the correct cluster $\{m_1, m_6\}$ but mistakes all the other edges, whereas System 2 correctly detects $\{m_2, m_3\}$ only. The two systems are equivalent considering the count of correct/incorrect edges but any expert of CR would judge the first system more accurate since it correctly detects an entire entity, i.e., *Tona*.

This kind of analyses has led to the design of many measures, e.g., eight different versions of the MELA score [Pradhan et al., 2012]. MELA has received the largest consensus from the CR community, although several drawbacks have been pointed out [Moosavi and Strube, 2016], e.g., simple loss functions for structured prediction cannot optimize it, thus motivating research directions on direct measure optimization [Le and Titov, 2017]. CR setting is extremely complex for machine learning designers as they need to closely work with the CR domain experts to model loss surrogates capturing the intuition over the specific domain measure.

We study the possibility to automatize the transfer of the expert knowledge on the required measure to the learning algorithm. We focus on the CR application domain as it offers the possibility of learning loss functions that optimize structured prediction algorithms over complex domain-specific measures. We follow the general framework for learning with a learned loss defined in Chapter 3. More in detail, first, we learn MELA using efficient linear regression models based on aggregate features, e.g., Precision, Recall, and their approximaitons. We generate labelled training data for our regressors, applying MELA to the structures generated during the conventional structured prediction learning. It should be noted that MELA requires the alignment of mentions between the generated output and the gold standard mention clustering. This results in a non-convex and non differentiable function. Moreover, the running time is very large, preventing any possibility to use MELA directly in a learning algorithm.

Secondly, since we could not find a factorization of the corresponding loss Δ_ρ, we designed a latent structured perceptron LSP$_\rho$ that can optimize non-factorizable loss functions on CR graphs. We experimented with LSP using Δ_ρ and other traditional loss functions using the same setting of the CoNLL–2012 Shared Task on CR [Pradhan et al., 2012], thus enabling an

exact comparison with previous work. In particular, we tested our LSP_ρ on different conditions, e.g., with smaller feature spaces, languages and datasets. The results show that (i) Δ_ρ can be effectively learned and improve the state of the art of the LSP models, (ii) our approach requires much fewer number of epochs for convergence than the counterparts; and (iii) it delivers a large improvement in a setting, where fewer manually engineered features are available, e.g., on the Arabic CoNLL–2012 data. It is worth noting that Δ_ρ learned for the English task can be applied to a different domain (and language) – the Arabic CR task, demonstrating that Δ_ρ is invariant to the different CR settings, as expected.

Thirdly, we studied the influence of the joint learning of a loss function and models utilizing it (see algorithms in Section 3.3 of Chapter 3). The further improvement produced by this approach suggests interesting research lines in learning models together with their loss functions.

5.3.1 Related work

$\text{SVM}^{\text{cluster}}$ by Finley and Joachims [2005] – the first structured output approach applied to CR, introduced in Section 2.2.2 – enables the optimization of any clustering loss function (including non-decomposable ones). The authors show experimentally that optimizing a loss function with approximated clustering techniques results into a better classification accuracy in terms of the same very loss. The loss functions explored in their work allow for fast computation, but the approach is not viable for more realistic benchmarks (large dataset) and measures, such as MELA.

SampleRank – a large margin stochastic approach for structured prediction of Wick et al. [2011] – also directly optimizes a global but simple coreference loss, which has a certain degree of factorizability, and does not need to be fully reevaluated for each local search step. Tarlow and Zemel [2012] study the cases from image processing when a global loss does not factorize over graphical structure. The authors' remark that putting the complexity into the loss rather than into the model facilitates inference on test: this also applies to our approach. The examined loss functions again allow for fast computation, thus, the main difficulty, as well as the focus, is on customized methods for optimizing them, which include relaxing conditions on the loss compounds, constructing dynamic programming procedures to facilitate the loss optimization. Ranjbar et al. [2010] propose a piecewise linear approximation method for optimizing a complex loss and a complex feature function. Their approximation requires a target complex measure to be computed by the contingency table, which is not the case of MELA. In contrast, the MELA induced loss by itself shows prohibitively expensive computation time, which our work contributes to alleviate.

Regarding the direct optimization of CR metrics, the solution proposed by Zhao and Ng [2010] consists in finding an optimal weighting (by beam search) of training instances, which would deliver a classifier maximizing the target coreference metric on the same training set.

Their models optimizing MUC and B^3 delivered 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. Throughout this work, we use structured perceptron models due to their efficiency, however, one may think of optimizing our loss approximation using $\text{SVM}^{\text{cluster}}$, SampleRank or an alternative structural perceptron update rule derived by McAllester et al. [2010], which, in a long run, approximates the direct structural loss minimization. Le and Titov [2017] derive a differentiable representation of B^3. Clark and Manning [2016] optimize B^3 directly. For efficiency reasons, both the previous works omitted the optimization of CEAF_e, which is instead part of MELA.

5.3.2 Surrogate loss functions

When defining a loss function, it is very important to preserve the factorization along the graph edges to exploit the efficient inference algorithms (see Sec. 5.3.4). Fernandes et al. use 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

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

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

Yu and Joachims' loss, based on undirected graphs, is computed as follows:

$$\Delta_{YJ}(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}}) = n(\mathbf{y}) - k(\mathbf{y}) + \sum_{\mathbf{e} \in \hat{\mathbf{h}}} l(\mathbf{y}, \mathbf{e}), \tag{5.4}$$

where $n(\mathbf{y})$ is the number of graph nodes, $k(\mathbf{y})$ is the number of clusters in the ground truth $\mathbf{y}$, and $l(\mathbf{y}, \mathbf{e})$ assigns -1 to any edge $\mathbf{e} = (m_i, m_j)$ connecting nodes from the same cluster in $\mathbf{y}$, and r otherwise:

$$l(\mathbf{y}, \mathbf{e}) = \begin{cases} -1, & \text{if } \mathbf{y}(i) = \mathbf{y}(j), \\ r, & \text{otherwise.} \end{cases}$$

In our experiments, we employ both the loss functions, however, we measure Δ_F, in contrast to Fernandes et al., always 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. Thus, we adopt

$$\Delta_F(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}}) = \sum_{\mathbf{e} \in \hat{\mathbf{h}}} l(\mathbf{y}, \mathbf{e}), \tag{5.5}$$

where

$$l(\mathbf{y}, \mathbf{e}) = \begin{cases} 1 + r, & \text{if } m_i = root \wedge m_j \text{ does not start a new cluster in } \mathbf{y}, \\ 1, & \text{if } m_i \neq root \wedge \mathbf{y}^i \neq \mathbf{y}^j, \\ 0, & \text{otherwise.} \end{cases}$$

Note, that $\forall \mathbf{h}^* \in H(\mathbf{x}, \mathbf{y})$, $\forall \hat{\mathbf{h}} \in H(\mathbf{x})$, $\Delta_F(\mathbf{h}^*, \hat{\mathbf{h}}) \geq \Delta_F(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}})$, as in Equation 5.5 we do not penalize edges which are correct according to the gold clustering $\mathbf{y}$.

The loss functions in Equations 5.4 and 5.5 are simple in a sense of the general definition by Equation 3.2 in Section 3.1, since based on counting the mistaken edges. They 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.

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

Proof. Equations 5.5 and 5.4 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. For instance, 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. 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.

5.3.3 Automatically learning loss functions

As computational reasons prevent us from injecting MELA directly in LSP (see our inexact search algorithm in Section 5.3.4), we study methods for approximating it with a linear regressor. Considering the specifics of the CR task, we suggest slight customizing of the general formulation of the learned loss in Equation 3.3. Since MELA($\tilde{\mathbf{y}}, \hat{\mathbf{y}}$) score lies in the interval $[100, 0]$, a simple approximation of the loss could be:

$$\Delta_\rho(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}}) = 100 - \mathbf{w}_\rho \cdot \Psi(\tilde{\mathbf{y}}, \hat{\mathbf{y}}). \tag{5.6}$$

Note that, in this formulation, we make reference to the real gold standard clustering $\tilde{\mathbf{y}}$. This way, the clustering decisions during max-violated inference with respect to Δ_ρ will also reflect the mention detection accuracy. We use a simple linear SVM to learn a model $\mathbf{w}_\rho$.

In the following, we describe the features constituting $\Psi(\tilde{\mathbf{y}}, \hat{\mathbf{y}})$, show the improved version of Δ_ρ and LSP$_\rho$ for learning with it based on inexact search.

Features for learning measures

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).

Generating training and test data

The features described characterize the clustering variables $\hat{\mathbf{y}}$. For generating training data, we collected all $\hat{\mathbf{y}}$ in correspondence to the max-violating trees $\hat{\mathbf{h}}$ produced during LSP$^{\text{E}}$ learning (using Δ_F) and associate them with their correct MELA scores from the scorer and the above features. 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$^{\text{E}}$ on each document separately to obtain more variability in $\hat{\mathbf{y}}$'s.

