Perturbation Based Learning for Structured NLP Tasks with Application to Dependency Parsing

Amichay Doitch * **Ram Yazdi** * **Tamir Hazan** **Roi Reichart**
Faculty of Industrial Engineering and Management, Technion, IIT
`{amichayd@campus|sram@campus|tamir.hazan|roiri}.technion.ac.il`

Abstract

The best solution of structured prediction models in NLP is often inaccurate due to limited expressive power of the model or to non-exact parameter estimation. One way to mitigate this problem is sampling candidate solutions from the model's solution space, reasoning that effective exploration of this space should yield high quality solutions. Unfortunately, sampling is often computationally hard and many works hence back-off to sub-optimal strategies such as extraction of the best scoring solutions of the model, which are not as diverse as sampled solutions. In this paper we propose a perturbation-based approach where sampling from a probabilistic model is computationally efficient. We present a learning algorithm for the variance of the perturbations, and empirically demonstrate its importance. Moreover, while finding the argmax in our model is intractable, we propose an efficient and effective approximation. We apply our framework to cross-lingual dependency parsing across 72 corpora from 42 languages and to lightly supervised dependency parsing across 13 corpora from 12 languages, and demonstrate strong results in terms of both the quality of the entire solution list and of the final solution.[1]

1 Introduction

Structured prediction problems are ubiquitous in Natural Language Processing (NLP) (Smith, 2011). While in most cases models for such problems are designed to predict the highest quality structure of the input example (e.g. a sentence or a document), in many cases a diverse list of meaningful structures is of fundamental importance.

This can stem from several reasons. First, it can be a defining property of the task. For example, in extractive summarization (Nenkova and McKeown, 2011) good summaries are those that consist of a high quality and diverse list of sentences extracted from the text. In other cases the members of the solution list are exploited when solving an end goal application. For example, dependency forests were used in order to improve machine translation (Tu et al., 2010; Ma et al., 2018) and sentiment analysis (Tu et al., 2012).

In yet other cases it is a first step towards learning a high quality structure, that cannot be learned by the model through standard argmax inference. For example, in the well-studied reranking setup (Collins, 2002; Collins and Koo, 2005; Charniak and Johnson, 2005; Son et al., 2012; Kalchbrenner and Blunsom, 2013), a K-best list of solutions is first extracted from a baseline learner, that typically has a limited feature space, and is then transferred to another feature-rich model that chooses the best solution from this list. Other examples include bagging (Breiman, 1996; Sun and Wan, 2013) and boosting (Bawden and Crabbé, 2016) as well as other ensemble methods (Surdeanu and Manning, 2010; Täckström et al., 2013; Kuncoro et al., 2016) that are often applied when the data available for model training is limited, in cases where exact argmax inference in the model is indefeasible or when training is not deterministic. In such cases, an ensemble of approximated solutions is fed into another model that extracts a final high quality solution.

Unfortunately, both alternatives suffer from inherent limitations. K-best lists can be extracted by extensions of the argmax inference algorithm for many models: the K-best Viterbi algorithm (Golod, 2009) for Hidden Markov Models (Rabiner, 1989) and Conditional Random Fields (Lafferty et al., 2001), K-best Maximum Spanning Tree (MST) algorithms for graph-based depen-

* Both authors equally contributed to this work.
[1]Our code is at: https://github.com/ramyazdi/perturbations

dency parsing (Camerini et al., 1980; Hall, 2007) etc. However, the members of K-best lists are typically quite similar to each other and do not substantially deviate from the argmax solution of the model.[2] Ensemble techniques, in contrast, are often designed to encourage diversity of the K-list members, but they require the training of multiple models (often one model per solution in the K-list) which is prohibitive for large K values.

In this work we propose a new method for learning K-lists from machine learning models, focusing on structured prediction models in NLP. Our method is based on the MAP-perturbations model (Hazan et al., 2016). A particularly appealing property of the perturbations framework is that it supports computationally tractable sampling from the perturbated model, although this comes at the cost of the argmax operation often being intractable . This property allows us to sample high quality and diverse K-lists of solutions, while training only the base (non-perturbated) learner and a smooth noise function. We propose a novel algorithm that automatically learns the noise parameter of the perturbation model and show the efficacy of this approach in generating high quality K-lists (§ 2). To overcome the intractability of the argmax operation we employ an approximation and experimentally demonstrate its efficacy.

Particularly, we introduce a Gibbs-perturbation model: a model that augments a given machine learning model with an additive or multiplicative Gaussian noise function (Keshet et al., 2011; Hazan et al., 2013). In order to approximate the argmax of the perturbated model we employ a max over marginals (MOM) procedure over the K-list members. We learn the variance of the Gaussian noise function such that the final solution distilled from the K-list is as close to the gold standard solution as possible. To the best of our knowledge, the final solution distillation method and the variance learning algorithm are novel in the context of perturbation-based learning.

To evaluate our framework, we consider two dependency parsing setups: cross-language transfer and lightly supervised training. We focus on these tasks because they are prominent NLP challenges where the model (the non-perturbated dependency parser) is a good fit to the task and data, as indicated by the high quality trees generated in monolingual setups with abundance of in-domain training data, but the training setup makes parameter estimation challenging. Hence, the argmax solution of the model is often not the highest quality one. In such cases it is likely that a diverse list of high quality solutions will be valuable.

Particularly, we experiment with the Universal Dependencies (UD) Treebanks (Nivre et al., 2016; McDonald et al., 2013). For cross-language parser transfer we consider 72 corpora from 42 languages. We train a perturbated delexicalized parser for each target language. The non-perturbated parser is first trained on data from all languages except from the target language and then we learn the variance of the noise distribution on additional data from those languages. Finally, we employ the trained perturbated parser K times to the target language test set, perturbating the parameters of the base parser using noise sampled from the trained noise distribution. The final solution is extracted from this K-list by the MOM algorithm. The experiments in the lightly supervised setup are similar, except that we consider 13 UD corpora (written in 12 languages) which have limited training data. This setup is mono-lingual, we train and test on data from the same corpus.

Our results demonstrate the quality of the K-lists generated by our algorithm and of the tree returned by the MOM procedure. We compare our lists and final solution to those of a variety of alternative algorithms for K-list generation, including the K-best variant of the parser's argmax inference algorithm, and demonstrate substantial gains. Finally, even though we integrate our method into a linear parser (Huang and Sagae, 2010), our modified parser outperforms a state-of-the-art (non-perturbated) BiLSTM parser (Kiperwasser and Goldberg, 2016) on our tasks.

2 K-lists in NLP

Structured models in NLP Many NLP tasks, particularly tagging and parsing, involve the inference of a high-dimensional discrete structure $y = (y_1, ..., y_m)$. For example, in part-of-speech (POS) tagging of an n-word input sentence, each y_i variable corresponds to an input word (and hence $m = n$), and is assigned a value in $\{1, . . . , P\}$ where P is the number of POS tags. In dependency parsing, a graph $G = (V, E)$ is de-

[2]Many machine translation (MT) works aimed to generate diverse K-lists of translated sentences (e.g. (Macherey et al., 2008; Gimpel et al., 2013; Li and Jurafsky, 2016)). However, these methods are specific to MT, while we focus on a general framework for structured prediction in NLP.

fined over an n-word input sentence such that each vertex corresponds to a word in the input sentence ($|V| = n$) and each arc corresponds to an ordered word pair ($|E| = m = n^2$). In the structured model, each ordered pair of words in the input sentence is assigned a variable y_i, and the resulting parse tree is a vector $(y_1, ..., y_m) \in \{0, 1\}^m$ that forms a spanning tree in the graph G. For every spanning tree $y_e = 1$ if the arc $e \in E$ is in the spanning tree and $y_e = 0$ otherwise. In what follows, we proceed with the dependency parsing notation although our ideas are equally relevant to any task defined over discrete structures.[3]

The common practice in structured prediction is that structures are scored by a function that assigns favorable structures with high scores and unfavorable ones with low scores. The number of structures ($|\mathcal{T}|$) is often exponential in m, as in our running dependency parsing example. Hence, in order to avoid exponential complexity, the scoring function has to factorize. In our running example this is done through:

$$\theta(y_1, ..., y_m) = \sum_{e \in E} \theta_e y_e \qquad (1)$$

The standard approach is to train the model (estimate the θ parameters of the scoring function) so that the highest scoring configuration (namely $y^* = \arg\max_{y \in \mathcal{T}} \theta(y)$) is as similar as possible to the human generated ("gold") structure. For dependency parsing, this is equivalent to finding the maximal spanning tree of the graph G.

