A Study of Latent Structured Prediction Approaches to Passage Reranking

Iryna Haponchyk
DISI, University of Trento
38123 Povo (TN), Italy
iryna.haponchyk@unitn.it

Alessandro Moschitti[*]
Amazon
Manhattan Beach, CA, USA
amosch@amazon.com

Abstract

The structured output framework provides a helpful tool for learning to rank problems. In this paper, we propose a structured output approach which regards rankings as latent variables. Our approach addresses the complex optimization of Mean Average Precision (MAP) ranking metric. We provide an inference procedure to find the max-violating ranking based on the decomposition of the corresponding loss. The results of our experiments on WikiQA and TREC13 datasets show that our reranking based on structured prediction is a promising research direction.

1 Introduction

The current state-of-the-art learning approaches for answer sentence reranking in question answering (QA) are mostly based on learning pairwise ranking signals or simple binary classification (relevant versus irrelevant labels). Intuitively, global information over a rank should improve the ranker accuracy. Thus, 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). These works employ the structured output learning framework to represent a ranking as a structured object, with respect to which it is possible to directly optimize a ranking measure.

Direct optimization of the target ranking measures is affordable when they 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. In contrast, MAP is rather complex making its treatment harder. Yue et al. (2007) could still find an exact solution to the hinge-loss relaxation of Average Precision (AP) for the structural SVM approach. It is found for the particular case of a combined feature mapping of inputs and structured outputs. Such a mapping accounts for respective orderings of the pairs of candidate items, where one item is relevant and the other is not, without explicitly encoding the order of all the items in the rank. Encoding such order (Chapelle et al., 2007; Weston and Blitzer, 2012), i.e., adding yet more complexity to the structural feature space, might lead to intractability of the previous exact max-violating inference with respect to MAP. Furthermore, the feature representation of the gold standard rankings, which could be many for a given candidate list of items, is not unique anymore.

In this work, we study the effect of using structured ranking representations (Chapelle et al., 2007) within the large-margin structured prediction framework versus direct MAP optimization on the most representative task of QA, i.e., passage reranking. To make two ends meet, we have to tackle the above two issues, i.e., i) intractability of the max-violating inference with respect to MAP, and ii) multiplicity of the ground truths.

Regarding the latter, it should be noted that different rankings can correspond to optimal performance, thus, Chapelle et al. (2007) select one among all possible correct rankings at random to build the ground truth for training. Weston and Blitzer (2012) bypass the necessity of comparison to a complete ranking during training and sample the candidate pairs. In this work, we show how this issue can be seamlessly circumvent using the latent structured prediction formulation.

For optimizing MAP, 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. More

[*]Most of this work was carried out before joining Amazon.

specifically, we provide two structured output approaches optimizing the MAP metric based on Latent Structured Perceptron (LSP) (Sun et al., 2009; Fernandes et al., 2014) and Latent Structural SVM (LSSVM) (Yu and Joachims, 2009) algorithms.

We compare LSP and LSSVM using our MAP optimization strategy on WikiQA (Yang et al., 2015) and TREC13 (Wang et al., 2007) datasets against an SVM classifier and SVM^{map} – the structural approach of Yue et al. (2007). All the models use state-of-the-art traditional feature vectors for the task. Our experiments on WikiQA dataset show a large improvement of our structural approaches over the SVM baseline, i.e., more than 7 absolute points in MAP, MRR and Precision@1.

However, we acknowledge the fact that neural models can produce better representations, which can lead to a superior performance. Thus, to collocate our results in a more general setting, we also carried out experiments using the embeddings of questions and passages, produced by an accurate Convolutional Neural Network (CNN) for passage reranking. In this setting, the structured output models approach the state of the art, confirming the positive impact of our models, which may also be used to train neural networks.

2 Related work

The work related to our approach can be divided in two research areas: (i) structured prediction for reranking problems and (ii) passage reranking in question answering systems.

Structured prediction Seminal work on large-margin structured prediction is by Tsochantaridis et al. (2004), which enables the optimization of multivariate loss functions, exploiting structural dependencies within complex output variables. Chapelle et al. (2007) devise an approach based on the structural SVM, which enables direct optimization of the NDCG ranking measure. Le et al. (2018) facilitate the optimization of a range of ranking measures, within a unified structured output formulation. However, they do not provide any solution for MAP. Weston and Blitzer (2012) optimize a retrieval AUC loss, which decomposes into pairwise decision variables.

The approach most similar to ours is SVM^{map} by Yue et al. (2007), who provide a structural solution for optimizing MAP. They find exactly the max-violating ranking structure with respect to AP within the structural hinge loss formulation. This is done for the case when a ranking is represented by aggregating pairwise outputs between all the relevant and all the irrelevant items in the rank. Our approach is an alternative to this technique, providing an approximate max-violating inference with respect to AP for a more general case of the ranking represenation.