5.3.4 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$_\rho$ algorithm that can use our learned loss in an approximated max search.

A general inexact decoding algorithm

If the loss function can be factorized over tree edges (see Equations 3.2, 5.5) the max-violating constraint in Line 5 of Algorithm 2 can be efficiently found by exact decoding using the same inference procedure as for prediction, e.g., Edmonds' algorithm as in Fernandes et al. [2012, 2014] or Kruskal's as 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 by Algorithm 5.

Algorithm 5 Max-violating Spanning Tree

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

Algorithm 6 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; set of the incoming candidate edges, $E(\mathbf{v})$, $\mathbf{v} \in V$; $\mathbf{w}$
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: $\quad\quad \mathbf{h} = \hat{\mathbf{h}} \setminus \mathbf{E}_{\hat{\mathbf{h}}}(\mathbf{v})$
9: $\quad\quad \hat{\mathbf{e}} = \underset{\mathbf{e} \in E(\mathbf{v})}{\operatorname{argmax}}\, \mathbf{w} \cdot \phi(\mathbf{e}) + C \cdot \Delta(\mathbf{y}, \hat{\mathbf{y}}, \mathbf{h} \cup \mathbf{e})$
10: $\quad\quad \hat{\mathbf{h}} = \mathbf{h} \cup \hat{\mathbf{e}}$
11: $\quad\quad score = score + \mathbf{w} \cdot \phi(\hat{\mathbf{e}})$
12: $\quad$ **end for**
13: $\quad score = score + \Delta(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}})$
14: **until** $score = prev\_score$
15: **return** max-violating tree $\hat{\mathbf{h}}$

Such efficient inference procedure cannot be applied to Δ_ρ, which is non-factorizable. Thus, we designed Algorithm 6, which is a greedy solution for finding an approximate maximum spanning tree with respect to a general non-factorizable global loss Δ. Its main idea is, starting from an empty tree (edge set), $\hat{\mathbf{h}} = \emptyset$, to build a locally-max violating tree as follows: for each vertex $\mathbf{v}$ of the candidate graph G at a time (Line 7), we remove its incoming edge, $\mathbf{E}_{\hat{\mathbf{h}}}(\mathbf{v})$,

from the current solution $\hat{\mathbf{h}}$, in order to check if there is a more violating edge connecting $\mathbf{v}$ with respect to the current $\hat{\mathbf{h}}$. For this purpose, Line 9 finds the edge $\hat{\mathbf{e}}$ connecting $\mathbf{v}$ that scores maximum $\mathbf{w} \cdot \phi(\mathbf{e})$ with respect to model and the loss $\Delta(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}} \cup \mathbf{e})$. After this step, the new max-violating tree contains $\hat{\mathbf{h}}$ built up to now plus a candidate edge $\hat{\mathbf{e}}$ (Line 10), where $\hat{\mathbf{h}}$ is partial if it is the first run over the nodes.

On a high level, Algorithm 6 resembles the inference procedure of Wiseman et al. [2016], who use it to optimize global features coming from an RNN. Differently from them, we repeat the procedure, after processing all the vertices, until the score of $\hat{\mathbf{h}}$ no longer improves. Intuitively, even if Algorithm 6 does not guarantee to deliver the max-violating constraint, a solution optimizing a more accurate loss function may lead to higher overall accuracy and faster convergence.

Notes on convergence

The following proposition assesses the convergence of our algorithm based on the learned loss function.

Proposition 2 (LSP$_\rho$ convergence)**.** *The Latent Structured Perceptron (Algorithm 2) using the greedy Algorithm 6 to optimize $\Delta_\rho(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}})$ defined in Equation 5.6 always converges when the data is linearly separable.*

Proof. Algorithm 6 converges as it greedily searches for all edges providing a loss increase and stops when there is no increase. More specifically, steps in Lines 8 and 9 assure the monotonicity of the scoring function. Since the score cannot decrease, the exit condition in Line 14 will eventually be met.

If the data is linearly separable, LSP$_\rho$ – Algorithm 2 using Algorithm 6 for max-violating inference – converges since it is basically a perceptron algorithm optimizing an objective function augmented with a loss function. □

Björkelund and Kuhn [2014] perform inexact search on the same latent tree structures as well, to extend the model to non-local features. In contrast to our approach, they use beam search and accumulate the early updates. Their tests show that early perceptron updates, in themselves, considerably slowdown the convergence of the perceptron.

Apart from the necessity to the design of an algorithm enabling the use non-factorizable loss Δ_ρ, there are other intricacies caused by the lack of factorization that need to be taken into account, which are revealed in the next section.

Approaching factorization properties

The Δ_ρ defined by Equation 5.6 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 Algorithm 6, 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.

5.3.5 Experimental study

In our experiments, we first show that our regressor for learning MELA approximates it rather accurately. According to the results of our study on comparison of the structured output approaches in Section 5.2, LSP$^{\text{E}}$ reaches the highest accuracy at the lowest cost. Thus, we examine the impact of our Δ_ρ by comparing LSP$_\rho$, derived on the basis of LSP$^{\text{E}}$, to LSP$^{\text{E}}$ models optimizing the other loss functions, Δ_F and Δ_{YJ}, which are defined in Section 5.3.2. We show that the impact of Δ_ρ is amplified when learning in smaller feature spaces.

Setup

Here, we follow the setup previously defined in Section 5.2.2.

Data In addition to the English data, we use also the Arabic part of the corpus from CoNLL 2012-Shared Task. The Arabic data includes 359, 44, and 44 documents for training, dev. and test sets, respectively.

Models We employ the LSP$^{\text{E}}$ implementation, which originally optimizes Δ_F. Here, we refer to this model as LSP$_{\text{F}}$, to distinguish the loss function optimized. We implement the LSP$^{\text{E}}$ version using Δ_{YJ} – LSP$_{\text{YJ}}$, and LSP$_\rho$ itself. As before, in the English experiments, we use the data preprocessed by cort [Martschat and Strube, 2015]. For Arabic, we used mentions and

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

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. The best T – the number of epochs – is selected on the development set from 1 to 50.

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

Learning loss functions

For learning MELA, we generated training and test examples from LSP_F according to the procedure described in Section 5.3.3. 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 5.4 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 5.5 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.

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

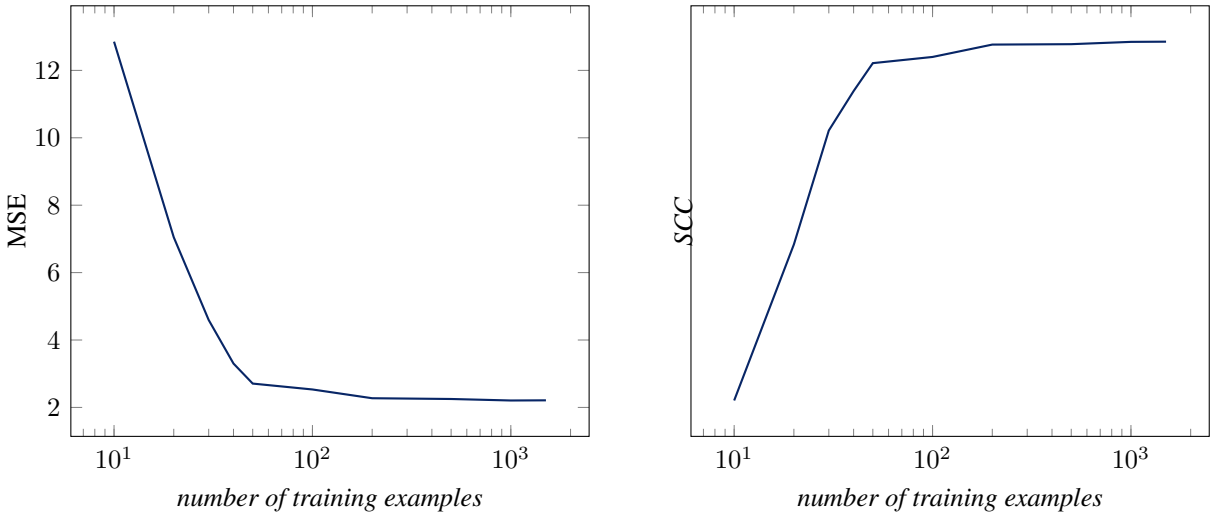


Figure 5.5: Regressor Learning curves.

The latter shows a similar behaviour, approaching a correlation of about 99.8% with real MELA.

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 5.5.

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).

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. [2012, 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 5.6 shows the breakdown of the MELA results in terms of its components on the test set. Interestingly, LSP_{ρ} 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 ~ 16.8M features, satisfy the separability condition of Proposition 1.

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 5.5: Results of our and previous work models evaluated on CoNLL-2012 English development and test sets, using for training all the training documents with All and $1M$ features. T_{best} is evaluated on the development set.