Prediction with K-lists Unfortunately, oftentimes the highest scoring structure is not the best one. This may happen in cases the model is not expressive enough, e.g. in first-order dependency parsing where only m local potentials (θ_e) are used to score exponentially many structures. This may also happen in cases where the values of the potential functions are inaccurate, as learning inherently has both statistical and variational errors.

A popular solution to this problem is exploiting the power of lists of structures. In the first stage of this framework, the list members are extracted and in the second stage, the final solution is extracted from this list - either by selecting one list member, or by distilling a new solution based on the statistics of the list members.

[3]To be more precise, our notation is that of the graph-based first-order dependency parsing problem, where weights are defined over individual candidate dependency arcs.

Ideally, such a list should be high quality and diverse, in order to explore candidate structures that add information over the structure returned by the argmax inference problem. Yet, the prominent approach in past research constructs a list of the K best solutions according to the scoring function (Equation 1). On the positive side, this approach is computationally feasible as the argmax inference algorithms of prominent structured NLP models can be efficiently extended to find the top scoring K structures (§ 1). However, in practice the top scoring K structures are similar to the top scoring structure (see our analysis in § 6), and important parts of the solution space remain unexplored.[4]

This calls for another approach that explores more diverse parts of the solution space. The approach we take here is based on sampling from probabilistic models.

Sampling-based K-lists Sampling is a possible solution to the diversity problem. In practice, many sampling algorithms require that the structured model will be defined as a probabilistic model. It is natural to impose a probabilistic interpretation of the model described in Equation 1. To do that, a posterior distribution over all structures (a.k.a the Gibbs distribution) is realized from the scoring function:

$$p_\theta(y_1, ..., y_m) \;\propto\; \exp\Big(\sum_{e \in E} \theta_e y_e\Big) \qquad (2)$$

The highest scoring structure under this probabilistic model is called the maximum a-posteriori (MAP) assignment, and is identical to the top scoring function from Equation 1:

$$\begin{aligned} y_1^*, ..., y_m^* &= \arg \max_{y_1, \ldots, y_m \in \mathcal{T}} p_\theta(y_1, ..., y_m) \qquad (3) \\ &= \arg \max_{y_1, \ldots, y_m \in \mathcal{T}} \sum_{e \in E} \theta_e y_e \end{aligned}$$

Likewise, the top K-list of this model – consisting of the K most probable structures of the Gibbs distribution – is also identical to that of the unnormalized model. As noted above, these structures are likely to be of high quality but also quite similar to each other.

The natural alternative that probabilistic models make possible is to sample from the Gibbs distribution instead. Such a strategy is likely to detect

[4]As noted in § 1, the other prominent approach, based on ensemble methods, is computationally demanding for high K value, as K different models have to be trained. In the rest of the paper we hence do not focus on this approach.

high quality structures even if they are not very similar to the best scoring solution, particularly in cases where the estimated model parameters do not fit well the test data. A final tree distilled from such a candidate list is more likely to be of higher quality than the list distilled from the list of the top scoring K structures, due to the better representation of the solution space.

Unfortunately, this approach comes with a caveat: sampling a structure from the Gibbs distribution is often slower than finding the MAP assignment (Goldberg and Jerrum, 2007; Sontag et al., 2008). In our running example, the sampling of first order graph-based dependency parsing depends on the mean hitting time of a random walk in a graph (Wilson, 1996; Zhang et al., 2014), which is slower than finding the maximum spanning tree of the same graph.

Perturbation-based K-lists Perturbation models define probability distributions over high-dimensional discrete structures for which sampling is as fast as solving the MAP problem of a base, non-perturbated, model (Papandreou and Yuille, 2011; Tarlow et al., 2012; Hazan and Jaakkola, 2012; Maddison et al., 2014). In our setting, perturbation models let us sample a spanning tree as fast as finding a highest scoring spanning tree of a base parser. In this setting, we can draw samples from the perturbated model by perturbing the potential functions of the base model and solving the resulting MAP problem. The MAP-perturbation approach samples random variables $\gamma_1, ..., \gamma_m$ from a posterior distribution around the base model weights $\theta_1, ..., \theta_m$ and solves the randomly perturbed argmax problem:[5]

$$y^\gamma = \arg\max_{y \in \mathcal{T}} \left\{ \sum_{e \in E} \gamma_e y_e \right\} \tag{4}$$

The posterior distribution around the model weights $q_\theta(\gamma)$ is defined such that it is centered around the model weights θ, namely, $\mathbb{E}_{\gamma \sim q_\theta}[\gamma] = \theta$. For example, $q_\theta(\gamma)$ can be a Gaussian probability density function:

$$q_\theta(\gamma) = \prod_e \frac{1}{\sqrt{2\pi}} e^{\frac{(\gamma_e - \theta_e)^2}{2}}.$$

[5]In practice, feature-based models are a bit more complicated. For example, linear models typically define $\theta_e = W \cdot f_e$ and then the number of random noise variables in the MAP-perturbation approach is $|W|$. For simplicity of presentation we describe here a model with one parameter per candidate edge (θ_e) and m noise variables.

For now we assume that the variance of the posterior $q_\theta(\gamma)$ is 1 and defer its learning to § 3. Perturbation models measure the probability a structure is of maximal score, when considering all perturbations:

$$p_\gamma(y_1, ..., y_m) \quad = \quad P_{\gamma \sim q_\theta}[y^\gamma = y] \tag{5}$$

A particular appealing property of Gibbs models is that in many cases the most likely structure can be computed or approximated efficiently using dynamic programming or efficient optimization techniques (Koller et al., 2009; Wainwright and Jordan, 2008). For example, finding the most likely dependency parse can be done by finding the maximum spanning tree of a graph (McDonald et al., 2005). In this work we want to enjoy the best of both worlds, exploiting the capability of MAP-perturbation models to sample by solving the MAP problem of the base model, while building on the efficient MAP approximation in Gibbs models. We do that by composing a perturbation model on top of a Gibbs model. This construction allows us to effectively sample high quality and diverse K-lists from MAP-perturbation models, and distill a high quality final structure.

3 Effective Sampling and Learning with MAP-Perturbation Models

A major practical issue when implementing perturbation models is the magnitude of the perturbation variables γ, or their variance. It is easy to see that the variance of these variables greatly influences the quality of the resulting probability model. If this variance is too high, the perturbation noise can easily shadow the signal learned from data , i.e. $\sum_e \gamma_e y_e \gg \sum_e \theta_e y_e$ with non-negligible probability, so the max-perturbation value becomes meaningless. Therefore, in this work we learn the variance of the perturbation posterior. For example, for a Gaussian noise $\gamma \sim N(0, \sigma_e^2)$ added to the Gibbs model parameters $\theta = [\theta_1, \ldots, \theta_m]$, the variance is introduced as

$$\text{(additive) } q_{\theta,\sigma}(\gamma) = \textstyle\prod_e \frac{1}{\sqrt{2\pi}\sigma_e} e^{\frac{(\gamma_e - \theta_e)^2}{2\sigma_e^2}}. \tag{6}$$

Our model is more flexible, and allows other types of noise. For example, we can assume a Gaussian multiplicative noise $\gamma \sim N(1, \sigma_e^2)$ to get

$$\text{(multiplicative) } q_{\theta,\sigma}(\gamma) = \prod_e \frac{1}{\sqrt{2\pi}\sigma_e} e^{\frac{(\gamma_e - \theta_e)^2}{2\theta_e^2 \sigma_e^2}}. \tag{7}$$

We divide this section to two. We first discuss our approach to variance learning in perturbation models. Then, we detail our recipe for learning with perturbation-based K-lists, so that each test example is eventually assigned a single structure.

