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 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 suffer from 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-
dependency 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.\footnote{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.} 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 \( y = (y_1, \ldots, y_m) \). 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 \( \{1, \ldots, P\} \) where \( P \) is the number of POS tags. In dependency parsing, a graph \( G = (V, E) \) is de-
defined 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 $y_v$, and the result-
ing parse tree is a vector $(y_1, ..., y_m) \in \{0, 1\}^m$
that forms a spanning tree in the graph $G$. For ev-
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.\footnote{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.}

The common practice in structured prediction is
that structures are scored by a function that assigns
favorable structures with high scores and unfavor-
able ones with low scores. The number of struc-
tures $(|T|)$ is often exponential in $m$, as in our run-
ning dependency parsing example. Hence, in or-
der to avoid exponential complexity, the scoring
function has to factorize. In our running example
this is done through:

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

(1)

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

**Prediction with K-lists** Unfortunately, often-
times 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 ($\theta_e$) are used
to score exponentially many structures. This may
also happen in cases where the values of the poten-
tial 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 ex-
tracted from this list - either by selecting one list
member, or by distilling a new solution based on
the statistics of the list members.

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.\footnote{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.}

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

**Sampling-based K-lists** Sampling is a possi-
bile solution to the diversity problem. In prac-
tice, many sampling algorithms require that the
structured model will be defined as a probabilistic
model. It is natural to impose a probabilistic inter-
pretation 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:

$$p_\theta(y_1, ..., y_m) \propto \exp \left( \sum_{e \in E} \theta_e y_e \right)$$

(2)

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

$$y^*_1, ..., y^*_m = \arg \max_{y_1, ..., y_m \in T} p_\theta(y_1, ..., y_m)$$

(3)

Likewise, the top K-list of this model – consist-
ing of the K most probable structures of the Gibbs
distribution – is also identical to that of the unnor-
malized model. As noted above, these structures
are likely to be of high quality but also quite simi-
lar to each other.

The natural alternative that probabilistic models
make possible is to sample from the Gibbs distri-
bution instead. Such a strategy is likely to detect
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 \( \gamma_1, \ldots, \gamma_m \) from a posterior distribution around the base model weights \( \theta_1, \ldots, \theta_m \) and solves the randomly perturbed argmax problem:

\[
y^\gamma = \arg \max_{y \in T} \left\{ \sum_{e \in E} \gamma_e y_e \right\}
\]

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

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

\(^5\)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 \( \theta_e \) 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:

\[
p_\gamma(y_1, \ldots, y_m) = P_{\gamma \sim q_\theta}[y^\gamma = y]
\]

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 \( \gamma_e \) 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 \( \gamma \sim N(0, \sigma^2) \) added to the Gibbs model parameters \( \theta = [\theta_1, \ldots, \theta_m] \), the variance is introduced as

\[
(\text{additive}) \quad q_{\theta, \sigma}(\gamma) = \prod_e \frac{1}{\sqrt{2\pi \sigma_e}} e^{-\frac{(\gamma_e - \theta_e)^2}{2\sigma_e^2}}.
\]

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

\[
(\text{multiplicative}) \quad q_{\theta, \sigma}(\gamma) = \prod_e \frac{1}{\sqrt{2\pi \sigma_e}} e^{-\frac{(\gamma_e - \theta_e)^2}{2\sigma_e^2}}.
\]
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 \((x_i, y_i)\) and the structures with
which they are labeled \((y_i)\), we learn the variance
with respect to the oracle loss \(\text{oracle}_K()\). This loss penalizes the perturbation parameters
\((\gamma_1, \ldots, \gamma_m)\) 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:

\[
\text{oracle}_K(\{\gamma_j\}_j=1^K, x_i, y_i) = \sum_{j=1}^K \text{HamDist}(\text{MOM}(\{\gamma_j\}_j=1^K, x_i), y_i)
\]

where \(\gamma_j = (\gamma_j^1, \ldots, \gamma_j^m)\) 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 \(y_i\):

\[
\text{HamDist}(s, t) = \sum_{j=1}^n \begin{cases} 
1 & \text{if } h_{s,j} = h_{t,j} \\
0 & \text{Otherwise}
\end{cases}
\]

where \(n\) is the number of words in the sentence,
and \(h_{s,j}\) is the head of the \(j\)-th word in \(y_i\).\(^6\)

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

\[
E_{\gamma_1, \ldots, \gamma^n} = \frac{1}{N} \sum_{(x_i, y_i) \in S} E_{\gamma_1, \ldots, \gamma^n} \text{oracle}_K(\{\gamma_j\}_j=1^K, x_i, y_i)
\]

And the optimal \(\sigma\) will minimize this loss:

\[
\sigma^* = \min_{\sigma} E,E_{\gamma_1, \ldots, \gamma^n} \text{oracle}_K(\{\gamma_j\}_j=1^K, x_i, y_i)
\]

Whenever \(q_{\theta,\sigma}(\gamma)\), the perturbation probability
density function (pdf), is smooth in \(\sigma\), 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 \(\sigma\) 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., \(\int |\partial q_{\theta,\sigma}(\gamma) | / \partial \sigma | d\gamma | < \infty\) then the gradient
of the EEL function with respect to \(\sigma\) as computed on \((x_i, y_i) \in S\) takes the form:

\[
\frac{\partial E_{\gamma_1, \ldots, \gamma^n} \text{oracle}_K(\{\gamma_j\}_j=1^K, x_i, y_i)}{\partial \sigma} = \sum_{(x_i, y_i) \in S} \frac{\partial E_{\gamma_1, \ldots, \gamma^n} \text{oracle}_K(\{\gamma_j\}_j=1^K, x_i, y_i)}{\partial \sigma} d\gamma
\]