# features	**Model**	**Test set accuracy**			
		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_ρ	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_ρ	72.09	60.11	57.07	63.09

Table 5.6: Results on CoNLL-2012 English test set using the same setting of Table 5.5 and the measures composing MELA.

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 the feature selection approach, described in Section 5.2.2, based on 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).

The MELA produced by our models using all the training data is presented in Figure 5.3.

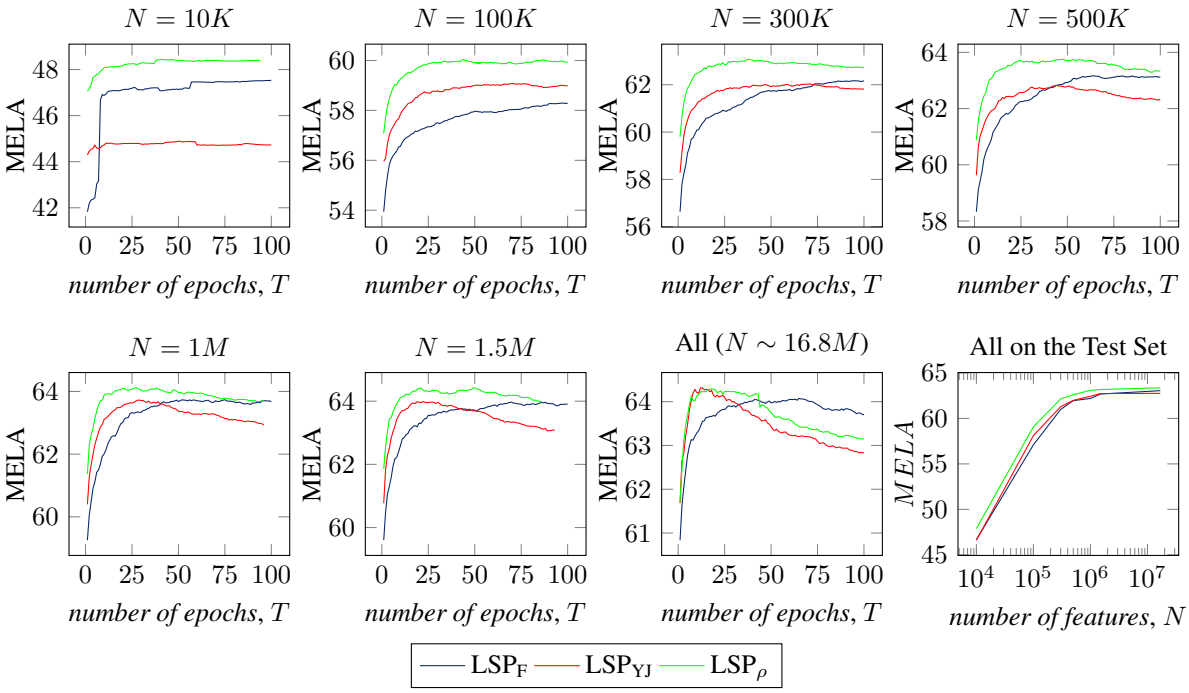


Figure 5.6: Results of LSP models on CoNLL-2012 English development set using different number of features, N. The last plot reports MELA score on the test set of the models using the optimal number of epochs tuned on the development set.

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

Secondly, when using all features, LSP_ρ 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_ρ is constantly better than the other models, though decreasing its effect as the feature number increases.

Next, in Column 1 (Selected) of Table 5.5, we report the model MELA using 1 million features. We note that LSP_ρ 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. Each of the features comprising regression examples x_ρ separately by definition satisfy the relaxed condition of Proposition 1, however, the learned

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

Table 5.7: Results of our and baseline models on CoNLL-2012 Arabic development and test sets, using all the training documents for training. T_{best} is evaluated on the development set.

model $\mathbf{w}_\rho$ is not exempt from containing also negative weights. However, by checking the MELA score obtained on the training set, we empirically verified that LSP_ρ always optimizes MELA, iterating for fewer epochs than LSP variants using the other loss functions.

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 5.7 reveal an indisputable superiority of LSP_ρ 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_ρ 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 5.3.5) in LSP_ρ on Arabic. This corresponds to LSP_ρ^{EN} model. Notably, it performs even better, 1.5 points more, than LSP_ρ 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.

5.4 Jointly Learning Loss and Model

In the previous section, we have shown that we can learn a linear regressor from a complex measure such as MELA for CR and we can transform it in a loss function that can be used in

	10k	100k	300k	500k	1M
LSP_F	43.59 ± 1.30	54.38 ± 0.66	57.18 ± 0.31	58.44 ± 0.35	59.01 ± 0.41
LSP_YJ	43.79 ± 0.58	55.61 ± 0.25	57.90 ± 0.31	58.33 ± 0.23	58.74 ± 0.33
LSP_ρ	47.03 ± 0.29	55.89 ± 0.28	58.17 ± 0.24	58.58 ± 0.31	58.80 ± 0.30
LSP^*_ρ	$\mathbf{47.07} \pm 0.22$	56.00 ± 0.28	$\mathbf{58.33} \pm 0.19$	$\mathbf{58.88} \pm 0.19$	$\mathbf{59.23} \pm 0.34$
$\text{LSP}^{*online}_\rho$	46.88 ± 0.13	$\mathbf{56.01} \pm 0.21$	58.27 ± 0.27	58.79 ± 0.21	59.05 ± 0.28

Table 5.8: Average MELA $\pm$ Standard Deviation on CoNLL-2012 English test set of models trained on eight disjoint samples of 100 documents from the training set.

the LSP algorithm. In this section, we apply the joint models LSP^*_ρ and $\text{LSP}^{*online}_\rho$ defined in Section 3.3.

5.4.1 Notes on convergence

The convergence of LSP^*_ρ may result tricky as the target of the optimization objective changes at each epoch, i.e., the loss function, Δ_ρ changes during training. However, our preposition below assesses that the joint learning can be reliably carried out in case of linearly separable data.

Proposition 3 (LSP^*_ρ convergence)**.** *Algorithm 3 that jointly learns a model together with* $\Delta_\rho(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}})$ *always converges when the data is linearly separable.*

Proof. Line 7 of Algorithm 3.3 generates data from the candidate tree $\hat{\mathbf{h}}_i$, found using the current model. Let us suppose that for some reasons the model does not converge. The model changes inside the loop and generates new training examples, x_ρ. In the worst case, the model can generate all possible examples, which are limited by all possible spanning trees for each document. Although, this can be a large number, eventually, the training set X_ρ will be a stable training set, consequently Δ_ρ will not change anymore, enabling the same convergence rationale of LSP_ρ (see Proposition 2). □

A similar rationale to Proposition 3 can be applied to the convergence of Algorithm 4. In practice, the convergence of online joint learning can be assured by a simple consideration: when the accuracy of the regressor for Δ_ρ is enough good, there is no need anymore to re-train it and not even to generate other data. Thus, we simple add the instruction at lines 18-20, which will disable the regressor training when it does not improve anymore, thus Proposition 2 can be applied again.

5.4.2 Results of the joint learning model

In these experiments, we tested if the joint modeling of CR and its loss is effective. Tab. 5.8 reports the comparison between the previous LSP model and LSP^*_ρ using the average of MELA

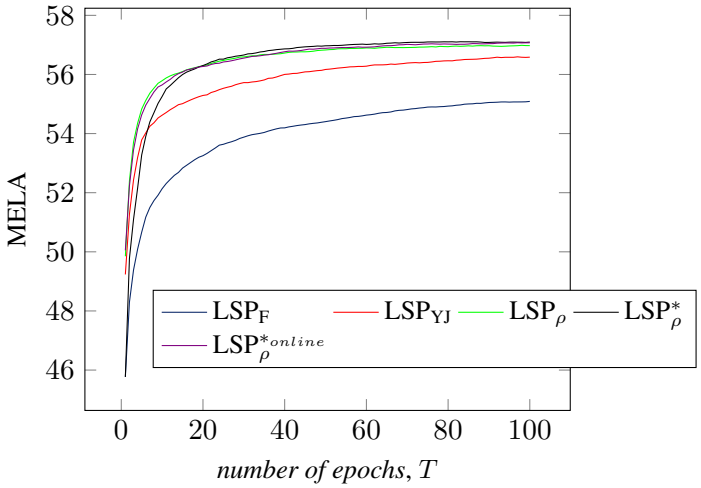


Figure 5.7: LSP learning curves on CoNLL-2012 English development set, averaged over 8 disjoint samples of 100 random documents from the training set.

over eight disjoint samples of 100 documents extracted from the training set and tested on all documents of the official test set. The different columns report experiments with different number of features. We note that (i) when the number of dimensions is small, our LSP$_{\rho}$ models are clearly superior, e.g., 4 absolute points better with 10K features; (ii) LSP$_{\rho}$, LSP$^{*}_{\rho}$ and LSP$^{*online}_{\rho}$ perform comparably with any number of features; and finally (iii) the improvement over traditional loss functions decreases when the space becomes easily separable, e.g., 1M dimensions.