Passage Reranking Most representative *pre-neural networks* work for answer selection/passage reranking is from Wang et al. (2007), who used quasi-synchronous grammar to model relations between a question and a candidate answer with syntactic transformations. Heilman and Smith (2010) and Wang and Manning (2010) applied Tree Edit Distance (TED) to learn the match between question and passage. Yao et al. (2013) applied linear chain CRFs with features derived from TED. Yih et al. (2013) used lexical semantics to build a word-alignment model.

Most recently, Deep Neural Networks (DNNs) have shown to be more competitive. DNN can learn relational patterns between a question (Q) and its passage (P) in a variety of ways, e.g., (i) by using a Q-to-P transformation matrix and simple Q-to-P similarity features (Yu et al., 2014; Severyn and Moschitti, 2015), (ii) by relying on RNN and LSTM architectures (Wang and Nyberg, 2015; Shen et al., 2017), (iii) by employing attention components (Yin et al., 2016; Shen et al., 2017; Wang et al., 2016a), (iv) by decomposing input into similarity and dissimilarity matches (Wang et al., 2016b) or (v) by comparing-aggregating matching results (Wang and Jiang, 2017; Bian et al., 2017).

Since our baselines, SVM and SVM^{map}, as well as our proposed models, LSP and LSSVM, do not apply such transformations, they may perform lower than the state of the art. Thus, we will use the embedding vectors generated by a CNN to show that they can achieve the state-of-the-art accuracy.

3 Structured prediction for ranking

In this section, we provide the task formulation with an introduction of structured prediction algorithms.

3.1 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 list of candidate items corresponding to q_i and

N_i is the number of candidates. $y_i = \{y_i^j : y_i^j \in \{0,1\}\}_{j=1}^{N_i}$ 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 $\mathbf{x}_i$, 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, $r(\mathbf{x}_i) = \{r^j\}_{j=1}^{N_i}$ is a permutation of D_i. In the following, we omit the example index i for simplification of the notation.

3.2 Structured prediction

Generally, structured output approaches aim at linking structured input and output patterns. More formally, in a linear case, such algorithms learn a scoring function $f : X \times Y \to \mathbb{R}$, $f(\mathbf{x}, \mathbf{y}) = \mathbf{w} \cdot \Psi(\mathbf{x}, \mathbf{y})$, where $\Psi(\mathbf{x}, \mathbf{y})$ is a combined feature mapping of input variables X and output variables Y. The predicted structure is derived as $\hat{\mathbf{y}} = \underset{\mathbf{y} \in Y}{\operatorname{argmax}} f(\mathbf{x}, \mathbf{y})$. Online structured prediction approaches, e.g., the structured perceptron by Collins (2002), are based on gradient descent updates of the model on the current predicted structure $\hat{\mathbf{y}}$ against the correct structure $\mathbf{y}^* : \mathbf{w} \leftarrow \mathbf{w} + \Psi(\mathbf{x}, \mathbf{y}^*) - \Psi(\mathbf{x}, \hat{\mathbf{y}})$. The large-margin perceptron variants augment the inference with the structural loss (Fernandes and Brefeld, 2011): $\hat{\mathbf{y}} = \underset{\mathbf{y} \in Y}{\operatorname{argmax}} f(\mathbf{x}, \mathbf{y}) + \Delta(\mathbf{y}^*, \mathbf{y})$. When exhaustive search over Y is impossible, the problem can still be solved if the loss Δ and the scoring function $f(\mathbf{x}, \mathbf{y})$ are decomposable over the subparts of the structure $\mathbf{y}$ and there exists an efficient procedure for finding the argmax using such a decomposition.

3.3 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: $\Psi(\mathbf{x}, r) = \Psi(q, D, r)$, which factorizes over the individual feature representations of items with respect to the query, weighted relatively to the item positions j in the rank:

$$\Psi(\mathbf{x}, r) = \Psi(q, D, r) = \sum_{j=1}^{N} v_j \psi(q, r^j). \quad (1)$$

The typically used weighting schema, v, implies non-increasing weights associated with the positions j: $v_1 \geq v_2 \geq ... \geq v_N \geq 0$, where the 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

$$\underset{r \in R(\mathbf{x})}{\operatorname{argmax}} \mathbf{w} \cdot \Psi(\mathbf{x}, r) \quad (2)$$

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

As the correct ranking r^* for an example $\mathbf{x}$ is often not unique, Chapelle et al. (2007) select one of the correct rankings at random as a gold label during training. This evidently biases the training towards such ground truths.

