# Probabilistic Structured-Output Perceptron: An Effective Probabilistic Method for Structured NLP Tasks

**Xu Sun**
MOE Key Laboratory of Computational Linguistics, Peking University

## Abstract

Many structured prediction tasks are using (structured) perceptrons. In spite of the advantages, the perceptron model unfortunately has relatively low accuracy due to the lack of probability information. To address this issue, we propose a novel probabilistic extension of perceptrons, *probPerc*, which decodes the $n$-best outputs, derives probabilities of the candidates, and performs simple additive updates scaled by probabilities. The proposed method on one hand keeps the key advantages of traditional perceptrons (fast learning and implementation convenience), and on the other hand gains probability information in learning, which leads to better accuracy than existing methods. Experimental results show that the proposed method can achieve state-of-the-art results on real-world natural language processing (NLP) tasks, and at the same time running as fast as perceptrons. To our knowledge, this is the first probabilistic extension of perceptrons.

## 1 Introduction

With the advantages of fast learning and implementation simplicity, many structured prediction tasks are using structured perceptrons [Collins, 2002] and their variations like margin infused relaxed algorithm (MIRA) [Crammer and Singer, 2003]. A major advantage of those methods is that in their learning phase the gradient is not needed and the learning is done by simply decoding and comparing the promising output candidates with the oracle labellings, and then update the model weights accordingly. By avoiding gradient computation, those methods have fast speed compared with gradient-based learning methods like conditional random fields (CRF) [Lafferty *et al.*, 2001; Le and Zuidema, 2014] and deep learning methods like recurrent neural networks (e.g., LSTM) [Hochreiter and Schmidhuber, 1997; Graves *et al.*, 2005].

Another advantage of perceptron family methods is that they are very easy to implement. The major implementation is simply decoding for the output candidates, and do the update accordingly. This is much simpler than the implementation of gradient-based models, because the gradient is typically difficult to compute.

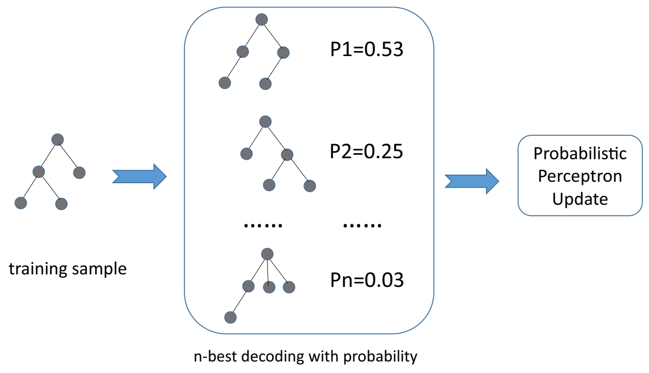
training sample
P1=0.53
P2=0.25
......
......
Pn=0.03
n-best decoding with probability
Probabilistic Perceptron Update

Figure 1: An illustration of the proposed method.

In spite of the advantages, the perceptron family methods unfortunately have relatively low accuracy on structured prediction due to the lack of probability information. Without probability information, in perceptron learning the update is conducted without weighting the confidence of the produced candidates. Instead, each output candidate is treated exactly the same in terms of the weighting in spite of the difference of the confidence, and this is harmful to the prediction accuracy of the learned models.

To address the issue, we propose a novel probabilistic extension of (structured) perceptrons, called *probabilistic perceptron (probPerc)*, for structured NLP problems. The proposed method does $n$-best decoding for the top-$n$ output candidates, derives probabilities based on the candidates, and performs simple additive updates scaled by the derived probabilities. An illustration of the proposed method is shown in Figure 1.