Learning the variance of the perturbation distribution Given a training set $S = \{(x_i, y_i)\}_{i=1}^N$ consisting of examples (x_i) and the structures with which they are labeled (y_i), we learn the variance with respect to the oracle loss $oracle_K()$. This loss penalizes the perturbation parameters $(\gamma_1, \dots, \gamma_m)$ according to the difference between the final structure extracted from the K-list of each example x_i and the gold tree of that example, y_i. In our running example, dependency parsing, it is straight forward to define this loss as:

$$oracle_K(\{\gamma^j\}_{j=1}^K, x_i, y_i) = HamDist(MOM(\{\gamma^j\}_{j=1}^K, x_i), y_i) \tag{8}$$

where $\gamma^j = (\gamma_1^j, \dots, \gamma_m^j)$ are the perturbation parameters of the i-th example, MOM is the max-over-marginals algorithm that distills a final tree from the K sampled trees (§ 4), and HamDist is the hamming distance between the MOM tree and the gold tree y_i:

$$HamDist(y_m, y_i) = \sum_{j=1}^{n} \begin{cases} 1 & \text{if } h_{y_m}(j) = h_{y_i}(j) \\ 0 & \text{Otherwise} \end{cases} \tag{9}$$

where n is the number of words in the sentence, and $h_y(j)$ is the head of the j-th word in y.[6]

We next define the expected empirical loss (EEL) with respect to the variance of the perturbation distribution:

$$EEL(\sigma, S) = \frac{1}{N} \sum_{(x_i,y_i)\in S} E_{\gamma^1,\dots,\gamma^K \sim q_{\theta,\sigma}}[oracle_K(\{\gamma^j\}_{j=1}^K, x_i, y_i)] \tag{10}$$

And the optimal σ will minimize this loss:

$$\sigma^* = \min_{\sigma} EEL(\sigma, S) \tag{11}$$

Whenever $q_{\theta,\sigma}(\gamma)$, the perturbation probability density function (pdf), is smooth in σ, the EEL is the integral of a smooth function (the pdf $q_{\theta,\sigma}(\gamma)$) and the non-smooth oracle function. In the following we prove that this integral is a smooth function of σ and therefore the optimal variance can be learned from data by using a gradient method to solve the problem in Equation 11.

Claim 1. If the probability density function $q_{\theta,\sigma}(\gamma)$ is smooth and its gradient is integrable, i.e., $\int |\partial q_{\theta,\sigma}(\gamma^j)/\partial\sigma_e| d\gamma^j < \infty$ then the gradient of the EEL function with respect to σ_e as computed on $(x_i, y_i) \in S$ takes the form:

$$\frac{\partial EEL(\sigma)}{\partial \sigma_e} = \sum_{(x_i,y_i)\in S} \int \frac{\partial q_{\theta,\sigma}(\gamma^j)}{\partial \sigma_e} oracle_K(\{\gamma^j\}_{j=1}^K, x_i, y_i) d\gamma^j \tag{12}$$

Proof. The expectation

$$E_{\gamma^1,\dots,\gamma^K \sim q_{\theta,\sigma}}[oracle_K(\{\gamma^j\}_{j=1}^K, x_i, y_i)]$$

is the integral

$$\int \prod_{j=1}^{K} q_{\theta,\sigma}(\gamma^j) f(\{\gamma^j\}_{j=1}^K) d\gamma^1 \cdots d\gamma^K,$$

where $f(\{\gamma^j\}_{j=1}^K) = oracle_K(\{\gamma^j\}_{j=1}^K, x_i, y_i)$ is a non-differentiable function. Notably, the function $f(\{\gamma^j\}_{j=1}^K)$ is independent of σ and therefore its non-differentiability does not affect the differentiability of $EEL(\sigma)$. Moreover, $f(\{\gamma^j\}_{j=1}^K) \leq N$ for some constant N, therefore the function $q_{\theta,\sigma}(\gamma^j) f(\{\gamma^j\}_{j=1}^K)$ is bounded by the integrable function $N q_{\theta,\sigma}(\gamma^j)$ and its derivative with respect to σ is bounded by the function $N|\partial q_{\theta,\sigma}(\gamma^j)/\partial\sigma_e|$. Following Theorem 2.27 by Folland (1999) the function $EEL(\sigma)$ is differentiable and its gradient is attained by differentiating under the integral. $\square$

The above claim shows how to learn the optimal variance of the random perturbation variables with a gradient method. Note that $oracle_K$ and hence also $EEL(\sigma, S)$ are defined with respect to a given K-list size (K). K is a hyper-parameter that can be estimated using, e.g., a grid-search for optimal value using development data. Our experiments are with: $K = 10, 100, 200$ (§ 5).

Once σ and K are determined, we can generate meaningful samples, i.e., the perturbation value $\gamma_e y_e$ will not shadow the data signal $\theta_e y_e$. We are now ready to provide a learning process with perturbation-based K-lists.

[6] The hamming distance is equivalent to the Unlabeled Attachment Score (UAS) between the trees.

Learning with perturbation-based K-lists Our goal is to train a model so that it can eventually output a single high-quality structure, y^*, hopefully of a higher quality than the output (MAP) of the Gibbs (base) model. Since joint learning of θ (the Gibbs model parameters) and σ (the variance of the perturbation distribution) is intractable, we first learn θ and then σ.

We assume two training sets: $S = \{(x_i, y_i)\}_{i=1}^N$ and $S' = \{(x'_i, y'_i)\}_{i=1}^{N'}$. Our training recipe is as follows:

1. Learn the parameters θ of the Gibbs (base) model with the training set S.

2. Learn the parameter σ and the hyper-parameter K with the training set S' by minimizing $EEL(\sigma, S')$ while keeping the θ parameters learned at step (1) fixed.

The test-time recipe for the i-th test example is:

1. Sample K values of the perturbation variables: $\{\gamma^j \sim q_{\theta,\sigma} | j \in \{1, \dots, K\}\}$.

2. for $j \in \{1, \dots, K\}$ find y^{γ^j} according to Equation 4.

3. Extract the final structure y^* from $\{y^{\gamma^j}\}_{j=1}^K$.

The only missing piece is the method for extracting y^* from $\{y^{\gamma^j}\}_{j=1}^K$. Note that this method is employed both at step (2) of the training recipe (as it is part of the definition of $EEL(\sigma, S')$) and at step (3) of the test-time recipe. In the next section we describe an approximation algorithm for this problem.

4 Max Over Marginals (MOM) Inference

Our oracle loss considers the hamming distance of max-over-marginals (MOM). For this aim, let us consider the single variable (candidate edge) marginal probabilities of the Gibbs-perturbation model:

$$\mu_e = P_\gamma[y_e^\gamma = 1] \quad (13)$$

We then define the approximated argmax inference in the Gibbs-perturbation model as predicting the best spanning tree with respect to the log of these marginals:

$$y^* = \arg\max_{y \in \mathcal{T}} \Big\{ \sum_{e \in E} y_e \log \mu_e \Big\} \quad (14)$$

Notice that for first order parsing, our running example in this paper, this approach is essentially identical to the inference algorithm of Kuncoro et al. (2016), which was aimed at distilling a final solution from an ensemble of parsers. However, this MOM approach can naturally be extended beyond single variable potentials. For example, we can consider variable pair potentials or potentials over variable triplets and perform exact (Koo and Collins, 2010) or approximated (Martins et al., 2013; Tchernowitz et al., 2016) inference for second and third order problems. Here, for simplicity, we focus on single variable potentials and solve the resulting MOM problem directly with an exact MST algorithm.

In what follows we first show that the MOM approach – recovering the best spanning tree according to the log-marginals of one Gibbs-perturbation model – can be interpreted as a MAP approach over marginal probabilities of a continuous-discrete Gibbs model. We then discuss how we estimate the marginal probabilities μ_e (Equation 13).

MOM as MAP of a Continuous-discrete Gibbs Model We show that MOM in one Gibbs-perturbation model can be interpreted as MAP over marginals in another continuous-discrete Gibbs model.

$$\begin{aligned}
p_M(y_1, ..., y_m) &\propto \exp\Big(\sum_{e \in E} y_e \log \mu_e\Big) \\
&\propto \prod_e \mu_e^{y_e} \\
&\propto \prod_e (P_\gamma[y_e^\gamma = 1])^{y_e} \\
&\propto \prod_e \Big(E_\gamma 1[y_e^\gamma = 1]\Big)^{y_e} \\
(*) \quad &\propto E_{\gamma^{(1)},...,\gamma^{(m)}} \prod_e (1[y_e^{\gamma^{(e)}} = 1])^{y_e}
\end{aligned}$$

The starred equivalence holds when the product function of expectations is the expectation of the same product function. This equivalence holds when the random variables $1[y_e^\gamma = 1]$ are independent. To enforce the independence assumption, the starred equivalence requires an independent perturbation vector $\gamma^{(e)} = (\gamma_1^{(e)}, ..., \gamma_m^{(e)})$ for each edge.