3.4 Latent structured prediction

The above problem can be alleviated using a latent structured prediction framework. We describe now the general idea of latent variables, and after that we introduce our approach, in which, we implement this idea for ranking tasks.

Latent variables $\mathbf{h}$ are auxiliary structures which are not fully observed in the training data (Yu and Joachims, 2009). The training examples are extended with $\mathbf{h}$ – $(\mathbf{x}, \mathbf{y}, \mathbf{h})$, and the learning is shifted to the space H of latent output structures $\mathbf{h}$. Normally, each $\mathbf{h}$ corresponds to one $\mathbf{y}$, while the opposite is not the case. The problem of multiplicity of the ground truths $\mathbf{h}$ is overcome by finding the best $\mathbf{h}$ explaining the gold $\mathbf{y}^*$:

$$\mathbf{h}^* = \underset{\mathbf{h} \in H(\mathbf{x}, \mathbf{y}^*)}{\operatorname{argmax}} \mathbf{w} \cdot \Psi(\mathbf{x}, \mathbf{h}), \quad (3)$$

using the current model weights $\mathbf{w}$ at each iteration of the training, and $\mathbf{h}^*$ used as gold labels.

4 Our learning approach

In the following, we describe the two classical steps in structured prediction, i.e., learning and inference. Regarding inference, we show our new approximation of MAP.

4.1 Learning

We deal with a non fully observed case as the ground truth ranking labels r we intend to learn are not given in the input data. The case suits perfectly the latent structural formulation (Sec. 3.4), where r can be regarded as latent variables $\mathbf{h}$. Consider the following latent structural large-margin objective (Yu and Joachims, 2009), in terms of the structured ranking variables:

$$\min_{\mathbf{w}} \Big[\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \max_{r\in R(\mathbf{x}_i)} [\Delta(y_i, r)+ \\ + \mathbf{w}\cdot\Psi(\mathbf{x}_i, r)] - C\sum_{i=1}^{n} \max_{r\in R(\mathbf{x}_i, y_i)} \mathbf{w}\cdot\Psi(\mathbf{x}_i, r)\Big], \tag{4}$$

in which the upper bound on the training loss Δ involves (i) finding the max-violating ranking structure, $\hat{r}_i$, over the set $R(\mathbf{x}_i)$ of all possible rankings for the example $\mathbf{x}_i$, under the first max, and (ii) the current ground truth ranking structure, r_i^*, over the set $R(\mathbf{x}_i, y_i)$ of all rankings that comply with the gold label y_i, under the second max.

We adapt the loss-augmented LSP algorithm (Fernandes and Brefeld, 2011) for ranking. LSP is essentially a gradient descent operated on the objective in Eq. 4 with a gradient taken with respect to the example variable. The pseudocode of our adaptation of the algorithm is shown in Alg. 1.

Iterating over the training examples $(\mathbf{x}_i, y_i)$, the algorithm, for each example, first finds the max-violating $\hat{r}_i$ with respect to a ranking loss $\Delta(y_i, r)$, over the set $R(\mathbf{x}_i)$ for the example $\mathbf{x}_i$ (Line 5). Δ can represent any arbitrary ranking loss. In this work, we instantiate Δ with the loss corresponding to the MAP ranking metric. Sec. 4.2 describes the procedure we use here for the max-violating inference with respect to it.

If the max-violating ranking $\hat{r}_i$ is erroneous (Line 6), the algorithm updates the model $\mathbf{w}$. In Line 7, the current ground truth ranking structure – the best correct r_i^* corresponding to the current model weights $\mathbf{w}_t$ – is found. The search here is restricted to the set $R(\mathbf{x}_i, y_i)$ of all correct rankings of the example $\mathbf{x}_i$, i.e., those 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. This step corresponds exactly to imputing latent variables, described by Eq. 3, in the general latent formulation. In Line 8, we update the weights $\mathbf{w}$ using the structural feature representations (defined by Eq. 1) of the two ranking outputs, the current ground truth r_i^* and the max-violating $\hat{r}_i$.

[1]C here is a loss scaling parameter.

Algorithm 1 Latent Structured Perceptron for Ranking

```
1:  Input: $X = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{w}$, $C^1$, $T$
2:  $\mathbf{w}_0 \leftarrow \mathbf{w}$; $t \leftarrow 0$
3:  repeat
4:      for $i = 1, ..., n$ do
5:          $\hat{r}_i \leftarrow \operatorname{argmax}_{r\in R(\mathbf{x}_i)} \mathbf{w}_t \cdot \Psi(\mathbf{x}_i, r) + C \times \Delta(y_i, r)$
6:          if $\Delta(y_i, \hat{r}_i) > 0$ then
7:              $r_i^* \leftarrow \operatorname{argmax}_{r\in R(\mathbf{x}_i, y_i)} \mathbf{w}_t \cdot \Psi(\mathbf{x}_i, r)$
8:              $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \Psi(\mathbf{x}_i, r_i^*) - \Psi(\mathbf{x}_i, \hat{r}_i)$
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 $\mathbf{w}$
```