The above experiments suggest that there is no much difference in accuracy using joint learning. Thus, we explored its impact on convergence by plotting MELA according to the number of epochs, T, ranging from 1 to 100. Figure 5.7 shows the learning curves of models using different loss functions and different algorithms: LSP$_{\rho}$ and LSP$^{*}_{\rho}$ seem much better than models using standard loss functions, i.e., LSP$_{\mathrm{F}}$ and LSP$_{\mathrm{YJ}}$, both in terms of accuracy and convergence speed.

LSP$^{*}_{\rho}$ seems equivalent to LSP$_{\rho}$, however, the former does not require to generate data in advance for learning Δ_{ρ}.

5.5 Summary

In this chapter, we tested an interesting machine learning problem – learning loss functions for structured prediction, targeting CR, which becomes particularly compelling when an evaluation measure captures specific background knowledge of the application domain, which cannot be surrogated by simple loss functions.

First, we conducted a comparative analysis of online and batch methods for structured pre-

diction 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 illustrates that all these methods, inherently optimizing the same objective, are able to achieve the same optimum. Additionally, we observed 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.

We also demonstrated that a noticeable benefit to the 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. Therefore, given the scale of our investigation, we further limit to LSP our study on the learned loss.

We showed that a complex measure, such as MELA, can be learned by a linear regressor with high accuracy and effective generalization. For optimizing the corresponding learned loss Δ_ρ, we design a new LSP$_\rho$ based on inexact search. Its results demonstrate that an automatically learned loss can be optimized and achieve competing performance in a real setting, including thousands of documents and millions of features, the CoNLL–2012 Shared Task. The results support the property of optimal loss functions: the improvement of LSP$_\rho$ over the counterparts using simple loss functions is not significant in the cases close to separability. However, when separability is more complex, LSP$_\rho$ is more accurate and faster. The joint models, which learn the LSP model and the loss Δ_ρ simultaneously, improve over it both in accuracy and speed.

Our approach of loss learning exhibits also good generalization properties. The cross-lingual experiments, in which a loss model learned from data coming from one language was applied to training of a model for another language, delivered significant improvements over the baselines.

This study opens several future directions, ranging from defining algorithms based on automatically learned loss functions to learning more effective measures from expert examples. There is also a lot of room for developing an interesting theory, which can impact on practical applications.

Chapter 6

Structured Prediction for Ranking

This chapter contains our recent developments and ongoing work on applying structured prediction methods to ranking. The structured output framework provides a helpful tool for learning complex ranking representations. We propose a structured perceptron approach which regards rankings as latent structural variables and combines them with auxiliary cluster graphs. The approach addresses such hard to optimize ranking metric as Mean Average Precision (MAP). We provide an inference procedure for finding the max-violating joint ranking and clustering structure based on the decomposition of the MAP loss. We give our preliminary results of the experiments we conduced for the task of answer passage re-ranking for question answering (QA).

6.1 Task formulation

We have training examples of the form $\{\mathbf{x}_i, y_i\}$, where $\mathbf{x}_i = \{q_i, D_i\}$, q_i is a query, $D_i = \{d_i^j\}_{j=1}^{N_i}$ is a set of items corresponding to q_i; the gold labelling $y_i = \{y_i^j : y_i^j \in \{0,1\}\}_{j=1}^{N_i}$ is a vector of gold item labels, label y_i^j corresponding to item d_i^j, taking value of 1 for relevant (good or positive) items and 0 – for irrelevant (bad or negative).

The task is to learn to predict, for each example x_i, a ranking of its items $r(\mathbf{x}_i) = r(q_i, D_i)$, such that the relevant items, d_i^j with gold labels $y_i^j = 1$, are always at top positions in $r(q_i, D_i)$. Finally, $r(q_i, D_i) = \{r^j\}_{j=1}^{N_i}$ is a permutation of the set D_i. In the following, we omit the example index i, where it is not needed, for simplification of the notation.

6.2 Overview

Ranking models are trained to reorder a list of candidate items D according to their relevancy to some query q. A reranking function is learned to score the relevant items higher than irrelevant.

In simple unstructured models, it is searched for as a real-valued function defined over the space of the query-item feature representations $f_{qd} : \phi(q, d) \rightarrow \mathbb{R}$. In preference ranking [Joachims, 2002], such a linear mapping is learned from the pairs of items (d^j, d^k), where d^j is higher in the optimal rank r than d^k with respect to q. This way, the training examples are treated independent with respect to the rest of the candidate list D and the learning is based only on the match between the item d and the query q.

The current state-of-the-art learning approaches for answer sentence re-ranking in QA mostly base on learning pairwise ranking signals or simple binary classification (relevant vs irrelevant). There have been promising attempts to learn global ranking functions which encompass the signals of all the candidates for a given query [Chapelle et al., 2007; Weston and Blitzer, 2012; Le et al., 2018]. Employing the structured output learning framework, such works represent a ranking as a structured object, with respect to which it is possible to optimize directly the ranking measures.

The structured ranking models exploit also the information about the structure of the candidate list D, e.g., that of how much similar or diverse the items in D are, in order to preserve such (in)consistency in the ranking. Thus, Weston and Blitzer [2012] propose a class of structural models over the latent embedding spaces:

$$f(q, r) = \sum_{j=1}^{|D|} w_j f_{qd}(q, r^j) + \sum_{j,k=1}^{|D|} w_j w_k f_{dd}(r^j, r^k),$$

which in addition to the ranking score based on individual query-item scores considers item-item similarities in the second term. Choosing weights w_j associated with the positions according to the schema:

$$w_j = \frac{1}{j}, \text{ if } i \leq m \text{ and } 0 \text{ otherwise},$$

they enforce consistency between top m items of the rank and diversity of the rest of the list with respect to them.

In this work, we abstract from the above structured formulation, and allow the presence of groups of similar items in the ranking list. There could be distinct groups of similar relevant items. At the same time, there could be similar irrelevant items as well. Consider an example form the Community Question Answering (cQA):

q =”Can anybody recommend me a dentist? A good one.”

d^1 =”Thanks for your answer.”

d^2 =”Thank you guys!”

Both d^1 and d^2 being bad for the query q are very similar, and it might be beneficial for a model to use this information and to place the two candidates in the rank close to each other.

Thus, we propose a structural model that combines ranking and clustering in one place:

$$f(q, r, c) = f_{rank}(q, r) + f_{clust}(q, c),$$

where the ranking r and clustering c are tied between each other in a way that elements of the same cluster c_i should occupy neighbouring positions in r without being interfered by the elements of the other clusters. Essentially, we deal with the combined structural object $\mathbf{y} = (r, c)$. The same ranking can correspond to different clusterings. Consider two extreme cases: i) putting all the D's items into one cluster v.s. ii) leaving them all singletons. We can obtain the same r by ordering the elements of one cluster or the singletons in the same way.

The re-ranking task does not fall straightforwardly within the structured prediction formulation as the ground truths for ranking objects are usually not provided in the training data (only relevance labels for candidates). Chapelle et al. [2007] select one among all possible correct rankings at random as a ground truth for training. Weston and Blitzer [2012] bypass the necessity of comparison to a complete ranking during training and sample the candidate pairs. This issue is seamlessly circumvent by using the latent structured prediction formulation.

Optimization of the target ranking measures is affordable when measures are factorizable, e.g., the structural SVM of Chapelle et al. [2007] makes use of the factorization properties of the Normalized Discounted Cumulative Gain (NDCG) ranking score. The case of MAP is rather involving. Yue et al. [2007] found an exact solution to the hinge-loss relaxation of Average Precision (AP) for the structural SVM approach. Chen et al. [2009] found tight upper bounds on AP using simple learning to rank loss functions. We derive a strict decomposition of the loss corresponding to AP and propose an approximate method for inference of the max-violating constraint with respect to it.

6.3 Structured Prediction for Ranking

The structured prediction framework for ranking [Chapelle et al., 2007; Le et al., 2018] considers a joint feature representation of an input example $\mathbf{x}$ together with an output ranking r: $\Phi(\mathbf{x}, r) = \Phi(q, D, r)$, which factorizes over the individual feature representations of items with respect to the query, weighted relative to the item positions j in the rank:

$$\Phi(\mathbf{x}, r) = \Phi(q, D, r) = \sum_{j=1}^{N} w_j \phi(q, r^j), \tag{6.1}$$

The typically used weighting schema, w, implies non-increasing weights associated with the positions j: $w_1 \geq w_2 \geq ... \geq w_N \geq 0$, where importance decreases gradually from the top to the bottom of the ranking.

Inferring a ranking corresponding to a linear model $\mathbf{w}$, i.e., finding

$$\underset{r \in R(\mathbf{x})}{\operatorname{argmax}} \, \mathbf{w} \cdot \Phi(\mathbf{x}, r)$$

among all possible rankings $R(\mathbf{x}) = R(q, D)$, reduces simply to ordering the items by scores $\mathbf{w} \cdot \phi(q, d)$, since w_j are fixed.