The contributions of this work are as follows:

- To the best of our knowledge, this is the first probabilistic extension of perceptrons. The proposed probPerc method has better accuracy than existing large margin methods like perceptrons, MIRA, and even existing probabilistic models like CRF and LSTM.
- The proposed method has fast training and testing speed, which is almost as fast as traditional perceptrons. Compared with existing probabilistic methods like CRF and

**Algorithm 1** Probabilistic perceptron learning

```
 1: definitions: $\boldsymbol{y}^*$ is oracle labelling, $\boldsymbol{F}(\boldsymbol{y})$ is feature vector,
    $n$ is n-best decoding parameter, $\gamma$ is learning rate
 2: repeat
 3:   Draw a sample $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y}^*)$ from training set
 4:   Decode top-$n$ outputs $Y_n = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_n\}$
 5:   Add $\boldsymbol{y}^*$ to $Y_n$ if $\boldsymbol{y}^* \notin Y_n$ (avoid ill-defined prob.)
 6:   for $\forall \boldsymbol{y}_k \in Y_n$ do
 7:     $P_k \leftarrow P(\boldsymbol{y}_k|Y_n)$ (derive prob. of outputs)
 8:     $\boldsymbol{w} \leftarrow \boldsymbol{w} - \gamma P_k \boldsymbol{F}(\boldsymbol{y}_k)$
 9:   end for
10:   $\boldsymbol{w} \leftarrow \boldsymbol{w} + \gamma \boldsymbol{F}(\boldsymbol{y}^*)$
11: until Convergence
12: output: the (averaged) learned weights $\boldsymbol{w}$
```

**Algorithm 2** Conventional perceptron learning

```
1: repeat
2:   Draw a sample $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y}^*)$ from training set
3:   Decode top-1 output $\boldsymbol{y}_1$
4:   if $\boldsymbol{y}_1 \neq \boldsymbol{y}^*$ then
5:     $\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}_1)$
6:     $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}^*)$
7:   end if
8: until Convergence
9: output: the (averaged) learned weights $\boldsymbol{w}$
```

LSTM, the probPerc is an order of magnitude faster.

- We give theoretical analysis to show that probPerc is convergent in training.

## 2 Proposal

We first describe the proposed probabilistic perceptron algorithm, then we discuss the implementation issues and the convergence properties of the proposed method.

### 2.1 Probabilistic Perceptron

The proposed probabilistic perceptron method has the key components as follows: top-$n$ decoding, a scheme for calculating probabilities, and the additive update of weights based on the derived probabilities. We introduce the technical details of those schemes as follows, and after that we summarize the algorithm in a figure.

First, the proposed method draws a training sample $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y}^*)$ at random from training set $S$, where $\boldsymbol{y}^*$ represents an annotated oracle output. Then, the proposed method decodes for the top-$n$ outputs:

$$Y_n = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_n\}$$

To avoid ill-defined probability in case that $\boldsymbol{y}^*$ is not contained in top-$n$ decoding results, we add the oracle output $\boldsymbol{y}^*$ to $Y_n$ if $\boldsymbol{y}^* \notin Y_n$. Our n-best decoding method is implemented following the idea of Liang and Chiang [2005], which can perform very efficient top-$n$ decoding on different structures (e.g., linear chain and tree structures, see more detailed analysis of the efficiency at the experiment section).

Then, for every $\boldsymbol{y}_k \in Y_n$, we compute the probabilities of the candidate outputs with a log-linear function:

$$P_k \triangleq P(\boldsymbol{y}_k|Y_n, \boldsymbol{x}, \boldsymbol{w}) \triangleq \frac{\exp[\boldsymbol{w}^T \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}_k)]}{\sum_{\forall \boldsymbol{y} \in Y_n} \exp[\boldsymbol{w}^T \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y})]} \quad (1)$$

where $\boldsymbol{w}$ is the vector of the model weights, $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}_k)$ is the feature vector based on $\boldsymbol{x}$ and $\boldsymbol{y}_k$, and $Y_n$ is the n-best decoding outputs defined before. With this definition, we can see that $\sum_{k=1}^{n} P_k = 1$.

After that, the proposed method updates the weights with a additive fashion scaled with the derived probabilities. For every predicted output $\boldsymbol{y}_k \in Y_n$, the weights are updated as follows:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \gamma P_k \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}_k) \quad (2)$$

As we can see, this is similar to the perceptron update, except each update is scaled with a probabilistic confidence $P_k$ and smoothed with a learning rate $\gamma$. A typical choice of learning rate is as follows [Collins *et al.*, 2008]: $\gamma_t = \frac{\gamma_0}{1+t/|S|}$, where $\gamma_0$ is a constant, $t$ is the update counter, and $S$ is like before.