Likewise, we adapt the Latent Structural SVM (LSSVM) algorithm (Yu and Joachims, 2009) for ranking. We employ also LSSVM in our experiments for its generalization guarantees. The only minor difference from the LSP adaptation is that, in the LSSVM adaptation, we consider only the top items in the joint feature representation in Eq. 1, i.e., $\Psi(\mathbf{x}, r) = \sum_{j=1}^{P} v_j \psi(q, r^j)$, where P is the number of good/positive items in the candidate list D of the example $\mathbf{x}$. This is relevant only at the training phase and only for the updates of the model; for the max-violating inference, still all the items at all N positions participate. By doing so, we help LSSVM to keep the balance between positive and negative items. The test phase inference (respective to Eq. 2), for both LSP and LSSVM, consists only in ordering of all the items by their weight with respect to the model.

Weston and Blitzer (2012) adopt the same structural feature representation of Eq. 1 for ranking, however, in the latent embedding space. They do online SGD updates in correspondence to the positive-negative item pairs, and not on the whole rank, which relates to the way we proceed with LSSVM. In their case, this is due to the impossibility of the global inference (their model is also augmented with the structural component describing item-item interactions) and the scale of the task (they do ranking for recommendation domain). However, they perform a cascade-like inference of the rank which is scattered over the iterations.

4.2 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 $I_j^+ = I_{[y(r^j)=1]}$ and $I_j^- = I_{[y(r^j)\neq 1]}$ the indicator functions that the item at 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 the negative items. We rewrite the AP formula as follows:

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

Here, I_j^+ inside the parentheses becomes 1 in the right-hand side because $I_j^+ * I_j^+ = I_j^+$. Then,

$$\Delta_{ap}(y, r) = 1 - \frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+(\sum_{k=1}^{j-1}I_k^+ + 1) = \frac{1}{P}(P - \sum_{j=1}^{N}\frac{1}{j}I_j^+(\sum_{k=1}^{j-1}I_k^+ + 1)) = \frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+(j - 1 - \sum_{k=1}^{j-1}I_k^+) = \frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+\sum_{k=1}^{j-1}(1 - I_k^+) = \frac{1}{P}\sum_{j=1}^{N}\frac{I_j^+}{j}\sum_{k=1}^{j-1}I_k^- = \frac{1}{P}\sum_{j=1}^{N-1}I_j^-\sum_{k=j+1}^{N}\frac{I_k^+}{k}. \quad (5)$$

According to the last line of Eq. 5, Δ_{ap} decomposes into a sum of quantities $l_j(y, r) = \frac{1}{P}\sum_{k=j+1}^{N}\frac{I_k^+}{k}$ over all positions j with negative items (those activating I_j^-) except for the last position N.

Note that $I_j^- l_j(y, r)$ gives the loss at 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. Specifically, we start with the last Nth position of the rank and put there the minimum weighted item:

$$\hat{r}^N = \operatorname*{argmin}_{d\in D} v_N \mathbf{w}\cdot\psi(q, d). \quad (6)$$

According to the decomposition in Eq. 5, loss is always 0 at position N. At each of the following steps j, $\hat{r}^{N-j} =$

$$= \operatorname*{argmin}_{d\in D\setminus\{\hat{r}^{N-k}\}_{k=0}^{j-1}} v_{N-j}\mathbf{w}\cdot\psi(q, d) + I_{N-j}^- l_{N-j}(y, \hat{r}). \quad (7)$$

Note that the loss part, $I_{N-j}^- l_{N-j}(y, \hat{r})$, in the above formula will be invariant for all the negative items remained in the candidate list, as well as it is for all the positive ones (equal to 0). Thus, $\hat{r}^{N-j}$ is essentially the argmin taken over only two candidate items from $D\setminus\{\hat{r}^{N-k}\}_{k=0}^{j-1}$: one is the positive item with the minimal weight $\mathbf{w}\cdot\psi(q, d)$, and the other, respectively, is the minimal weighted negative one. It is sufficient then to sort independently the positive and the negative items, in the beginning of the whole procedure, in the increasing order of their weights $\mathbf{w}\cdot\psi(q, d)$. Argmin's in equations 6 and 7 are then to be taken over the first items of the two sorted lists, which have not been selected at previous steps. This goes in line with the observation of Yue et al. (2007), that the max-violating ranking output $\hat{r}$ is an interleaving of such two sorted lists, which turns true also for our choice of the structural joint feature representation $\Psi(\mathbf{x}, r)$ in Eq. 1. However, the exact algorithm for max-violating inference of Yue et al. (2007) cannot be applied in our case, since our Ψ, due to distinctive contributions of items at different rank positions scaled with v_j weights, does not satisfy its conditions for an arbitrary choice of v_j.