Since the correct ranking r^* for an example $\mathbf{x}$ is often not unique, Chapelle et al. [2007] choose one of the correct rankings at random as a gold label for training. This evidently biases the training towards the chosen ground truths. The above kind of problems is effectively alleviated within a latent structured prediction framework.

6.3.1 Our learning approach

Learning

We deal with not fully observed case as the ranking labels r, we intend to learn, are not given in the input data. We use LSP (Algorithm 2), which, in Line 7, finds such structure for the current example $(\mathbf{x}_i, y_i)$ – the best correct ranking r^* corresponding to the current model weights $\mathbf{w}_t$. The search here is restricted to the set $R(\mathbf{x}_i, y_i)$ of all rankings of the example $\mathbf{x}_i$ that comply with the gold label y_i, i.e., at which good items take top positions and bad – bottom positions. Thus, the operation is reduced to simple ordering of the good and bad items (separately) by weights, and putting the former to the top, and the latter – to the bottom of the resulting ranking. The max-violating $\hat{r}$ is found in Line 5 with respect to a ranking loss $\Delta(y_i, r)$, over all possible rankings $R(\mathbf{x}_i)$. Sec. 6.3.1 describes the procedure we use here for max-violating inference with respect to the loss corresponding to the MAP ranking metric. In Line 8, the weights $\mathbf{w}$ are updated using the structural feature representations (defined by Equation 6.1) of the two ranking outputs r^* and $\hat{r}$.

Max-violating inference

Our target is to optimize the MAP ranking metric. Thus, in training, we intend to minimize the following loss on structural examples which is the inverse of the average precision (AP):

$$\Delta_{ap}(y, r) = 1 - AP(y, r).$$

AP is a global measure, non-decomposable in a strict sense over the position variables, so that to enable iterative exact inference. Here, we propose a method for approximate inference with respect to Δ_{ap}, which is efficient and enables exact local search.

Let us denote by $P = |\{d|y(d) = 1\}|$ the number of good/positive items in the candidate list D, and by $I_j^+ = \mathbb{1}_{[y(r^j)=1]}$ and $I_j^- = \mathbb{1}_{[y(r^j)\neq 1]}$ – the indicator functions that the items in position

j in r is good and not good (positive and negative), respectively. Then,

$$AP(y,r) = \frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+\sum_{k=1}^{j}I_k^+.$$

We can have a strict decomposition of Δ_{ap} over negative items. We rewrite the AP formula as follows:

$$AP(y,r) = \frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+(\sum_{k=1}^{j-1}I_k^+ + I_j^+) = \frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+(\sum_{k=1}^{j-1}I_k^+ + 1).$$

Then,

$$\begin{aligned}\Delta_{ap}(y,r) &= 1 - \frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+(\sum_{k=1}^{j-1}I_k^+ + 1) = \frac{1}{P}(P - \sum_{j=1}^{N}\frac{1}{j}I_j^+(\sum_{k=1}^{j-1}I_k^+ + 1)) = \\ &= \frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+(j - 1 - \sum_{k=1}^{j-1}I_k^+) = \frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+\sum_{k=1}^{j-1}(1 - I_k^+) = \\ &= \frac{1}{P}\sum_{j=1}^{N}\frac{I_j^+}{j}\sum_{k=1}^{j-1}I_k^- = \frac{1}{P}\sum_{j=1}^{N-1}I_j^-\sum_{k=j+1}^{N}\frac{I_k^+}{k}. \end{aligned} \tag{6.2}$$

According to the last line of Equation 6.2, Δ_{ap} decomposes into a sum over all the positions j with negative items (those activating I_j^-) of quantities $l_j(y,r) = \frac{1}{P}\sum_{k=j+1}^{N}\frac{I_k^+}{k}$, except for the last position N.

Note that $I_j^- l_j(y,r)$ gives the loss at the position j considering the correct items below position j in the ranking. Therefore, we can use it for a bottom-up (max-violating) inference procedure, which first finds the best candidate item to be put at the lowest position of the rank and proceeds filling the positions in the ascending order.

To use the loss decomposition of Equation 6.2 in Line 5 of Algorithm 2, we start with the last Nth position of the rank and put there the minimum weighted item:

$$\hat{r}^N = \operatorname*{argmin}_{d\in D_i} w_N\mathbf{w}\cdot\phi(q,d).$$

According to the decomposition in Equation 6.2, loss is always 0 at position N. At each of the following steps j: $\hat{r}^{N-j} =$

$$= \operatorname*{argmin}_{d\in D\setminus\{\hat{r}^{N-k}\}_{k=0}^{j-1}} w_{N-j}\mathbf{w}\cdot\phi(q,d) + I_{N-j}^- l_{N-j}(y,\hat{r}).$$

Since, in the loss decomposition, the position-wise components are not independent of the de-

cisions for the other positions, using a greedy procedure does not find a global optimum, but finds a local optimum with respect to the loss exactly. Namely, an item chosen at each position is optimal with respect to to the partial rank constructed at the previous steps of the inference procedure.

6.4 Joint Ranking and Clustering

For an input $\mathbf{x} = (q, D)$, let c be a clustering of its items $d \in D$. Let $\mathbf{y} = (r, c)$ be a new structured output object combining in one ranking r and clustering c, which implies that the items clustered together according to c occupy neighbouring positions in r.

The combined joint feature representation would be:

$$\Phi(x, \mathbf{y}) = \Phi(x, (r, c)) = [\Phi(\mathbf{x}, r), \Phi(\mathbf{x}, c)]^\top.$$

We use a linear scoring model $\mathbf{w}$:

$$\mathbf{w} \cdot \Phi(x, \mathbf{y}) = [\mathbf{w}_r, \mathbf{w}_c][\Phi(\mathbf{x}, r), \Phi(\mathbf{x}, c)]^\top =$$

$$= \mathbf{w}_r \cdot \Phi(\mathbf{x}, r) + \mathbf{w}_c \cdot \Phi(\mathbf{x}, c) = f_{rank}(q, r) + f_{clust}(q, c). \tag{6.3}$$

6.4.1 Structured clustering

For the clustering part of the structure, we follow the approach of Yu and Joachims [2009]. In the current setting, however, we are not provided with the ground truth for clustering c. We only assume that there might be some similar items in the candidate item list D. Thus, we consider correct any c whose clusters contain items d_i with the same label $y(d_i) = y$, i.e., only positive or only negative items. Respectively, any $\mathbf{h}$ containing only correct links is considered correct, plus $\mathbf{h} = \emptyset$. This is in contrast to the original approach of Yu and Joachims, in which the supervision for gold clustering $\mathbf{y}$ is available and fixed, and a correct $\mathbf{h}$ should necessarily reproduce it. As in Equation 2.9, we impose a decomposition of the joint feature vector of an input-output pair into a sum of the feature representations of the edges of $\mathbf{h}$:

$$\Phi(\mathbf{x}, c, \mathbf{h}) = \sum_{(d_k, d_l) \in \mathbf{h}} \phi(d_k, d_l).$$

The loss function optimized by the clustering approach decomposes over the edges of $\mathbf{h}$. In this work, we use a simple loss, that counts the number of edge mistakes:

$$\Delta_{clust}(y, \mathbf{y}) = \Delta_{clust}(y, c, \mathbf{h}) = \sum_{(d_k, d_l) \in \mathbf{h}} \mathbb{1}_{[y(d_k) \neq y(d_l)]}, \tag{6.4}$$

unlike the original loss in Equation 5.4.

6.4.2 Joint inference

The prediction rule to be learned then is

$$\operatorname*{argmax}_{\mathbf{y}\in\mathbf{Y}} \mathbf{w}\cdot\Phi(\mathbf{x},\mathbf{y}) = \operatorname*{argmax}_{(r,\mathbf{h})\in R\times H} \mathbf{w}\cdot\Phi(\mathbf{x},r,c,\mathbf{h}).$$

The ranking and clustering parts of the model decompose in different ways: the former decomposes over the item variables d_i and the latter – over their pairs (d_i, d_j). This complicates the inference with respect to the combined objective in Equation 6.3. In Algorithm 7, we provide a joint inference procedure, which starts from a clustering c composed of all singletons (Line 4) and $\mathbf{h} = \emptyset$. In Line 6, we find the corresponding maximum ranking. Further, the algorithm follows the Kruskal's procedure. It iteratively tries to add edges with positive weight not making loops to the spanning forest $\mathbf{h}$ in decreasing order of their weight, or, in other words, to merge clusters (Lines 16–28). After the merge, the maximum ranking is found for a new clustering, and if this results in the increased overall score (Line 20), the merge persists. When finding the maximum ranking corresponding to a clustering (using function $FindMaxRank$), the preliminarily ranked clusters are repetitively swapped until the highest scoring ordering is found.