On the other hand, for the oracle output $\boldsymbol{y}^*$, the weights are updated by

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \gamma \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}^*) \quad (3)$$

As we can see, this step is simply the conventional perceptron update smoothed with a learning rate $\gamma$. There is no need to use a probability scaler here, because the probabilistic confidence can be seen as 1 here.

To sum up, the proposed learning algorithm is summarized in Figure 1, where we simply use $P(\boldsymbol{y}|Y_n)$ to denote $P(\boldsymbol{y}|Y_n, \boldsymbol{x}, \boldsymbol{w})$, and $F(\boldsymbol{x}, \boldsymbol{y})$ to denote $F(\boldsymbol{y})$. The conventional perceptron learning algorithm is shown in Figure 2 for the convenience of comparisons.

As we can see, the proposed method is very simple to implement. We only need to replace the 1-best decoding with n-best decoding, which can be conveniently done following Huang and Chiang [2005].

### 2.2 Implementation

**N-Best Decoding**

There are several methods to implement top-$n$ decoding. One solution is to use the A* search algorithm [Hart *et al.*, 1968] with an admissible heuristic function [Sun and Tsujii, 2009]. Another solution is to apply the n-best decoding idea of Huang and Chiang [2005], so that we can decode the top-$n$ outputs by dynamically building the successive candidate by using the historical decoding information of the prior candidate. With this general purpose idea of incremental decoding, the n-best decoding can be efficiently implemented for no matter tree structures or linear chain structures [Huang and Chiang, 2005].

**Testing Stage**

We do not use probability during testing stage. Simply using 1-best decoding (i.e., Viterbi decoding) will be fine for the test data. Actually, many probabilistic models only use non-probabilistic Viterbi-decoding for test, including CRF. It is a similar case for probPerc.

**Probability Function**

We also tested other forms of probabilities, like computing the probability with a linear function and a polynomial function. We find that the log-linear function works better than the linear function and the polynomial function in modeling the probability distribution. It is because log-linear function leads to a probability distribution that is more concentrated on the top-$n$ candidates, such that it fits the n-best decoding scheme of the proposed method.

## 2.3 Theory Justifying the Algorithm

Here we analyze the theoretical properties of the proposed probPerc, we show that it has as good theoretical properties as traditional perceptrons.

**Separable Data**

At each time step, probabilistic perceptron draws a sample $(x, y^*)$, and search for the top-n outputs:

$$Y_n = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, ..., \boldsymbol{y}_n\} \tag{4}$$

Recall that the update is as follows:

$$\boldsymbol{w}^{i+1} = \boldsymbol{w}^i + \gamma \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}^*) - \sum_{k=1}^{n} \gamma P_k \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}_k) \tag{5}$$

Here, $P_k$ and $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}_k)$ are the probability and feature vector of the searched output $\boldsymbol{y}_k$, while $\boldsymbol{w}^i$ is the parameter vector at $i^{th}$ time step.

For the probPerc on separable data, we have the following convergence theorem:

**Theorem 1.** *For any sequence of training samples $(\boldsymbol{x}_i, \boldsymbol{y}_i^*)$ which is separable with margin $\delta$:*

$$\boldsymbol{U} \cdot \boldsymbol{F}(\boldsymbol{x}_i, \boldsymbol{y}_i^*) - \boldsymbol{U} \cdot \boldsymbol{F}(\boldsymbol{x}_i, \boldsymbol{y}_i) \geq \delta \tag{6}$$

*where $\boldsymbol{U}$ is an existing separation vector (because the data is separable) satisfying $\|\boldsymbol{U}\| = 1$, then the probabilistic perceptron has bounded number of updates before convergence:*

$$t \leq R^2/\delta^2 \tag{7}$$

*where $R$ has the same definition following Collins [2002] such that $\|\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}^*) - \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}_k)\| \leq R$.*

The proof is shown in Appendix. Theorem 1 shows that probabilistic perceptron has bounded number of updates before convergence, and the number of updates is bounded by $t \leq R^2/\delta^2$. It means that probabilistic perceptron has as good convergence rate as traditional perceptron in separable cases.