Since, in our loss decomposition, the position-wise components are not independent of the decisions 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 step is optimal with respect to the partial rank constructed at the previous steps of the inference procedure.

Regarding the running time complexity of our greedy inference procedure, it is bounded by the complexity of the sort operation, $\mathcal{O}(N \log N)$, for the candidate lists of size N. In comparison, the worst case complexity of the exact inference in SVM^{map} by Yue et al. (2007) is $\mathcal{O}(N^2)$. In several cases, as shown by our experiments, doing inexact inference produces also higher MAP values compared to SVM^{map}.

5 Experiments

In our experiments, we compare the proposed structural ranking approach with the classification and structural baselines.

5.1 Setup

The setup reports on data, large margin models, CNN and measures we use in our experiments.

WikiQA 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 857 examples for training from train set, 237 – for testing from test, and 122 – for validation from development (dev.) set.

TREC13 We apply the same evaluation strategy as above on TREC13 dataset (Wang et al., 2007), however, for training, we limit to 10 the number of answer candidates for each question. This gives us 970 training examples, 65 examples for validation, and 68 test examples.

5.1.1 Large margin methods

We implement our structural ranking approach described in Sec. 3.3 using both LSP and LSSVM[2] algorithms, denoting the resulting models LSP-AP and LSSVM-AP, respectively.

We compare the models to an SVM baseline using the same feature set for the pairs, (q, d_i), and a polynomial kernel. We consider also a couple of structural baselines: (i) the standard LSP model (Sun et al., 2009), without loss-augmented inference, which we use in order to explore the impact of optimizing the target evaluation measure. This model still follows Alg. 1, however, the difference is that instead of finding the max-violating $\hat{r}_i$ in Line 5, it finds the following max-scoring:

$$\hat{r}_i \leftarrow \operatorname*{argmax}_{r \in R(\mathbf{x}_i)} \mathbf{w}_t \cdot \Psi(\mathbf{x}_i, r).$$

And another structural baseline is (ii) SVM^{map} [3] (Yue et al., 2007) – a structural SVM approach affording exact max-violating inference with respect to AP.

Features In our study, we use two feature settings: (i) simple textual similarity features – the 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), and (ii) powerful features coming from the embeddings trained with the state-of-the-art neural networks (Tymoshenko et al., 2017).

Parametrization We use the following weighting schema for the ranking structures: $v_j = \frac{1}{j}$, in LSP, LSP-AP, and LSSVM-AP. LSP-AP requires specifying a loss scaling parameter C. In LSSVM and SVM^{map}, C is the standard trade-off between regularization and training error. In all the three models, we select C on dev. set from the values $\{1, 10, 100, 1000, 2000, 5000\}$. The max number of epochs, T, is set to 100, for both LSP and LSP-AP. We apply weight averaging in the LSP models. We derive the best number, T_{best}, with respect to the MAP score on dev. set. The baseline SVM is trained with polynomial kernels of degree 3.

Cross-validation On TREC13, which has a very small test set we apply cross-validation. On WikiQA, we obtain results on the official test set as well as applying cross-validation. We employ disjoint cross-validation as in Tymoshenko et al. (2017). For each approach, we train 5 models on the training set following the traditional 5-fold cross-validation strategy. We split dev. and test sets in 5 subsets each, and use ith dev. subset to tune the parameters of the models trained on the ith fold, and ith test subset – to test them. We report the results averaged over 5 test subsets.

5.1.2 Convolutional Neural Networks

We borrowed the CNN embeddings of questions and answer passages produced by the neural model for passage reranking of Tymoshenko et al. (2017); Severyn and Moschitti (2015)[4].

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

[3] http://projects.yisongyue.com/svmmap/

[4] Several other neural models provide superior accuracy but replicating their results is challenging as shown by

	DEV.		
	MAP	MRR	P@1
SVM	63.37	64.37	50.00
LSP	68.98	69.93	55.74
SVM^{map}	66.15	67.17	51.64
LSP-AP	69.41	69.95	**54.92**
LSSVM-AP	68.67	69.29	**54.92**
LSP-AP*	**69.51**	**70.14**	54.10
LSSVM-AP*	67.48	68.47	51.64

	TEST		
	MAP	MRR	P@1
SVM	54.67	55.90	39.66
LSP	64.35	65.98	48.95
SVM^{map}	61.50	63.10	46.84
LSP-AP	**64.50**	**66.25**	**49.37**
LSSVM-AP	62.39	63.78	46.84
LSP-AP*	63.65	65.48	48.52
LSSVM-AP*	63.74	65.08	47.26

Table 1: Experimental results on WikiQA.