Proposition 4 (Joint exact search)**.** *The joint inference procedure by Algorithm 7 finds* $\mathbf{y}$ *maximazing the joint objective (6.3) exactly.*

Proof. The maximum of the ranking part of the joint objective $f_{rank}(q, r)$ is achieved on the two trivial clusterings of the item list D, i.e., on all singletons and on all merged into one cluster since there is the highest freedom in reordering the rank. Adding an edge we can only decrease $f_{rank}(q, r)$. Since we add only edges with positive score starting from one with the highest weight, the proof reduces to the correctness to the Kruskal's algorithm. □

Algorithm 7 Joint inference

Input: example $(\mathbf{x}, y) = (q, D, y)$, $\mathbf{w} = [\mathbf{w}_r, \mathbf{w}_c]$
Initialize $\mathbf{h} \leftarrow \emptyset$
for $i = 1$ **to** $|D|$ **do**
  $c(d_i) \leftarrow \{d_i\}$
end for
$(\mathbf{y}, score) = FindMaxRank(\mathbf{x}, c)$
for $i = 1$ **to** $|D|$ **do**
  for $j = i + 1$ **to** $|D|$ **do**
    $edge\_weight = \mathbf{w}_c \cdot \phi(d_i, d_j)$
    if $edge\_weight > 0$ **then**
      $E = E \cup (d_i, d_j)$
    end if
  end for
end for
$E_s = SortDesc(E)$
for $(u, v) \in E_s$ **do**
  if $c(u) \neq c(v)$ **then**
    Merge$(c(u), c(v))$
    $(\mathbf{y}_{new}, score_{new}) = FindMaxRank(c)$
    if $score < score_{new} + \mathbf{w}_c \cdot \phi(u, v)$ **then**
      $\mathbf{y} = \mathbf{y}_{new}$
      $score = score_{new}$
      $\mathbf{h} = \mathbf{h} \cup (u, v)$
    else
      Unmerge$(c(u), c(v))$
    end if
  end if
end for

Algorithm 8 Joint Latent Structured Perceptron

Input: $X = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{w}_0^r, \mathbf{w}_0^c$, C, T
$\mathbf{w}_{r_0} \leftarrow \mathbf{w}_0^r$; $\mathbf{w}_{c_0} \leftarrow \mathbf{w}_0^c$; $t \leftarrow 0$
repeat
  for $i = 1, ..., n$ **do**
    $\hat{\mathbf{y}} \leftarrow \underset{\mathbf{y} \in \mathbf{Y}(\mathbf{x}_i)}{\operatorname{argmax}} \mathbf{w}_t, \Phi(\mathbf{x}_i \cdot \mathbf{y}) + C \cdot \Delta_{map}(y_i, \mathbf{y})$
    if $\Delta_{map}(y_i, \hat{\mathbf{y}}) > 0$ **then**
      $\mathbf{y}^* \leftarrow \underset{\mathbf{y} \in \mathbf{Y}(\mathbf{x}_i, y_i)}{\operatorname{argmax}} \mathbf{w}_t \cdot \Phi(\mathbf{x}_i, \mathbf{y})$
      $\mathbf{w}_{r_{t+1}} \leftarrow \mathbf{w}_{r_t} + \Phi(\mathbf{x}_i, r(\mathbf{y}^*)) - \Phi(\mathbf{x}_i, r(\hat{\mathbf{y}}))$
    end if
    $\mathbf{w}_{c_{t+1}} \leftarrow \mathbf{w}_{c_t}$
    $t \leftarrow t + 1$
    $\hat{\mathbf{y}} \leftarrow \underset{\mathbf{y} \in \mathbf{Y}}{\operatorname{argmax}} \mathbf{w}_t \cdot \Phi(\mathbf{x}_i, \mathbf{y}) + C \cdot \Delta_{clust}(y_i, \mathbf{y})$
    if $\Delta_{clust}(y_i, \hat{\mathbf{y}}) > 0$ **then**
      $\mathbf{h}^* \leftarrow \underset{\mathbf{h} \in H(\mathbf{x}_i, y_i)}{\operatorname{argmax}} \mathbf{w}_{c_t} \cdot \Phi(\mathbf{x}_i, \mathbf{h})$
      $\mathbf{w}_{c_{t+1}} \leftarrow \mathbf{w}_{c_t} + \Phi(\mathbf{x}_i, c, \mathbf{h}^*) - \Phi(\mathbf{x}_i, c, \mathbf{h}(\hat{\mathbf{y}}))$
    end if
    $\mathbf{w}_{c_t} \leftarrow \vec{0}$
    $t \leftarrow t + 1$
  end for
until $t < 2nT$
$\mathbf{w} \leftarrow \frac{1}{t} \sum_{i=1}^{t} \mathbf{w}_i$
return $\mathbf{w}$

6.4.3 Learning

In the absence of the supervision for the combined ranking-clustering object $\mathbf{y}$, we target the latent structural large-margin objective (see Equation 2.6):

$$\min_{\mathbf{w}} \big[\frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{n} \max_{\mathbf{y} \in \mathbf{Y}(\mathbf{x}_i)} [\Delta(y_i, \mathbf{y}) + \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y})] -$$

$$- C \sum_{i=1}^{n} \max_{\mathbf{y} \in \mathbf{Y}(\mathbf{x}_i, y_i)} \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y})\big]. \tag{6.5}$$

We alternatively optimize the objective in Equation 6.5 with respect to the two loss functions Δ_{map} and Δ_{clust}. We adapt the standard LSP algorithm to alternate between updating the rank-

ing $\mathbf{w}_r$ and clustering $\mathbf{w}_c$ parts of the structural model. The pseudocode of the alternate LSP is depicted in Algorithm 8. In Lines 5–9, we perform an update of the ranking model part $\mathbf{w}_r$. First, we find the max-violating $\hat{\mathbf{y}}$ n Line 5 according to the inference procedure in Algorithm 7. The clue on the adaptation of the inference procedure to involve loss is given in Section 6.3.1. If the ranking part $\hat{r}$ within the combined object $\hat{\mathbf{y}}$ is not correct, we find the current ground truth $\mathbf{y}^*$ corresponding to the current model weights $\mathbf{w}_t$ in Line 7 and update the ranking part weights $\mathbf{w}_r$ in Line 8. The search for $\mathbf{y}^*$ is restricted to the set of all possible correct $\mathbf{y} \in \mathbf{Y}(\mathbf{x}_i, y_i)$, i.e., those whose ranking r complies with the gold label y_i and whose clusters c can be formed only of the items having the same labels, using only correct links (d_k, d_l). This means that in $FindMaxRank()$ clusters of positive items take always top rank positions, and of negative – the bottom ones, and a cluster can be swapped only with another cluster containing items having the same label.

We proceed with finding the max-violating structure with respect to the clustering loss Δ_{clust} in Line 12. Δ_{clust} is straightforwardly injected into the inference procedure of Algorithm 7 as the loss factorizes over the edges (Equation 6.4). Since we do not have ground truth for the clustering part – a correct tree can contain from 0 to the maximum possible number of correct edges – in Line 15, we update only on the wrong edges in $\mathbf{h}(\hat{\mathbf{y}})$ v.s. correct edges $\mathbf{h}^*$ preferred by the clustering model. $argmax$ in Line 14 is found by running the Kruskal's algorithm.

6.5 Experiments

In our experiments, we compare the proposed structured ranking approaches with the classification baselines.

6.5.1 Setup

Data We conduct our experiments on WikiQA dataset for answer sentence selection. We use only examples with at least one correct and at least one incorrect answer candidate Yang et al. [2015] both for training and evaluation. This corresponds to 873 examples for training from the train set, 243 – for testing from test, and 126 – for validation from the development set.

Models We implement the structured ranking approach described in Section 6.3.1, denoted SR. We compare it to the baseline classification approach using the same feature set of question-answer features (q, d_i) – an SVM with polynomial kernels trained using SVM-Light-TK[1]. The

[1] `http://disi.unitn.it/moschitti/Tree-Kernel.htm`

Model	MAP	MRR	P@1
Develoment			
SVM	63.37	64.37	50.00
SR	68.67	69.65	53.28
JSRC	69.21	69.72	54.10
Test			
SVM	54.67	55.90	39.66
SR	62.14	63.46	45.15
JSRC	64.03	65.57	48.52

Table 6.1: Results of re-ranking systems on WikiQA dataset.

joint structural model from Section 6.4 goes by name JSRC – joint structured ranking and clustering. JSRC uses in addition pairwise item-item features $\phi(d_k, d_l)$.

Features In our study, we use the features and setting by Barrón-Cedeño et al. [2016], i.e., cosine similarity over the text pair, the similarity based on the PTK score, longest common substring/subsequence measure, Jaccard similarity, word containment measure, greedy string tiling, ESA similarity based on Explicit Semantic Analysis (ESA), IR score (optional) given by the IR engine, if available.