**Inseparable Data**

For inseparable data, following the analysis of traditional perceptron [Collins, 2002], we have the following theorem for probabilistic perceptron:

**Theorem 2.** *For a training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i^*)\}_{i=1}^m$, we define $D_{\boldsymbol{U},\delta}$ as a measure of how close $\boldsymbol{U}$ is to separate the data with margin $\delta$:*

$$m_i = \boldsymbol{U} \cdot \boldsymbol{F}(\boldsymbol{x}_i, \boldsymbol{y}_i^*) - \max_{\boldsymbol{y}_i'} \boldsymbol{U} \cdot \boldsymbol{F}(\boldsymbol{x}_i, \boldsymbol{y}_i') \tag{8}$$

$$\epsilon_i = \max\{0, \delta - m_i\} \tag{9}$$

$$D_{\boldsymbol{U},\delta} = \sqrt{\sum_{i=1}^{m} \epsilon_i^2} \tag{10}$$

*then probabilistic perceptron has finite number of mistakes for the first pass over the training set:*

$$err \leq \min_{\boldsymbol{U},\delta} \frac{(R + D_{\boldsymbol{U},\delta})^2}{\delta^2} \tag{11}$$

The proof is similar to that of traditional perceptron [Collins, 2002]. Theorem 2 shows that the algorithm makes a small number of mistakes for inseparable data, and the bound is as good as traditional perceptron case.

## 3 Related Work

Among the existing structured learning methods, the most close methods to the proposed one are conventional structured perceptrons [Collins, 2002], max-violation perceptrons [Yu *et al.*, 2013], and n-best MIRA [McDonald *et al.*, 2005].

If we compare the proposed method with the conventional structured perceptron [Collins, 2002], the conventional perceptron can be seen as an extreme case of the proposed method with $n = 1$ (i.e., using top-1 decoding instead of top-$n$ decoding). The max-violation perceptron [Yu *et al.*, 2013] is an extension of the perceptron with max-violation updates. The proposed method is also substantially different compared with max-violation perceptrons because of the probability estimations.

The MIRA algorithm has a variation of n-best MIRA which also uses n-best decoding [Crammer and Singer, 2003]. Nevertheless, the proposed probabilistic perceptron is substantially different compared with n-best MIRA. The major difference is that probabilistic perceptron has probability distribution of different outputs, and n-best MIRA treat different outputs equally without probability difference. Another difference is that probabilistic perceptron does not use the "minimum change of weights" optimization criterion of MIRA during weight update, which on the other hand is the core algorithmic component in MIRA and n-best MIRA.

## Appendix

Here we give a brief proof of the Theorem 1. The proof of Theorem 2 is similar to that of structure perceptron [Collins, 2002].

Note that the probabilities of top-n outputs add up to 1: $\sum_{k=1}^{n} P_k = 1$, Thus, Eq.5 can be rewritten as:

$$\boldsymbol{w}^{i+1} = \boldsymbol{w}^i + \sum_{k=1}^{n} \gamma P_k (\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}^*) - \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}_k)) \tag{12}$$

Hence, it goes to:

$$\begin{aligned} \boldsymbol{U} \cdot \boldsymbol{w}^{i+1} &= \boldsymbol{U} \cdot \boldsymbol{w}^i + \sum_{k=1}^{n} \gamma P_k \boldsymbol{U} \cdot (\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}^*) - \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}_k)) \\ &\geq \boldsymbol{U} \cdot \boldsymbol{w}^i + \sum_{k=1}^{n} \gamma P_k \delta \\ &= \boldsymbol{U} \cdot \boldsymbol{w}^i + \gamma \delta \end{aligned}$$

Since the initial parameter $\boldsymbol{w}^1 = 0$, we will have that $\boldsymbol{U} \cdot \boldsymbol{w}^{t+1} \geq t\gamma\delta$ at $t^{th}$ time step. Because $\boldsymbol{U} \cdot \boldsymbol{w}^{t+1} \leq \|\boldsymbol{U}\|\|\boldsymbol{w}^{t+1}\|$, we can see that

$$\|\boldsymbol{w}^{t+1}\| \geq t\gamma\delta \tag{13}$$

On the other hand, $\|\boldsymbol{w}^{i+1}\|$ can be written as:

$$\begin{aligned}\|\boldsymbol{w}^{i+1}\|^2 &= \|\boldsymbol{w}^i\|^2 + \|\sum_{k=1}^{n} \gamma P_k(\boldsymbol{F}(\boldsymbol{x},\boldsymbol{y}^*) - \boldsymbol{F}(\boldsymbol{x},\boldsymbol{y}_k))\|^2 \\ &+ 2\boldsymbol{w}^i \cdot (\sum_{k=1}^{n} \gamma P_k(\boldsymbol{F}(\boldsymbol{x},\boldsymbol{y}^*) - \boldsymbol{F}(\boldsymbol{x},\boldsymbol{y}_k))) \\ &\leq \|\boldsymbol{w}^i\|^2 + \|\sum_{k=1}^{n} \gamma P_k(\boldsymbol{F}(\boldsymbol{x},\boldsymbol{y}^*) - \boldsymbol{F}(\boldsymbol{x},\boldsymbol{y}_k))\|^2 \\ &\leq \|\boldsymbol{w}^i\|^2 + \gamma^2 R^2\end{aligned}$$

The first inequality is based on the property of perceptron update such that the incorrect score is always higher than the correct score (the searched incorrect structure has the highest score) when an update happens. Thus, it goes to:

$$\|\boldsymbol{w}^{t+1}\|^2 \leq t\gamma^2 R^2 \tag{14}$$

Combining Eq.13 and Eq.14, we have:

$$t^2\gamma^2\delta^2 \leq \|\boldsymbol{w}^{t+1}\|^2 \leq t\gamma^2 R^2 \tag{15}$$

Hence, we have:

$$t \leq R^2/\delta^2 \tag{16}$$

□

## References

[Bergstra *et al.*, 2010] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, page 3. Austin, TX, 2010.

[Collins *et al.*, 2008] Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *J. Mach. Learn. Res. (JMLR)*, 9:1775–1822, 2008.

[Collins, 2002] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002.

[Crammer and Singer, 2003] Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

[Gao *et al.*, 2010] Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 358–366, 2010.

[Graves *et al.*, 2005] Alex Graves, Santiago Fernndez, and Jrgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In Wlodzislaw Duch, Janusz Kacprzyk, Erkki Oja, and Slawomir Zadrozny, editors, *ICANN (2)*, volume 3697 of *Lecture Notes in Computer Science*, pages 799–804. Springer, 2005.

[Hart *et al.*, 1968] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost path. *IEEE Trans. On System Science and Cybernetics*, SSC-4(2):100–107, 1968.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Juergen Schmidhuber. Long Short-Term Memory. 9(8):1735–1780, 1997.

[Huang and Chiang, 2005] Liang Huang and David Chiang. Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technology*, pages 53–64, 2005.

[Lafferty *et al.*, 2001] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*, pages 282–289, 2001.

[Le and Zuidema, 2014] Phong Le and Willem Zuidema. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 729–739, 2014.

[McDonald *et al.*, 2005] Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 91–98, 2005.

[Sun and Tsujii, 2009] Xu Sun and Jun'ichi Tsujii. Sequential labeling with latent variables: An exact inference algorithm and its efficient approximation. In *Proceedings of EACL'09*, pages 772–780, 2009.

[Sun *et al.*, 2008] Xu Sun, Louis-Philippe Morency, Daisuke Okanohara, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. Modeling latent-dynamic in shallow parsing: A latent conditional model with imrpoved inference. In *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*, pages 841–848, 2008.

[Sun *et al.*, 2009] Xu Sun, Takuya Matsuzaki, Daisuke Okanohara, and Jun'ichi Tsujii. Latent variable perceptron algorithm for structured classification. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1236–1242, 2009.

[Sun *et al.*, 2010] Xu Sun, Jianfeng Gao, Daniel Micol, and Chris Quirk. Learning phrase-based spelling error models from clickthrough data. In *ACL 2010, Proceedings of the*

*48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 266–274, 2010.

[Sun *et al.*, 2012] Xu Sun, Houfeng Wang, and Wenjie Li. Fast online training with frequency-adaptive learning rates for chinese word segmentation and new word detection. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 253–262, 2012.

[Sun, 2014] Xu Sun. Structure regularization for structured prediction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2402–2410. 2014.

[Yu *et al.*, 2013] Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. Max-violation perceptron and forced decoding for scalable mt training. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1112–1123, 2013.