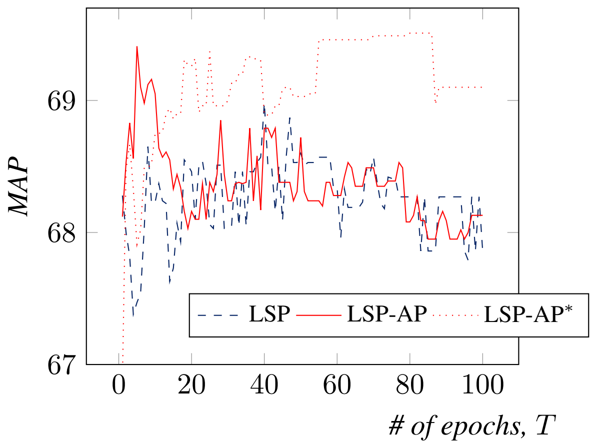

Figure 1: LSP curves on dev. set over the training epochs.

The neural model by Tymoshenko et al. (2017) includes (i) two sentence encoders that map input questions q_i and answer passages d_i^j into fixed size m-dimensional vectors $\phi(q_i)$ and $\phi(d_i^j)$ using a convolutional operation followed by a max pooling layer, and (ii) a feed forward neural network that computes the similarity between the two sentences in the input. The sentence vectors of a question and a passage, $\phi(q_i)$ and $\phi(d_i^j)$, are concatenated together and given in input, at stage (ii), to a standard NN architechture, constituted by a non-linear hidden layer and a sigmoid output layer, which optimizes binary cross-entropy loss.

Note that we use exactly the concatenated question-passage sentence vectors (CNN embeddings) from stage (i) of the above model as features in SVM and the structured output models: $\psi(q, d) = [\phi(q), \phi(d)]$.

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

5.2 Comparative analysis on WikiQA dataset

We first report the results using standard similarity features in all of our models. Then, we show the outcome of our models when fed with embeddings produced by the CNN.

5.2.1 Results with textual similarity features

In Tab. 1, we provide the results of our latent structural approaches optimizing MAP in comparison to SVM, LSP, and SVM^{map} models on WikiQA dataset. LSSVM-AP and LSP-AP are better than the SVM classifier baseline by roughly 8 and 10 points, respectively, in terms of MAP metric on the test set. It should be noted that SVM uses kernels, while LSSVM and LSP are simple linear models. Moreover, for SVM, we also had to limit the number of candidates to 10 for each query to make the positive/negative example rate more balanced.

The baseline LSP model (without augmented loss optimization) performs surprisingly well in this setting. It lacks only 0.15 of a MAP point on test set as compared to the highest scoring LSP-AP model. However, as LSP does not optimize the ranking measure directly, it may result unstable. We verified this hypothesis by exploring the performance of the two LSP models (with our loss and without) on dev. set, plotting the learning curves over the training epochs, T. Fig. 1 shows that the LSP-AP curve is distinctly superior to that of LSP only in the beginning of training, before epoch 17. We further verify this issue in cross-validation experiments in Sec. 5.2.3.

LSSVM-AP outperforms the structural baseline SVM^{map}. The latter targets direct optimization of MAP, and it clearly outperforms the SVM classifier. However, it is worse than the standard perceptron model, LSP, which does not optimize MAP. This suggests a higher appropriateness of the structural feature representation for rankings in Eq. 1. Indeed, it encodes the real positioning of items within a ranking (using positional weights v_i), compared to that used by SVM^{map}, agnostic to it and considering only pairwise relevant/irrelevant relative placements among

Reimers and Gurevych (2017); Crane (2018). Thus, we rely on the model that is easily replicable and achieves competitive results.

the items.

Finally, in the last lines of Dev. and Test subparts of Tab. 1, we report the results of the models denoted with *. In these model variants, we perform exact search for the max-violating ranking $\hat{r}$ with respect to Δ_{AP} over the structural hypothesis space $R(\mathbf{x})$, e.g., in Line 5 of Alg. 1, instead of our approximate inference procedure in Sec. 4.2.

In the current setting, exhaustive search over all possible rankings is reduced to an inference procedure, which inspects all interleavings of the two sorted lists, of positive and negative items, as pointed out in Sec. 4.2. In addition to the sorting complexity, this operation needs to traverse through a subset R' of ranking structures r, $R' \subset R(\mathbf{x})$, where $|R'|$ is estimated by the binomial coefficient $\binom{N}{N-P}$, which, for fixed P, grows polynomially as $\mathcal{O}(N^P)$. Recall that N is a total number of items in the candidate list D, among which P items are positive. In the context of WikiQA dataset, this is affordable, since the candidate lists D_i of the training examples are relatively short.