Using this independence assumption we are able to represent $p_M(y_1, ..., y_m)$ as the expectation of a product of functions,

$q_{\theta,\sigma}(\gamma^{(e)})1[y_e^{\gamma^{(e)}} = 1]$. This factorization naturally lends a Gibbs model over the factors $\psi_e(\gamma^{(e)}, y_e) \overset{def}{=} \log(q_{\theta,\sigma}(\gamma^{(e)})1[y_e^{\gamma^{(e)}} = 1])$. Hence, the MAP assignment of Equation 14 is the MAP over the structure variables y of the marginals over the continuous variables γ of the discrete-continuous Gibbs model:

$$p(y, \gamma) \propto \exp\Big(\sum_e \psi_e(\gamma^{(e)}, y_e)\Big) \tag{15}$$

Marginals Estimation The last detail required for the implementation of the MOM inference approach in Gibbs-perturbation models is recovering the marginals μ_e. Unfortunately, we are not aware of any direct way to do that. Instead, we propose to approximate the marginals by sampling K times from the model and computing the marginals using a maximum-likelihood approach on this sample. Particularly, in our first-order dependency parsing example we set μ_e to be the number of trees in the K-list that contain the edge e.

As noted above, the idea of computing an MST over single-edge marginals has been proposed in Kuncoro et al. (2016) where the marginals were computed in a manner similar to ours - using the K parse trees of their K ensemble members. Our novelty is with respect to the way the dependency trees in the K-list are extracted: while they built on the non-convexity of neural networks and ran an LSTM-based parser (Dyer et al., 2015) from different random initializations, we develop a perturbation-based framework. Our method for K-list generation is often more efficient than that of Kuncoro et al. (2016). While we train a parser and a noise function and can then generate the K-list by solving K argmax problems, their method requires the training of K LSTM parsers.

5 Tasks, Models and Experiments

5.1 Tasks and Data

Data. We consider two dependency parsing tasks: cross-lingual and mono-lingual but lightly supervised. For both tasks we consider Version 2.0 of the Universal Dependencies (UD) Treebanks (Nivre et al., 2016; McDonald et al., 2013).[7] The dataset consists of 77 corpora from 45 languages. We use the gold POS tags in our experiments.

We excluded 3 languages (Hindi, Urdu and Japanese) with 5 corpora from the dataset, as all models we experiment with (perturbated or not) demonstrated very poor results on these languages. An analysis revealed that the head-modifier distributions in these five corpora are very different from the corresponding distributions in the other corpora, which might explain the poor performance of the parsers.

Task1: Cross-lingual Dependency Parsing. In this setup, for each corpus we train on all the training sets of the corpora in the dataset as long as they are of another language (the source languages training sets), and test on the test set of the target corpus. For this purpose, for each of the 72 corpora we constructed a training set of 1000 sentences and a development set of 100 sentences, taken from the training and the development sets of the corpora, respectively.[8] Then, for each target corpus we train the parser parameters (θ) on a training set that consists of the training sets of all the corpora except from those of the target language (the source languages corpora), where for the non-perturbated models (see below) this training set is augmented with the development sets of the source language corpora. For the perturbated models, the development sets of the source languages are used for learning the noise parameter (σ). For test we keep the original test sets of the UD corpora.

To make the data suitable for cross-language transfer we discard words from the corpora. The parsers are then fed with the universal POS tags, that are identical across languages.

Task2: Lightly Supervised Mono-lingual Dependency Parsing. For this setup we chose 12 low-resource languages (13 corpora) which have between 300 to 5k training sentences: Danish, Estonian, Greek, Hungarian, Indonesian, Korean, Latvian, Old Church Slavonic, Persian, Turkish (2 corpora), Urdu and Vietnamese. For each language we randomly sample 300 sentences for its training set and test on its UD Treebank test set.

In this setup, to keep with the low resource language spirit, we do not learn the noise parameter (σ) but rather use fixed noise parameters for the perturbated models (see below). As opposed to the cross-lingual setup, all the parsers are lexicalized, as this is a mono-lingual setup.

[7] https://universaldependencies.org/

[8] 8 corpora had less than 1000 training sentences, and 8 corpora had less than 100 development sentences. For these we took the entire training or development set, respectively.

Previous Work Recent years have seen substantial efforts devoted to our setups. For cross-lingual parsing, the proposed approaches include the use of typological features (Naseem et al., 2012; Täckström et al., 2013; Zhang and Barzilay, 2015; Ponti et al., 2018; Scholivet et al., 2019), annotation projection and other means of using parallel text from the source and target languages (Hwa et al., 2005; Ganchev et al., 2009; McDonald et al., 2011; Tiedemann, 2014; Ma and Xia, 2014; Rasooli and Collins, 2015; Lacroix et al., 2016; Agić et al., 2016; Vilares et al., 2016; Schlichtkrull and Søgaard, 2017), similarity modeling for parser selection (Rosa and Zabokrtsky, 2015), late decoding (Søgaard and Schlichtkrull, 2017) and synthetic languages (Wang and Eisner, 2016, 2018b,a). Likewise, lightly supervised parsing has been addressed with a variety of approaches, including co-training (Steedman et al., 2003), self-training (Reichart and Rappoport, 2007) and inter-sentence consistency constraints (Rush et al., 2012).

Our goal is to provide a technique that can enhance any machine learning model for structured prediction in NLP in cases where high quality parameter estimation is challenging and the argmax solution is likely not to be the highest quality solution. We choose the tasks of cross-lingual and lightly supervised dependency parsing since they form prominent NLP examples for our problem. We hence focus our experiments on an in-depth exploration of the impact of our framework on a dependency parser, rather than on a thorough comparison to previously proposed approaches.

5.2 Models and Experiments

Parsing model. We implemented our method within the linear time incremental parser of Huang and Sagae (2010).[9] While our method is applicable to any parameterized data-driven machine learning model, including deep neural networks, we chose to focus here on a linear parser in which noise injection is straight-forward: all the weights in the weight vector of the model are perturbated. We chose to avoid implementation within LSTM-based parsers (Dyer et al., 2015; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017), as in such models the perturbation parameters may be multiplied by each other (due to the deep, recurrent, nature of the network) causing second-order effects. We leave decisions relevant for neural parsing, e.g. which subset of the LSTM parameter set should be perturbated in order to achieve the most effective model, for future research.

Models and Baselines We compare between seven models. The main two models are our perturbation-based parsing models, where the variance is learned from data. We consider additive learned noise (ALN) and multiplicative learned noise (MLN) (Equations 6 and 7). In order to quantify the importance of data-driven noise learning we compare to two identical models where the variance is not learned from data but is rather fixed to be 1.[10] These baselines are denoted with AFN and MFN, for additive fixed noise and multiplicative fixed noise, respectively. As noted above, for the mono-lingual setup we do not implement the ALN and MLN models so that to keep the small training data spirit.

The fifth model is the baseline "1-best" parser - that is, the linear incremental parser with its original inference algorithm that outputs the solution with the best score under the model's scoring function. The sixth model, denoted as the "K-best parser" is a variant of the incremental parser that outputs the K top scoring solutions under the parser's scoring function. The K-best inference algorithm is described in Huang and Sagae (2010) and is implemented in the parser code that we use.

Finally, while in this paper we do not explore the integration of perturbations into LSTM-based parsers, we do want to verify that our methods can boost a linear parser to improve over such neural parsers. For this aim, we also compare our results to the 1-best solution of the transition-based BiLSTM parser of Kiperwasser and Goldberg (2016). We refer to this parser as KG (1-best).[11]

We further explored alternatives to the MOM inference algorithm for distilling the final tree from the various K-lists. Among these are training a feature-rich reranker to extract the best tree from the list, and extracting the tree that is most or least similar to the other trees. As all these alternatives were strongly outperformed by the MOM algorithm, we do not discuss them further.

[9] `https://github.com/lianghuang3/lineardpparser`

[10] In our models that do learn the variance, variance values were in the (0,2] range. We hence consider the value of 1 as a decent proxy to the condition where the variance is not learned from data.

[11] Code was downloaded from the first author's homepage.