Parametrization We use the following weighting schema for the ranking structures: $w_j = \frac{1}{j}$. The structural model requires specifying a loss parameter C, whose value was chosen on the development set. The max number of the perceptron epochs T is set to 50. We derived the best number T_{best} with respect to the MAP score on the development set.
The baseline SVM is trained with polynomial kernels of degree 3.

Evaluation metrics We report Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and Precision@1 (P@1).

6.5.2 Experimental results

In Table 6.1, we provide the results of the models on the WikiQA dataset. Compared to the baseline SVM, the results of the structural approach (SR) are more than 5 points better on the development set, and around 7.5 points – on test. It should be noted that the SVM uses kernels, while SR is a simple linear model. For SVM, we also had to limit the number of candidates

to 10 for each query to balance the the classifier. The joint model (JSRC) delivers a further improvement. While, on the development set, we observe a slight increase of half a point in MAP, the improvement of around 2 points suggest the viability of the joint approach.

6.6 Summary

This study broadens the view on re-ranking as a structured prediction problem. We propose to leverage the multiplicity of gold rankings by introducing latent variables. Our approximate max-violating inference with respect to the proposed decomposition of the MAP loss seem to be helpful. However, we plan to verify its exact impact in comparison to the versions optimizing simpler surrogate losses. According to the results of our preliminary experiments, adding the clustering structure has a positive effect on the ranking structural model. As the scale of the answer passage re-ranking on the exploited data allows, in these attempts, so far, we performed the ”quasi”-exhaustive inference with respect to the joint ranking and clustering objective, in order to assess the potential benefit of the joint modelling. In future, we will look for more effective ways for exploring the joint space and extend our approach to other ranking tasks and datasets.

Chapter 7

Summary and future work

Machine learning methods for structured prediction have greatly facilitated learning for the tasks with complex structured output. In a broad sense, every structured output method needs to address the following components:

1. effective encoding of the output structure, imposing necessary structural dependencies,
2. defining efficient techniques for exploring the complex space of output structures,
3. providing mechanisms for the comparison of the output structures to the gold standard.

In this thesis, we explored methods for reinforcing the structured prediction through these channels using the case of supervised clustering, enabled by the structural models of Yu and Joachims [2009] and Fernandes et al. [2012, 2014].

Regarding the first point, our study on the comparison of these structural approaches for supervised clustering in application to coreference resolution in Section 5.2 confirms the trivial consideration of the importance of choosing an appropriate structural modelling for a particular task. In our experiments, the directed tree structures of Fernandes et al. were always better for coreference resolution. We also studied how well the models, under reasonable modifications, can accommodate other tasks. Our adaptation of the model of Fernandes et al. applied in the network domain in Chapter 4 resulted in an effective tool for detecting anomalies based on predicted clusters of traffic transmissions. Here, we assume the notion of order adopted in the original model was quite substantial for representing a continuos traffic flow. In future, we would like to experiment more with the graph model, e.g., identify how long should be a ”history” span to guarantee secure anomaly detection, or whether relying on a subset of referent transmission points, cluster representatives, is beneficial.

Our work also verified the possibility of joint structural models. In Chapter 6, we combined the structural representation of ranking with that of clustering in one structure, which delivered promising preliminary results. A further research in the direction of alternative ways of joint

structured output representations is to be done, addressing also the inference and learning issues. In particular, answering the questions of how to coordinate the respective contributions of each of the substructures in the joint structure score, how to optimize more effectively multiple loss functions, in general, potentially having heterogeneous nature (for now, we merely alternate the model updates, each involving independent inference with respect to one of the two loss functions), etc.

There is always a trade-off between the expressiveness of the structural model and the inference efficiency and exactness. Especially the latter can be subject to compromise in the max-violating case. Our experiments for both coreference resolution in Chapter 5 and ranking in Chapter 6, in which we attempted at optimizing the actual task evaluation measures, including their approximations, showed that allowing inexact inference but with respect to the task measure has better impact on convergence speed and accuracy, especially in hard cases in sense of separability, as compared to the exact inference with respect to surrogate objectives. Nevertheless, there is a room for improvement of the inference procedures for the tasks covered in this thesis: i) the max-violating inference procedure with respect to the approximation of MELA in Section 5.3, which is greedy, ii) the inference of the max-violating ranking structure with respect to MAP, which is only locally exact, i.e., an item selected for each rank position is an exact maximizer with respect to the current partial structure, and iii) the joint ranking-clustering inference procedure in Chapter 6. This is to be closely interlinked with the research for better factorizations of the objectives (their approximations) and appropriate structure modelling.

Finally and most notably, we introduce the idea of learning a complex structural loss from data, which arose in the application of clustering to coreference resolution in connection to the impossibility to optimize directly the task-specific metric due to its high computational complexity. Although, our study here is limited to the particular case of a clustering measure, its outcome indicates a promising research direction encompassing i) learning measures from expert examples, ii) finding effective representations of task-specific loss functions in terms of the structural components, substructure variables, delivering decomposition properties to losses, iii) enabling the loss computation, as in our case, etc.

Our experiments in Section 5.3 reflect the sufficiency of surrogate loss functions following the optimality property, given in Section 3.1, in conditions close to separability and their weakening as far as the separability worsen. The learned loss function mimicking the actual task-specific loss was able to deliver not only better convergence rate in separable case, but also a better separating hyperplane in terms of the task measure in more difficult learning conditions. We realize the necessity of further investigation of the viability of the approach, including, in the coreference resolution case, a strict comparison to the counterpart approaches optimizing other global measures [Clark and Manning, 2016; Le and Titov, 2017].

Regarding the work on learning structural losses, our plans for future work can be summa-

rized as follows:

- Studying the properties of the learned loss approximation, e.g., what is the possible impact of approximating a loss with a function linear or, more generally, smooth in some feature space.
- Verifying, whether the approach generalizes well,
  - empirically, to other structured prediction tasks and domains;
  - theoretically,
    - assessing the convergence speed,
    - finding upper bounds on the training loss and generalization error

    in terms of the approximation accuracy of the loss.

In conclusion, the idea of joint learning of the loss and the model, introduced in Chapter 4, provides a formulation of our framework for learning with a learned loss, an all-in-one solution, whose implementation in Section 5.4 resulted in even better accuracy, being at the same time more efficient. In general, it gives a common solution sketch as it meets the requirements of many real world machine learning scenarios with complex outputs and with non-trivial, and even partially defined, evaluation measures.

Bibliography

Aloise, Daniel; Deshpande, Amit; Hansen, Pierre, and Popat, Preyas. Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, May 2009. ISSN 1573-0565. doi: 10.1007/s10994-009-5103-0. URL https://doi.org/10.1007/s10994-009-5103-0.

Altun, Y.; Tsochantaridis, I., and Hofmann, T. Hidden Markov support vector machines. In *Proceedings of the International Conference on Machine Learning*, 2003.

Archive, The UCI KDD. Kdd cup 1999 data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999. [Online; accessed 09-Nov-2017].

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

Bansal, Nikhil; Blum, Avrim, and Chawla, Shuchi. Correlation clustering. *Machine Learning*, 56(1):89–113, Jul 2004. ISSN 1573-0565. doi: 10.1023/B:MACH.0000033116.57574.95. URL https://doi.org/10.1023/B:MACH.0000033116.57574.95.

Barrón-Cedeño, Alberto; Martino, Giovanni Da San; Joty, Shafiq; Moschitti, Alessandro; Obaidli, Fahad A. Al; Romeo, Salvatore; Tymoshenko, Kateryna, and Uva, Antonio. ConvKN at SemEval-2016 Task 3: Answer and question selection for question answering on Arabic and English fora. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 896–903, San Diego, California, USA, 2016.

Björkelund, Anders and Kuhn, Jonas. 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)*, pages 47–57. Association for Computational Linguistics, 2014. doi: 10.3115/v1/P14-1005. URL http://aclweb.org/anthology/P14-1005.

Björkelund, Anders and Kuhn, Jonas. 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)*, pages 47–57, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P14/P14-1005.

Cai, Jie and Strube, Michael. 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*, SIGDIAL '10, pages 28–36, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 978-1-932432-85-5. URL http://dl.acm.org/citation.cfm?id=1944506.1944511.

Chang, Chih-Chung and Lin, Chih-Jen. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011. ISSN 2157-6904. doi: 10.1145/1961189.1961199. URL http://doi.acm.org/10.1145/1961189.1961199.

Chang, Kai-Wei; Samdani, Rajhans; Rozovskaya, Alla; Rizzolo, Nick; Sammons, Mark, and Roth, Dan. *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, chapter Inference Protocols for Coreference Resolution, pages 40–44. Association for Computational Linguistics, 2011. URL http://aclweb.org/anthology/W11-1904.

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

Chang, Kai-Wei; Samdani, Rajhans, and Roth, Dan. A constrained latent variable model for coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 601–612. Association for Computational Linguistics, 2013. URL http://aclweb.org/anthology/D13-1057.