This way, LSSVM-AP* adds around 1.3 of a MAP point to its result using our approximate inference. LSP-AP*'s best number of epochs on dev., T_{best}, gives a nearly identical result on the test set in terms of MAP. However, its scores are lower compared to those of LSP-AP.

Fig. 1 illustrates a clear advantage of the direct optimization of AP using exact inference, which results in the best curve on dev. for LSP-AP*, stabilizing to its highest values on the interval between epochs 55 and 85. Still such a level of accuracy is nearly reachable by LSP-AP, although actually achieved at a very narrow interval (see the spike around epoch 5), while LSP's curve lies almost always lower. In future, we would like to study the impact of the loss approximation onto the convergence speed of the structural algorithms.

The LSP models in the current setting do not reveal a clear correlation between dev. and test results in Tab. 1, which might (i) signal of insufficient generalization power of LSP, and (ii) suggest that the effect of direct loss optimization can be reached by carefully selecting the epoch's number parameter, T, in the considered feature space. The test set results of LSSVM in terms of MAP instead conform appropriately to the optimized loss function (using approximate versus exact inference) in the large-margin objective in Eq. 4. This should be due to a better generalization capability of the SVM solver.

	DEV.		
	MAP	MRR	P@1
SVM	70.66	70.89	58.20
LSP	67.65	67.70	52.46
SVM^{map}	72.77	73.55	63.11
CNN	71.73	72.04	59.84
LSP-AP	70.67	70.50	56.56
LSSVM-AP	71.27	71.85	59.84
LSP-AP*	**73.67**	**74.55**	**64.75**
LSSVM-AP*	71.88	72.18	60.66
	TEST		
	MAP	MRR	P@1
SVM	65.00	66.70	53.16
LSP	65.28	66.76	52.32
SVM^{map}	66.91	68.63	54.85
CNN	**68.73**	**70.34**	56.12
LSP-AP	64.23	65.56	50.21
LSSVM-AP	67.62	69.24	55.70
LSP-AP*	65.74	67.51	52.32
LSSVM-AP*	67.93	69.73	**56.54**

Table 2: Models on WikiQA trained on CNN embeddings.

5.2.2 Results using neural network embeddings

In these experiments, we used the CNN embeddings, described in Sec. 5.1.2, as features in all of the models. This setting allows us to examine the performance of the models in a more complex and richer feature space, at the level of the state-of-the-art performance. It can also be seen as a coarse way to "neuralize" the structural ranking approaches.

The results of all the models are shown in Tab. 2. As before, we note the relative inconsistency of the performance of the LSP models between dev. and test sets. The non-loss-augmented structured perceptron, LSP, is the weakest of the models on dev. set, while on test it is better than LSP-AP by around 1 point in terms of MAP. It only slightly outperforms now the baseline SVM, which benefited greatly from using the embeddings. Recall, however, that the baseline SVM is trained with kernels. LSP-AP*, reaching considerably higher scores than the rest of the models, including CNN, on dev., is better than LSP by no more than 0.5 of a MAP point.

LSSVM is in general more robust and consistent as with similarity features. Although SVM^{map} outperforms it on dev., LSSVM-AP is better on test in each of the three metrics. LSSVM-AP* with exact inference further improves the results of LSSVM-AP. It outperforms SVM^{map} by more than 1 point in terms of MAP.

It should be noted that the embeddings that we use were trained in a classification setting, thus, giving an additional advantage to the classification models, e.g., the relative improvement of the baseline SVM when passing to embeddings is the highest among the models. Nonetheless, LSSVM approaches closest of all to CNN, with the variant of the model with exact search showing P@1 superior to that of CNN. This suggests that the structural ranking approaches are of decent capacity, and that the optimal solution lies in regions feasible to the structural linear model, considering also the high results of LSP-AP* on dev. set. This is despite the fact that, in contrast to CNN, which trains on the whole training set, we omit examples with only negative and only positive candidates (Sec. 5.1). Exploiting the information from such examples (subject to additional enhancement of our approach, as it would currently perform a zero update on them in Line 8 of Alg. 1 due to equal max-violated and ground truth rankings) might advance the performance. Thus, good features make our models competitive with the state of the art.

5.2.3 Cross-validation experiments

In Tab. 3, we repeat the main experiments in the cross-validation setting, using the similarity features. On average, LSSVM-AP outperforms SVM^{map} in terms of MAP and MRR, as in the standard setting, however, having relatively higher variance across the folds. The LSP models sustain their superiority using the similarity features also in cross-validation, with LSP-AP scoring the best across the models and with the least variance.