Hyper-Parameters The only hyper-parameter of the perturbation method is K - the size of the K-list. As noted in § 3, K can be estimated using, e.g., a grid-search for optimal value on development data. Here we keep with $K = 100$ as the major K value throughout our experiments. However, to get a better understanding of the behavior of our models as a function of K we also consider the setups where $K = 10$ and $K = 200$.[12] All hyper-parameters for both the incremental parser and the baseline BiLSTM parser are set to the default values that come with the authors' code.

6 Results

Cross-lingual Results: MOM Inference. Our results are summarized in Table 1. The final trees extracted by the MOM inference algorithm from the K-lists of the perturbated models with learned noise (the additive model ALN and the multiplicative model MLN) are clearly the best ones, with MLN being the best model both in terms of averaged and median UAS (67.4 and 71.4, respectively) and in terms of the number of corpora for which it performs best (39 out of 72).

Perturbation models with fixed noise (AFN and MFN) compare favorably to K-best inference. However in comparison to 1-best inference, AFN performs very similarly and MFN is outperformed in terms of averaged and median UAS. This emphasizes the importance of noise (variance) learning from data. Interestingly, the final tree extracted by the MOM algorithm from the parser's K-best list is worse than the parser's 1-best tree (averaged UAS of 58.5 vs. 66.4, median UAS of 62.8 vs. 70.2). Both the K-best and 1-best variants of the incremental parser do not provide the best UAS on any of the 72 corpora.

The 1-best solution of the KG BiLSTM parser is very similar to the 1-best solution of the incremental parser in terms of averaged and median UAS. This indicates that the incremental parser to which we integrate our perturbation algorithm does not lag behind a more modern neural parser when the training data is not a good representative of the test data - the case of interest in this work. Additionally, the KG parser is less stable - it is the best performing parser on 26 of 72 corpora, but on 34 corpora it is outperformed by the 1-best solution of the incremental parser, of which on 9 corpora the gap is larger than 3%. Detailed per language results are presented in Table 3.

[12]For $K = 200$, we set the beam width parameter of the parser's inference algorithm to 5000. Yet, even with this value the parser did not produce 200 trees for all sentences. The same pattern was observed for smaller K values, although less frequently.

Cross-lingual Results: List Quality. Since the focus of this paper is on the quality of the K-list, the table also reports the quality of each model assuming an oracle that selects the best tree from the K-list. Here the table clearly shows that perturbation with learned variance (MLN and ALN) provides substantially better K-lists. For example, MLN achieves an averaged UAS of 80.3, a median UAS of 83.4 and it is the best performing model on 58 of 72 corpora.

The gaps from the 1-best and K-best inference algorithms of the incremental parser as well as from the KG BiLSTM parser are substantial in this evaluation. For example, the average and median UAS of the KG BiLSTM parser are only 66.6 and 69.9, reflecting a gap of 13.7 and 13.5 UAS points from MLN. Moreover, the non-perturbated methods do not provide the best results on any of the 72 corpora in this oracle selection evaluation: MLN is the best performing inference algorithm in 58 cases and MFN in 14 cases.

As in MOM inference, noise learning (MLN and ALN) continues to outperform perturbation with fixed noise (MFN and AFN) both in terms of averaged and median USA. For example the averaged UAS of MLN is 80.3 compared to 77.1 for MFN, and the number of corpora on which MLN performs best is 58, compared to 14 of MFN.

The oracle results are very important as they indicate that improving the MOM inference method has a great potential to make cross-lingual parsing substantially better. None of the other models we consider extracts K-lists with candidate trees of the quality that our perturbated models do.

We next consider the quality of the full K-lists of the different methods, rather than of the oracle best solutions. Figure 1 (top) compares the averaged UAS of the trees in the 1, 25, 50, 75 and 100 percentiles of the K-lists produced by the various inference methods. The K-lists of the perturbation based methods are clearly better than those of the K-best list, with the ALN, AFN and MLN methods performing particularly well. Likewise, Figure 1 (bottom) demonstrates that the percentage of trees that fall into higher 10% UAS bins is substantially higher for MLN and ALN compared to K-best inference (the figure considers all the K-

Method	Av. UAS (M)	Md. UAS (M)	Av. UAS (O)	Md. UAS (O)	# Cor. (M)	# Cor. (O)
1-best	66.4	70.2	66.4	70.2	0	0
K-best	58.5	62.8	74.8	77.1	0	0
AFN	66.6	70.6	73.4	76.9	0	0
MFN	62.6	65.2	77.1	78.7	5	14
ALN	66.9	70.9	76.9	80.4	4	0
MLN	**67.4**	**71.4**	**80.3**	**83.4**	**39**	**58**
KG	66.6	69.9	66.6	69.9	26	0

Table 1: Results summary, cross-lingual parsing, $K = 100$. We report average (Av.) and median (Md.) UAS (across languages) of each model with MOM inference (M) and with an oracle that chooses the best tree out of the K-list produced by the model (O). The # Cor. columns report the number of corpora for which the model is the best scoring one (in case two models perform best on the same language, it counts for both). For 1-best and KG (1-best), both MOM (M) and Oracle (O) refer to the single tree produced by the model.

Method	A-U (M)	M-U (M)	A-U (O)	M-U (O)	#-C (M)	#-C (O)	A-U-T	M-U-T
10-best	63.1	67.4	72.7	75.5	3	5	9.8	10
MLN-10	**66.6**	**70.5**	**75.1**	**78.1**	**69**	**67**	6.86	8
200-best	57.2	60.7	75.1	77.3	0	1	126.1	130
MLN-200	**67.6**	**71.6**	**83.2**	**85.8**	**72**	**71**	92.7	96

Table 2: Cross-lingual parsing results as a function of K, the size of the K list for the K-best and MLN parsers. A-U and M-U refer to average and median UAS across languages, respectively. #-C refers to the number of corpora for which the model is the best scoring one. (M) refers to MOM inference, while (O) refers to oracle selection of the best tree from the list. A-U-T and M-U-T refer to the average and median number of unique trees in the list, respectively. As noted above, the K-best model cannot generate K trees for all sentences.

lists form the 72 test sets). That is, the perturbated lists are of higher quality than the K-best lists both when the oracle solution is considered and when the full lists are evaluated.

Figure 2 compares the full lists of MLN and ALN to the unique trees of the lists, in terms of averaged UAS (the bottom graph is limited to MLN, but the pattern for ALN is similar). The consistent pattern we observe is that the average quality of the full lists is higher than that of the unique trees of the lists. This means that the full lists have multiple copies of their higher quality trees, a property we consider desirable as our goal is to sample from the score space of the model and hence higher quality trees should be over-represented.

Cross-lingual Results: Results as a Function of K. Finally, Table 2 compares the K-lists of the MLN and the K-best inference algorithms for list size values (K) of 10 and 200. MLN is clearly much better both when the final tree is selected with MOM inference and when it is selected by the oracle. The two rightmost columns of the table indicate that the number of unique trees is much higher in the K-best list, as discussed above.

Lightly Supervised Mono-lingual Results Table 4 (which is equivalent to Table 1 for cross-lingual parsing) and Figure 3 (which is equivalent to Figure 1) summarize the results for the monolingual setup. We present these results more briefly due to space limitations. We recall that in this setup we do not learn the noise, due to the shortage of training data, but rather used the fixed noise variance parameter of 1 (§5.2).

The table shows that MFN is the best performing model both when MOM inference is used and when the best tree is selected by an oracle. As in the cross-lingual setup, the gap in the oracle selection case is much larger (e.g. an averaged UAS gap of 14.8 points from the 1-best parser, the second best model) than in the MOM inference setup (an averaged UAS gap of 1.5 points from 1-best).

However, in certain aspects the results in this setup indicate a stronger impact of perturbations. First, MFN performs best on 12 of 13 corpora with MOM inference and in 13 of 13 corpora with oracle selection. Moreover, its gap from the BiLSTM parser is larger than in the cross-lingual setup, probably due to the strong dependence of neural models on large training corpora.