Proof. The expectation
\(E_{\gamma_1, \ldots, \gamma^n} \text{oracle}_K(\{\gamma_j\}_j=1^K, x_i, y_i)\)
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 \(f(\{\gamma_j\}_j=1^K) = \text{oracle}_K(\{\gamma_j\}_j=1^K, x_i, y_i)\) is
a non-differentiable function. Notably, the function
\(f(\{\gamma_j\}_j=1^K)\) is independent of \(\sigma\) and therefore
its non-differentiability does not affect the differen-
tiability of \(E_{\gamma_1, \ldots, \gamma^n} \text{oracle}_K(\{\gamma_j\}_j=1^K, x_i, y_i)\). 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 inte-
grable function \(N q_{\theta,\sigma}(\gamma_j)\) and its derivative
with respect to \(\sigma\) is bounded by the function
\(N |\partial q_{\theta,\sigma}(\gamma_j) | / \partial \sigma \). Following Theorem 2.27 by
Folland (1999) the function \(E_{\gamma_1, \ldots, \gamma^n} \text{oracle}_K(\{\gamma_j\}_j=1^K, x_i, y_i)\)
is differentiable and its gradient is attained by differentiating
under the integral.

The above claim shows how to learn the optimal
variance of the random perturbation variables
with a gradient method. Note that \(\text{oracle}_K(\{\gamma_j\}_j=1^K, x_i, y_i)\) and
hence also \(E_{\gamma_1, \ldots, \gamma^n} \text{oracle}_K(\{\gamma_j\}_j=1^K, x_i, y_i)\) 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 experi-
ments are with: \(K = 10, 100, 200\) (§ 5).

Once \(\sigma\) and \(K\) are determined, we can generate
meaningful samples, i.e., the perturbation value
\(\gamma_{\theta,S}\) will not shadow the data signal \(\theta_{\theta,S}\). We are
now ready to provide a learning process with
perturbation-based K-lists.
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 $\theta$ (the Gibbs model parameters) and $\sigma$ (the variance of the perturbation distribution) is intractable, we first learn $\theta$ of the perturbation distribution and then $\sigma$ (the Gibbs model parameters) and the Gibbs (base) model. Since joint learning of $\theta$ fully of a higher quality than the output (MAP) of output a single high-quality structure, $y^*$, 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 $\mu_e$ (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.

\[
p_M(y_1, \ldots, y_m) \propto \exp \left( \sum_{e \in E} y_e \log \mu_e \right) \\
\propto \prod_{e} \mu_{y_e}^{y_e} \\
\propto \prod_{e} (P_{\gamma_e[y_e = 1]})^{y_e} \\
\propto \prod_{e} \left( E_{\gamma_e[1]}[y_e = 1] \right)^{y_e} \\
(\ast) \propto E_{\gamma_1, \ldots, \gamma_m} \prod_{e} (1[y_e^{(e)} = 1])^{y_e}
\]

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 $1[y_e^{(e)} = 1]$ are independent. To enforce the independence assumption, the starred equivalence requires an independent perturbation vector $\gamma^{(e)} = (\gamma_1^{(e)}, \ldots, \gamma_m^{(e)})$ for each edge.

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

\[y^* = \arg \max_{y \in T} \left\{ \sum_{e \in E} y_e \log \mu_e \right\}
\]
\( q_{\theta,\sigma}(\gamma^{(e)}) | y_{e}^{(e)} = 1 \). This factorization naturally lends a Gibbs model over the factors 
\( \psi_{e}(\gamma^{(e)}, y_{e}) \overset{\text{def}}{=} \log(q_{\theta,\sigma}(\gamma^{(e)}) | y_{e}^{(e)} = 1) \).
Hence, the MAP assignment of Equation 14 is the MAP over the structure variables \( y \) of the marginals over the continuous variables \( \gamma \) of the discrete-continuous Gibbs model:

\[
p(y, \gamma) \propto \exp \left( \sum_{e} \psi_{e}(\gamma^{(e)}, y_{e}) \right) \tag{15}
\]

**Marginals Estimation**  The last detail required for the implementation of the MOM inference approach in Gibbs-perturbation models is recovering the marginals \( \mu_{e} \). 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 \( \mu_{e} \) 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). 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. Then, for each target corpus we train the parser parameters \( (\theta) \) 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 \( (\sigma) \). 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 \( (\sigma) \) 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/
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 Rapport, 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.

9https://github.com/lianghuang3/lineardpparser

10In 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.

11Code 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 $K = 200$. 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.

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

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.

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

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.

### 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 34414 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 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-

![Image](image_url)
<table>
<thead>
<tr>
<th>Method</th>
<th>Av. UAS (M)</th>
<th>Md. UAS (M)</th>
<th>Av. UAS (O)</th>
<th>Md. UAS (O)</th>
<th># Cor. (M)</th>
<th># Cor. (O)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-best</td>
<td>69.2</td>
<td>71.1</td>
<td>69.2</td>
<td>71.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>K-best</td>
<td>58.4</td>
<td>55.8</td>
<td>77.8</td>
<td>78</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AFN</td>
<td>69.6</td>
<td>71.7</td>
<td>77.7</td>
<td>79.8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MFN</td>
<td><strong>70.7</strong></td>
<td><strong>72.7</strong></td>
<td><strong>84</strong></td>
<td><strong>83.4</strong></td>
<td><strong>12</strong></td>
<td><strong>13</strong></td>
</tr>
<tr>
<td>KG</td>
<td>65.8</td>
<td>66.9</td>
<td>65.8</td>
<td>66.9</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

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.

13BiLSTM 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


Zhaopeng Tu, Wenbin Jiang, Qun Liu, and Shouxun Lin. 2012. Dependency forest for