5.3 Experiments on TREC13 dataset

The results of our cross-validation experiments on TREC13 are depicted in Tab. 4. LSP-AP slightly improves over the baseline models in terms of MAP. The baseline LSP this time deviates the least over the folds and reaches better P@1 among all the models. LSSVM-AP instead underperforms in this experiment, which might be for the reason of shortage of the data for validation.

It is also true that in this work, by fixing the weighting schema v, we limited our study to one particular case of a structural ranking representation. However, finding an appropriate structural feature space, e.g., to the extent enabled by tuning the positional weights v_j for the particular application, can be potentially beneficial.

	MAP	MRR	P@1
SVM	55.20±3.60	56.68±3.33	39.69±5.27
LSP	64.27±2.52	65.63±2.62	48.10±4.32
SVM^{map}	60.88±2.57	62.27±2.78	45.18±4.19
LSP-AP	**64.47**±2.48	**65.88**±2.17	**48.95**±4.19
LSSVM-AP	62.00±4.97	63.39±4.59	44.79±8.06

Table 3: Cross-validation results on WikiQA.

	MAP	MRR	P@1
SVM	71.68±4.19	82.41±3.40	70.83±5.73
LSP	71.67±2.54	82.54±1.94	**72.08**±4.28
SVM^{map}	71.85±3.69	81.97±2.91	69.17±5.49
LSP-AP	**72.53**±5.29	**82.85**±3.84	71.01±8.57
LSSVM-AP	70.96±4.74	80.97±2.70	69.17±5.49

Table 4: Cross-validation results on TREC13.

6 Conclusions

In this paper, we proposed new structured prediction algorithms for ranking problems. In particular, we designed (i) a new loss function that leads to the direct optimization of MAP; and (ii) two new algorithms, based on LSP and LSSVM solvers, to optimize it. The comparative results on the benchmarks for passage reranking, WikiQA and TREC13, demonstrate an improvement of LSP-AP over the standard SVM classifier, which is particularly large in the case of WikiQA. LSP without any loss augmentation can achieve good performance, as well, subject to accurate tuning of the epoch number parameter. In the same setting, LSSVM-AP is comparable to SVM^{map} baseline.

Finally, we used CNN embeddings as more expressive features in our models. We found that (i) linear models can benefit from them; (ii) LSSVM-AP is more robust than the LSP models to the use of a complex representation; and (iii) traditional max margin methods may not be on par with neural networks on tasks, such as WikiQA, however providing them with right features (embeddings) can make them approach the performance of neural models. This suggests an interesting research line on using our structural models and loss function optimizing MAP in neural models.

Acknowledgments

We would like to thank the anonymous reviewers for their competent and useful suggestions. Many thanks to Kateryna Tymoshenko and Daniele Bonadiman for kindly providing us with the CNN embeddings for our experiments.

References

Alberto Barrón-Cedeño, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad A. Al Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. 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.

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A Compare-Aggregate Model with Dynamic-Clip Attention for Answer Selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1987–1990, New York, NY, USA. ACM.

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

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matt Crane. 2018. Questionable answers in question answering research: Reproducibility and variability of published results. *TACL*, 6:241–252.

Eraldo R. Fernandes and Ulf Brefeld. 2011. Learning from partially annotated sequences. In *Machine Learning and Knowledge Discovery in Databases*, pages 407–422, Berlin, Heidelberg. Springer Berlin Heidelberg.

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

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *NAACL*.

Quoc V. Le, Alex Smola, Olivier Chapelle, and Choon Hui Teo. 2018. Optimization of ranking measures. *Journal of Machine Learning Research*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348. Association for Computational Linguistics.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 373–382, New York, NY, USA. ACM.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-Weighted Alignment Network for Sentence Pair Modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1179–1189. Association for Computational Linguistics.

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

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

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2017. Ranking kernels for structures and embeddings: A hybrid preference and classification model. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 897–902. Association for Computational Linguistics.

Bingning Wang, Kang Liu, and Jun Zhao. 2016a. Inner Attention based Recurrent Neural Networks for Answer Selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1288–1297.

Di Wang and Eric Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. *Proceedings ACL 2015*, pages 707–712.

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *ACL*.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. *ICLR*.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016b. Sentence Similarity Learning by Lexical Decomposition and Composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1340–1349, Osaka, Japan. The COLING 2016 Organizing Committee.

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

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. 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. Association for Computational Linguistics.

Xuchen Yao, Benjamin Van Durme, Peter Clark, and Chris Callison-Burch. 2013. Answer extraction as sequence tagging with tree edit distance. In *NAACL*.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1744–1753, Sofia, Bulgaria. Association for Computational Linguistics.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

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

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. *NIPS Deep Learning and Representation Learning Workshop*.

Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. 2007. 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. ACM.