Finally, Figure 3 presents a similar effect to Fig-

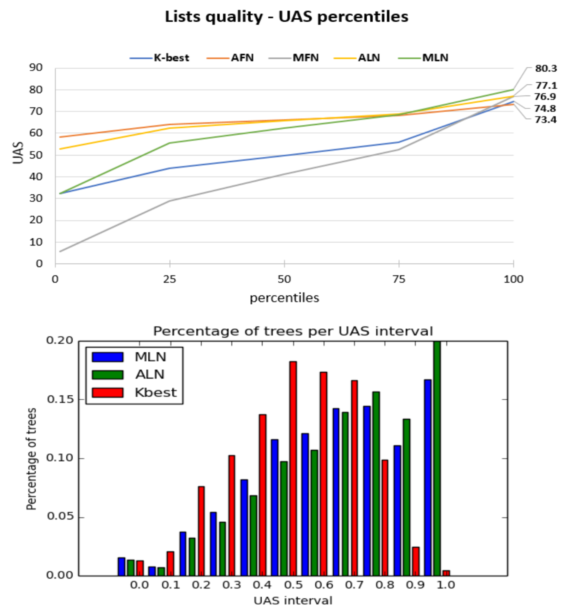

Figure 1: Cross-lingual parsing, $K = 100$. Top: Averaged UAS of the trees in the M-th percentile of the K-list of each model (values were computed for $M = 1, 25, 50, 75, 100$). Bottom: Percentage of trees in each 10% UAS bin, for the K-list of each model. In both cases the values are calculated across all the trees in the lists produced for all test sets.

ure 1. The K-lists of the perturbated models are clearly better than those of the K-best inference, which is reflected both by the percentile analysis (top graph) and the UAS histogram that is taken across all 13 experiments (bottom graph).

7 Additional Setups and Limitations

Our experimental setup has made several limiting assumptions. Here we address three of these assumptions and explore the extent to which they reflect true limitations of our framework.

Additional Task: Cross-lingual POS Tagging Our main results were achieved with a single incremental linear parser. We next explore the impact of our framework on another task: cross-lingual POS tagging. Training and development are performed with the training and development portions of the English (en) UD corpus (16371 and 3414 sentences, respectively) and the trained model is applied to six languages (11 corpora) from four different families: Italian, Portuguese (both are Italic, Romance), modern Hebrew, Arabic (both are Semitic), Chinese and Japanese.

Our POS tagger is a BiLSTM with two fully connected (FC) classification layers that are fed

Corpus	1-best	K-best	ALN	MLN	AFN	MFN	KG
Korean	40.4	35.9	40.3	38.1	40.4	**41.8**	37.9
English	69.2	56.3	69.3	**70.1**	69.1	67.5	69.1
English_pud	70.1	64.1	70.8	**70.9**	70.7	66.5	69.6
English_partut	72	66.6	72.6	**72.8**	72.4	69.9	**72.8**
English_lines	71	61.4	71.5	**72.2**	71.3	68.4	70.9
Gothic	63.8	50.1	64	64.3	63.8	57.3	**67.3**
Czech_pud	74.7	67.4	**75.5**	75.3	74.9	70.6	74.1
Czech_cac	75.3	66	75.7	**76.2**	75.6	70.4	75.7
Czech_cltt	69.6	66.3	70.9	**71.8**	70.3	61.5	64.4
Czech	72	64.2	72.7	**73.2**	72.3	66.7	72.5
Portuguese	79.1	73	79.7	79.7	79.5	75.2	**81.2**
Portuguese_br	77.2	73.6	77.6	**78**	77.3	74	77.4
Portuguese_pud	72.2	68.4	72.5	72.8	72.4	64.9	**74**
Chinese	34.2	33.4	34.7	**35.2**	34.5	35	33.9
Ancient_Greek_proiel	59.8	50.6	60.1	61	59.9	55.7	**61.5**
Ancient_Greek	51.3	43.7	51.7	**52.6**	51.6	49.3	49.1
Uyghur	32.2	27.9	32.6	33.1	32.4	**44.2**	42.3
Indonesian	70.2	62.6	70.7	**71.5**	70.6	64.4	65.4
Romanian	71.9	66	**72.8**	72.6	72.6	65.4	71.2
Slovak	78	63	**78.6**	78.3	78	72	76.6
Galician	69.3	67.7	69.9	70	69.6	65.6	**72.1**
Galician_treegal	77.5	72.2	77.8	78.5	77.7	72.4	**80.7**
Bulgarian	81.1	67.7	**81.9**	**81.9**	81.7	74.7	80.6
Hebrew	63.2	58.7	63.8	63.8	63.4	55.5	**64.1**
Croatian	72.7	66.8	**73.8**	73.6	73.3	64.8	73
Kazakh	45.6	33.4	46	47.3	45.7	47.6	**53.3**
Catalan	77.4	73.3	77.8	78.2	77.6	74.2	**79.2**
Latin_ittb	63.3	53.8	64	**64.3**	63.5	60.4	60.8
Latin	49.5	38.4	49.6	50.4	49.2	**52.2**	47.8
Latin_proiel	55.7	44.9	56.6	56.8	56	54.6	**58.1**
French	77.6	72.4	77.6	78.2	77	73.1	**79.7**
French_pud	69.9	67.9	70.6	70.9	70.4	65	**72.9**
French_sequoia	74.4	69.3	75.1	74.9	74.8	71.8	**77.7**
French_partut	79.7	75.4	80.2	80.4	79.8	73.8	**83**
Latvian	54.8	41.7	55	**55.9**	54.8	53.4	54.1
Greek	75.5	69.3	76.5	**76.9**	76.3	70.2	74.4
Danish	71	62	71.3	71.8	71.2	66.7	**72**
Persian	55.7	53.2	57.9	**59.1**	56.6	55.1	44.2
Dutch_lassysmall	68.5	56.1	68.4	**69.1**	68	64.6	65.2
Dutch	67.5	59.3	67.4	**68**	67.1	63.5	63.3
Ukrainian	76.8	64.6	77.6	**77.8**	77.4	71.7	75.4
Basque	47.4	34.9	47.8	**48.4**	47.3	42.8	47.3
Estonian	68	52.3	68.7	**70.3**	68.5	66.4	65.6
Spanish_ancora	76.8	72.6	76.9	77.1	76.6	64.2	**78.7**
Spanish_pud	72.8	69	73.2	73.7	73.2	66.8	**74.4**
Spanish	74.9	71.2	75.4	75.5	75.2	70.2	**76**
Arabic_pud	66.8	58.8	67.2	67.1	66.5	58.8	**68.5**
Arabic	52.2	51	53.3	55.1	52.6	48.6	**56.9**
Polish	81.8	57.7	82.5	**82.6**	82.2	74.8	79.9
Hungarian	60.8	55.7	61.2	**62.2**	61	57.7	58.1
Italian_pud	79.5	74.8	79.8	79.8	79.6	70.6	**81.3**
Italian	80.2	73.4	80.3	80.7	80.1	71.5	**82**
Swedish	76.5	65.1	76.6	**76.8**	76.5	71.7	76
Swedish_pud	76.4	67.6	76.4	**76.8**	76.3	72.7	76.1
Swedish_lines	76.5	63.6	76.5	**77**	76.5	71.5	**77**
Vietnamese	48.9	41.2	49.8	**50.8**	49.2	46	41.9
Norwegian_nynorsk	74	64.3	**74.1**	**74.1**	73.8	68.2	72.9
Norwegian_bokmaal	77.5	64.6	77.7	**77.9**	77.4	71.9	76.1
Old_Church_Slavonic	63.2	47.1	63	63.8	62.7	58.1	**69.6**
Russian_syntagrus	61.4	58.9	61.9	**62.4**	61.5	59.2	60.6
Russian_pud	72.2	65.8	72.6	**73.2**	72.3	66.5	71.2
Russian	70.7	63.1	70.6	**71.1**	70.5	63.9	70
Slovenian	80.8	71.7	81.4	**82**	81.1	69	80.5
Slovenian_sst	64.7	54.2	65	**65.4**	64.8	56.3	58.5
Finnish_pud	60	50.4	61.1	**61.7**	60.7	56.4	60.3
Finnish_ftb	51.2	43.8	51.3	51.6	51.3	51.4	**52.4**
Finnish	60.9	48.3	61.1	**61.4**	60.8	54.3	61.3
Turkish_pud	35.2	28.7	36.3	37.7	36	**46.2**	43.7
Turkish	34.4	28.2	34.7	35.3	34.7	**44.2**	39.1
Irish	61.2	54.3	61.7	**63.4**	61	57.9	62.3
German	70.2	60.4	70.9	**71.4**	70.5	69.5	65.8
German_pud	75.8	68.1	76.5	**76.7**	76.3	71.3	69.9
Avg.	66.4	58.5	66.9	**67.4**	66.6	62.6	66.6

Table 3: Corpus UAS, cross-lingual parsing, $K = 100$.

with the hidden vector produced for each input word. MLN noise was injected only to the final FC layer to avoid second-order effects where per-

Method	Av. UAS (M)	Md. UAS (M)	Av. UAS (O)	Md. UAS (O)	# Cor. (M)	# Cor. (O)
1-best	69.2	71.1	69.2	71.1	0	0
K-best	58.4	55.8	77.8	78	0	0
AFN	69.6	71.7	77.7	79.8	0	0
MFN	**70.7**	**72.7**	**84**	**83.4**	**12**	**13**
KG	65.8	66.9	65.8	66.9	1	0

Table 4: Results summary, mono-lingual parsing, $K = 100$. Table format is identical to Table 1.