Chapelle, Olivier; Le, Quoc V., and Smola, Alex. Large margin optimization of ranking measures. In *NIPS Workshop: Machine Learning for Web Search*, 2007.

Chen, Wei; Liu, Tie-Yan; Lan, Yanyan; Ma, Zhiming, and Li, Hang. Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pages 315–323, 2009. URL http://papers.nips.cc/paper/3708-ranking-measures-and-loss-functions-in-learning-to-rank.

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

Clark, Kevin and Manning, Christopher D. 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)*, pages 643–653, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1061.

Collins, Michael. 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, 2002. Association for Computational Linguistics. doi: 10.3115/1118693.1118694. URL http://dx.doi.org/10.3115/1118693.1118694.

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

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

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

Eick, C. F.; Zeidat, N., and Zhao, Z. Supervised clustering - algorithms and benefits. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 774–776, Nov 2004. doi: 10.1109/ICTAI.2004.111.

Farnaaz, Nabila and Jabbar, M.A. Random forest modeling for network intrusion detection system. *Procedia Computer Science*, 89(Supplement C):213 – 217, 2016. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2016.06.047. URL http://www.sciencedirect.com/science/article/pii/S1877050916311127. Twelfth International Conference on Communication Networks, ICCN 2016, August 1921, 2016, Bangalore, India Twelfth International Conference on Data Mining and Warehousing, ICDMW 2016, August 19-21, 2016, Bangalore, India Twelfth International Conference on Image and Signal Processing, ICISP 2016, August 19-21, 2016, Bangalore, India.

Fernandes, Eraldo R. and Brefeld, Ulf. Learning from partially annotated sequences. In Gunopulos, Dimitrios; Hofmann, Thomas; Malerba, Donato, and Vazirgiannis, Michalis, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 407–422, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-23780-5.

Fernandes, Eraldo Rezende; dos Santos, Cícero Nogueira, and Milidiú, Ruy Luiz. 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 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W12-4502.

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

Finley, Thomas and Joachims, Thorsten. Supervised clustering with support vector machines. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 217–224, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102379. URL http://portal.acm.org/citation.cfm?id=1102351.1102379.

Finley, Thomas and Joachims, Thorsten. Supervised k-means clustering. *The NCSTRL collection of Computer Science Technical Reports*, 2008. URL http://hdl.handle.net/1813/11584.

Fung, B. C. M.; Wang, K., and Ester, M. Hierarchical document clustering using frequent itemsets. In *Proc. of the 3rd SIAM International Conference on Data Mining (SDM)*, pages 59–70, San Francisco, CA, May 2003. SIAM.

Haponchyk, Iryna and Moschitti, Alessandro. 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*, pages 203–207, Pisa, Italy, 2014. URL http://www.fileli.unipi.it/projects/clic/proceedings/vol1/CLICIT2014139.pdf.

Joachims, Thorsten. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: 10.1145/775047.775067. URL http://doi.acm.org/10.1145/775047.775067.

Joachims, Thorsten. A support vector method for multivariate performance measures. In Raedt, Luc De and Wrobel, Stefan, editors, *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), August 7-11, 2005, Bonn, Germany*, pages 377–384. ACM Press, New York, NY, USA, 2005. ISBN 1-59593-180-5. URL http://doi.acm.org/10.1145/1102351.1102399.

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

Kummerfeld, K. Jonathan; Berg-Kirkpatrick, Taylor, and Klein, Dan. An empirical analysis of optimization for max-margin nlp. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 273–279. Association for Computational Linguistics, 2015. doi: 10.18653/v1/D15-1032. URL http://aclweb.org/anthology/D15-1032.

Lassalle, Emmanuel and Denis, Pascal. Joint anaphoricity detection and coreference resolution with constrained latent structures. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2274–2280. AAAI Press, 2015. ISBN 0-262-51129-0. URL http://dl.acm.org/citation.cfm?id=2886521.2886637.

Le, Phong and Titov, Ivan. Optimizing differentiable relaxations of coreference evaluation metrics. *Proceedings of CoNLL*, 2017.

Le, Quoc V.; Smola, Alex; Chapelle, Olivier, and Teo, Choon Hui. Direct optimization of ranking measures. *Journal of Machine Learning Research*, 2018. URL https://arxiv.org/pdf/1802.07400.

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

Martschat, Sebastian and Strube, Michael. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:405–418, 2015. ISSN 2307-387X.

McAllester, David A.; Hazan, Tamir, and Keshet, Joseph. Direct loss minimization for structured prediction. In Lafferty, John D.; Williams, Christopher K. I.; Shawe-Taylor, John; Zemel, Richard S., and Culotta, Aron, editors, *NIPS*, pages 1594–1602. Curran Associates, Inc., 2010.

Moosavi, Nafise Sadat and Strube, Michael. 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)*, pages 632–642, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1060.

Peng, Haoruo; Chang, Kai-Wei, and Roth, Dan. A joint framework for coreference resolution and mention head detection. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 12–21. Association for Computational Linguistics, 2015. doi: 10.18653/v1/K15-1002. URL http://aclweb.org/anthology/K15-1002.

Pradhan, Sameer; Moschitti, Alessandro; Xue, Nianwen; Uryupina, Olga, and Zhang, Yuchen. *Joint Conference on EMNLP and CoNLL - Shared Task*, chapter CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes, pages 1–40. Association for Computational Linguistics, 2012. URL http://aclweb.org/anthology/W12-4501.

Raman, M. R. Gauthama; Somu, Nivethitha; Kannan, Kirthivasan; Liscano, Ramiro, and Sriram, V. S. Shankar. An efficient intrusion detection system based on hypergraph - genetic algorithm for parameter optimization and feature selection in support vector machine. *Knowl.-Based Syst.*, 134:1–12, 2017. doi: 10.1016/j.knosys.2017.07.005. URL https://doi.org/10.1016/j.knosys.2017.07.005.

Ranjbar, Mani; Mori, Greg, and Wang, Yang. Optimizing complex loss functions in structured prediction. In *European Conference on Computer Vision*, 2010.

Samdani, Rajhans; Chang, Kai-Wei, and Roth, Dan. A discriminative latent variable model for online clustering. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages I–1–I–9. JMLR.org, 2014. URL http://dl.acm.org/citation.cfm?id=3044805.3044807.

Subba, B.; Biswas, S., and Karmakar, S. A neural network based system for intrusion detection and attack classification. In *2016 Twenty Second National Conference on Communication (NCC)*, pages 1–6, March 2016. doi: 10.1109/NCC.2016.7561088.

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

Tarlow, Daniel and Zemel, Richard S. Structured output learning with high order loss functions. In *Proceedings of the 15th Conference on Artificial Intelligence and Statistics*, 2012.

Tavallaee, Mahbod; Bagheri, Ebrahim; Lu, Wei, and Ghorbani, Ali A. A detailed analysis of the kdd cup 99 data set. In *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, CISDA'09, pages 53–58, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-3763-4. URL http://dl.acm.org/citation.cfm?id=1736481.1736489.

Tsochantaridis, Ioannis; Hofmann, Thomas; Joachims, Thorsten, and Altun, Yasemin. 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, 2004. ACM. ISBN 1-58113-838-5. doi: 10.1145/1015330.1015341. URL http://doi.acm.org/10.1145/1015330.1015341.

Uryupina, Olga; Saha, Sriparna; Ekbal, Asif, and Poesio, Massimo. Multi-metric optimization for coreference: The unitn/i-itp/essex submission to the 2011 conll shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, CONLL Shared Task '11, pages 61–65, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 9781937284084. URL http://dl.acm.org/citation.cfm?id=2132936.2132944.

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

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

Weston, Janson and Blitzer, John. Latent structured ranking. In *Conference on Uncertainty in Artificial Intelligence*, 2012.

Wick, Michael; Rohanimanesh, Khashayar; Bellare, Kedar; Culotta, Aron, and McCallum, Andrew. Samplerank: Training factor graphs with atomic gradients. In Getoor, Lise and Scheffer, Tobias, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 777–784, New York, NY, USA, June 2011. ACM. ISBN 978-1-4503-0619-5.

Wiseman, Sam; Rush, Alexander M., and Shieber, Stuart M. 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, 2016. URL http://aclweb.org/anthology/N/N16/N16-1114.pdf.

Yang, Yi; Yih, Wen-tau, and Meek, Christopher. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL https://aclweb.org/anthology/D/D15/D15-1237.

Yu, Chun-Nam John and Joachims, Thorsten. 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, June 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553523. URL http://doi.acm.org/10.1145/1553374.1553523.

Yue, Yisong; Finley, Thomas; Radlinski, Filip, and Joachims, Thorsten. A support vector method for optimizing average precision. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 271–278, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-597-7. doi: 10.1145/1277741.1277790. URL http://doi.acm.org/10.1145/1277741.1277790.

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

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

Zhao, Ying and Karypis, George. Criterion functions for document clustering: Experiments and analysis. Technical report, 2002.