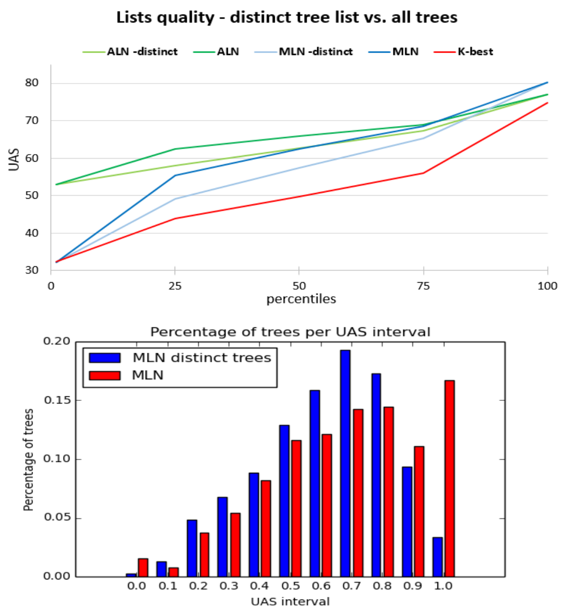

Figure 2: Cross-lingual parsing, $K = 100$. Graphs format is identical to Figure 1, but the comparison is between the full K-list and the unique trees in the K-list for each model.

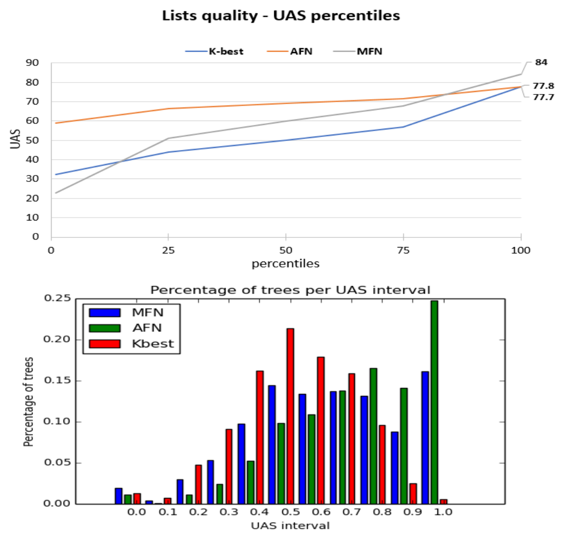

Figure 3: Mono-lingual parsing, $K = 100$. Graphs format is identical to Figure 1.

turbation parameters are multiplied by each other. While we consider here a deep learning model, the noise injection scheme is very simple.[13]

To close the lexical gap between languages we train the English model with the English fastText word embeddings (Bojanowski et al., 2017; Grave et al., 2018). Then, at test time the target language fastText embeddings are mapped to a bilingual space with the English embeddings using the Babylon alignment matrices (Smith et al., 2017).

We consider a $K = 100$ list size. For our MLN method we perform greed search over two ranges of the noise parameter: [0.001, 0.01] and [0.1,0.5]. Noticing that BiLSTMs predict the POS of each word independently, beam search cannot be applied for K-best list generation in this model. Hence, we generate the K-best list with a greedy search strategy that gets the 1-best solution of the model as input and iteratively makes a single word-level POS change with the minimal (negative) impact on the model score. When we do that, we keep track of previously generated solutions so that to generate K unique solutions. We distilled the final solution from the K-lists (ours and the K-best) with a per-word majority vote.

Our results indicate a clear advantage for the perturbated model. Particularly, for all 11 target corpora it is the final solution of this model that scores best. On average across the 11 corpora, the accuracy of our model is 53.05%, compared to 51.44% of the 1-best solution and 41.56% of the solution distilled from the K-best list. This low number of the latter solution is a result of its low quality lists which contain many poor solutions.

Cross-lingual Parsing with Predicted POS Tags

Our main results were achieved with gold POS tags. However, in low resource setups gold POS tags may be unavailable. To explore the impact of gold POS tags availability on our results we run a cross-lingual parsing setup identical to the one of § 5 with MLN and $K = 100$, except that the target language sentences are automatically POS tagged

[13]BiLSTM layer sizes are: word embedding: 300, output representations: 256, first FC: 512, second FC: 216.

before they are fed to the parser. We consider the 11 target corpora of the 6 languages in our cross-lingual POS tagging experiments, and the English-trained non-perturbated BiLSTM tagger.

The result pattern we observe is very similar to the cross-lingual parsing with gold POS tags, although the absolute numbers are lower. Particularly, the averaged UAS of the final solution of our model is 29.8, compared to 26.7 for K-best and 28.1 for 1-best. However, the quality of the perturbated list is much higher than that of the K-best list, as is indicated, for example, in the gap between their best oracle solutions (46 vs. 37.6). These results emphasize the importance of high quality POS tags for cross-lingual parsing. Presumably, manual POS tagging is a substantially easier task compared to dependency parsing so this requirement is hopefully not very restricting.

Well Resourced Monolingual Parsing Finally, our framework was developed with the motivation of addressing cases where the argmax solution of the model is likely not the highest quality one. We hence focused our experiments in cross-lingual and lightly-supervised parsing setups. However, it is still interesting to evaluate our framework in setups where abundant labeled training data from the target language is available.

For this aim we implemented an in-language well-resourced parsing setup, identical to the $K = 100$ lightly-supervised parsing setup of § 5, except that the incremental linear parser and the MLN parameter are trained, developed and tested on the corresponding portions of a single UD corpus. We run this experiment with 31 corpora of 14 UD languages: Arabic, German, English, Spanish, French, Hebrew, Japanese, Korean, Dutch, Portuguese, Slovenian, Swedish, Vietnamese and Chinese. We chose these languages in order to experiment with a wide range of corpus sizes. As in § 5, for the perturbation model the parser is trained on the training set and the noise parameter is learned on the development set, while the base parser is trained on a concatenation of both sets.

In this more challenging setup, the distilled solution of the perturbated parser does not outperform the 1-best solution: on average across corpora its UAS is 82.5 while the 1-best scores 82.3. Interestingly, the distilled solution of the K-best list achieves an average UAS of only 72.9. However, in terms of list quality the perturbation model still excels. For example, the averaged UAS of its oracle best solution is 91.7 compared to 87.3 of the K-best list. Likewise, its 25%, 50% and 75% percentile solutions score 70.1, 75.2 and 79.6 on average, respectively, while the respective numbers for the K-best list are only 58.2, 63.6 and 69.3. From these results we conclude that our model can substantially contribute to the quality and diversity of the extracted list of solutions even in the well-resourced in-language setup, but that its potential impact on a single final solution is more limited.

8 Conclusions

We presented a perturbation-based framework for structured prediction in NLP. Our algorithmic contribution includes an algorithm for data-driven estimation of the perturbation variance and a MOM algorithm for distilling a final solution from the K-list. An appealing theoretical property of our method is that it can augment any machine learning model, probabilistic or not, and draw samples from a probabilistic model defined on top of that base model. In setups like cross-lingual and lightly supervised parsing where the training and the test data are drawn from different distributions and the argmax solution of the base model is of low quality, our method is valuable in extracting a high quality solution list and it also modestly improves the quality of the final solution. Yet, we note that our current implementation mostly applies to linear models, although we demonstrate initial cross-lingual results with a BiLSTM POS tagger.

In future work we will aim to develop better algorithms for final solution distillation. Our stronger list quality results indicate that an improved distillation algorithm can increase the impact of our framework. Note, however, that MOM is used as part of the noise learning procedure (§3) which yields high quality lists. We would also like to develop means of effectively applying our ideas to deep learning models. While theoretically our framework equally applies to such models, their layered organization requires a careful selection of the perturbated parameters and noise values.

Acknowledgements

We would like to thank the action editor and the reviewers, as well as the members of the IE@Technion NLP group for their valuable feedback and advice. This research was partially funded by ISF personal grants No. 1625/18 and 948/15.

References

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.

Rachel Bawden and Benoît Crabbé. 2016. Boosting for efficient model selection for syntactic parsing. In *Proceedings of COLING*, pages 1–11.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5:135–146.

Leo Breiman. 1996. Bagging predictors. *Machine learning*, 24(2):123–140.

Paolo M. Camerini, Luigi Fratta, and Francesco Maffioli. 1980. The k best spanning arborescences of a network. *Networks*, 10(2):91–109.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of ACL*, pages 173–180.

Michael Collins. 2002. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of ACL*, pages 489–496.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of ICLR*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL*, pages 334–343.

Gerald Folland. 1999. Real analysis: Modern techniques and their applications, john wiley & sons. *New York*.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of ACL-AFNLP*, pages 369–377.

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A systematic exploration of diversity in machine translation. In *Proceedings of EMNLP*, pages 1100–1111.

Leslie Ann Goldberg and Mark Jerrum. 2007. The complexity of ferromagnetic Ising with local fields. *Combinatorics Probability and Computing*, 16(1):43.

Daniil Golod. 2009. The k-best paths in Hidden Markov Models. Algorithms and applications to transmembrane protein topology recognition. Master's thesis, University of Waterloo.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of LREC*.

Keith Hall. 2007. K-best spanning tree parsing. In *Proceedings of ACL*, pages 392–399.

Tamir Hazan and Tommi Jaakkola. 2012. On the partition function and random maximum a-posteriori perturbations. In *Proceedings of ICML*, pages 1667–1674.

Tamir Hazan, Subhransu Maji, Joseph Keshet, and Tommi Jaakkola. 2013. Learning efficient random maximum a-posteriori predictors with non-decomposable loss functions. In *Proceedings of NIPS*, pages 1887–1895.

Tamir Hazan, George Papandreou, and Daniel Tarlow. 2016. *Perturbations, Optimization, and Statistics*. MIT Press.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL*, pages 1077–1086.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(3):311–325.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of EMNLP*, pages 1700–1709.

Joseph Keshet, David McAllester, and Tamir Hazan. 2011. PAC-bayesian approach for minimization of phoneme error rate. In *Proceedings of ICASSP*, pages 2224–2227.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association of Computational Linguistics*, 4:313–327.

Daphne Koller, Nir Friedman, and Francis Bach. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT press.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*, pages 1–11.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one MST parser. In *Proceedings of EMNLP*, pages 1744–1753.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of HLT-NAACL*, pages 1058–1063.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

Jiwei Li and Dan Jurafsky. 2016. Mutual information and diverse decoding improve neural machine translation. *arXiv preprint arXiv:1601.00372v2*.

Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Tiejun Zhao, and Eiichiro Sumita. 2018. Forest-based neural machine translation. In *Proceedings of ACL*, pages 1253–1263.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of ACL*, pages 1337–1348.

Wolfgang Macherey, Franz Josef Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of EMNLP*, pages 725–734.

Chris Maddison, Danny Tarlow, and Tom Minka. 2014. A* sampling. In *Proceedings of NIPS*, pages 2085–2093.

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of ACL*, pages 617–622.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of ACL*, pages 92–97.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of EMNLP*, pages 523–530.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of EMNLP*, pages 62–72.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of ACL*, pages 629–637.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of LREC*, pages 1659–1666.

George Papandreou and Alan Yuille. 2011. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *Proceedings of ICCV*, pages 193–200.

Edoardo Maria Ponti, Roi Reichart, Anna Korhonen, and Ivan Vulić. 2018. Isomorphic transfer of syntactic structures in cross-lingual NLP. In *Proceedings of ACL*, pages 1531–1542.

Lawrence R. Rabiner. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of EMNLP*, pages 328–338.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Proceedings of ACL*, pages 616–623.

Rudolf Rosa and Zdenek Zabokrtsky. 2015. $KL_{cpos}3$ - a language similarity measure for delexicalized parser transfer. In *Proceedings of ACL-IJCNLP*, pages 243–249.

Alexander M. Rush, Roi Reichart, Michael Collins, and Amir Globerson. 2012. Improved parsing and POS tagging using inter-sentence consistency constraints. In *Proceedings of EMNLP*, pages 1434–1444.

Michael Schlichtkrull and Anders Søgaard. 2017. Cross-lingual dependency parsing with late decoding for truly low-resource languages. In *Proceedings of EACL*, pages 220–229.

Manon Scholivet, Franck Dary, Alexis Nasr, Benoit Favre, and Carlos Ramisch. 2019. Typological features for multilingual delexicalised dependency parsing. In *Proceedings of HLT-NAACL*, pages 3919–3930.

Noah A. Smith. 2011. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool.

Samuel L. Smith, David H.P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *proceedings of ICLR*.

Anders Søgaard and Michael Sejr Schlichtkrull. 2017. Cross-lingual dependency parsing with late decoding for truly low-resource languages. In *Proceedings of EACL*, pages 220–229.

Le Hai Son, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of HLT-NAACL*, pages 39–48.

David Sontag, Talya Meltzer, Amir Globerson, Tommi Jaakkola, and Yair Weiss. 2008. Tightening lp relaxations for map using message passing. In *Proceedings of UAI*, pages 503–510.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of EACL*, pages 331–338.

Weiwei Sun and Xiaojun Wan. 2013. Data-driven, PCFG-based and pseudo-PCFG-based models for Chinese dependency parsing. *Transactions of the Association of Computational Linguistics*, 1:301–314.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proceedings of HLT-NAACL*, pages 649–652.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of HLT-NAACL*, pages 1061–1071.

Daniel Tarlow, Kevin Swersky, Richard S. Zemel, Ryan Prescott Adams, and Brendan J. Frey. 2012. Randomized optimum models for structured prediction. In *Proceedings of AISTATS*, pages 1221–1229.

Ilan Tchernowitz, Liron Yedidsion, and Roi Reichart. 2016. Effective greedy inference for graph-based non-projective dependency parsing. In *Proceedings of EMNLP*, pages 711–720.

Jörg Tiedemann. 2014. Rediscovering annotation projection for cross-lingual parser induction. In *Proceedings of COLING*, pages 1854–1864.

Zhaopeng Tu, Wenbin Jiang, Qun Liu, and Shouxun Lin. 2012. Dependency forest for sentiment analysis. In *Natural Language Processing and Chinese Computing*, pages 69–77. Springer.

Zhaopeng Tu, Yang Liu, Young-Sook Hwang, Qun Liu, and Shouxun Lin. 2010. Dependency forest for statistical machine translation. In *Proceedings of COLING*, pages 1092–1100.

David Vilares, Carlos Gómez-Rodríguez, and Miguel A. Alonso. 2016. One model, two languages: Training bilingual parsers with harmonized treebanks. In *Proceedings of ACL*, pages 425–431.

Martin J. Wainwright and Michael I. Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.

Dingquan Wang and Jason Eisner. 2016. The Galactic Dependencies treebanks: Getting more data by synthesizing new languages. *Transactions of the Association for Computational Linguistics*, 4:491–505.

Dingquan Wang and Jason Eisner. 2018a. Surface statistics of an unknown language indicate how to parse it. *Transactions of the Association for Computational Linguistics*, 6:667–685.

Dingquan Wang and Jason Eisner. 2018b. Synthetic data made to order: The case of parsing. In *Proceedings of EMNLP*, pages 1325–1337.

David Bruce Wilson. 1996. Generating random spanning trees more quickly than the cover time. In *Proceedings of STOC*, pages 296–303.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of EMNLP*, pages 1857–1867.

Yuan Zhang, Tao Lei, Regina Barzilay, Tommi Jaakkola, and Amir Globerson. 2014. Steps to excellence: Simple inference with refined scoring of dependency trees. In *Proceedings of ACL*, pages 197–207.