Unsupervised Neural Dependency CRF Autoencoder (almost)

Sophia Sklaviadis [1]

[1] Johns Hopkins University

Abstract

We would like to combine Cai et al. [2017]'s unsupervised dependency CRF with Zhang et al. [2020]'s supervised neural parametrization of tree CRF models, neuralizing the former model with Rush [2020]'s pytorch-struct library. Cai et al. [2017] extend to latent dependency trees Ammar et al. [2014]'s CRF autoencoder for latent linear chains (applied to part of speech sequences). Cai et al. [2017]'s non-neural model of latent syntactic dependence relations is estimated in a an unsupervised way through expectation maximization (EM). Zhang et al. [2020] describe a neural parametrization of a tree CRF model for supervised parsing. Combining Cai et al. [2017]'s use of EM and Zhang et al. [2020]'s neural parametrization in an unsupervised neural dependency CRF autoencoder leads to an unsupervised model of neural grammar that has as far as we know not yet been implemented. The unsupervised neural dependency CRF autoencoder is comparable to Yang et al. [2020]'s unsupervised neural dependency model with valence, where grammatical relations are more loosely parametrized through several Categorical distributions rather than through the globally normalized CRF. For experiments we plan to use Universal Dependency trees primarily in Ancient Greek, English, German, and Modern Greek. Aside from a useful exercise, the neural dependency CRF can be used as the variational distribution in more complicated models with a transition-based decoder.

The (usual) Problem

Given an observed sequence $x = (x_1, ..., x_N)$, we want to predict a corresponding latent structure $y = (y_1, ..., y_N)$. We may assume either

1 a sequential "chain" latent structure with first-order Markov dependencies among the elements of y,

or

2 a general dependency tree with first-order Markov dependencies, where each y_i is a pair $< t_i, h_i >$ of the parent token of x_i i.e. t_i and its index in the sequence x i.e. h_i.

Autoencoder Framework

Following Ammar et al. [2014] (and Cai et al. [2017], Zhang et al. [2017], Han et al. [2020]), the model introduces a new observed variable $\hat{x}$ which represents a reconstruction of the input x (e.g. identity, Brown cluster, word embeddings), and assumes that x and $\hat{x}$ are conditionally independent given y

$$p(\hat{x}, y|x) = p_\lambda(y|x)p_\theta(\hat{x}|y) \quad . \tag{1}$$

We think of $p_\lambda(y|x)$ as an **encoder**, and of $p_\theta(\hat{x}|x)$ as a **decoder**. We define probability of a latent structure given an observed sequence as a CRF:

$$p_\lambda(y|x) = \frac{e^{\phi(x,y)}}{Z(x) \equiv \sum_y e^{\phi(x,y)}} \quad . \tag{2}$$

Our decoder is a set of categorical distributions which represent the probability of independently generating individual reconstruction elements $\hat{x}_n$ conditioned on y_n. (Alternative distributions could be used for different forms of the reconstruction such as embeddings.)

As in simple latent variable models, reconstruction-based autoencoder models are most easily estimated via direct marginal likelihood optimization.

We compare direct marginal likelihood optimization of various models with variations on the Expecation Maximization (EM) algorithm involving analytical forms of the θ estimators.

Objective

Based on a set of M training sentences $\{x_{1:N}^{(m)}\}_{m=1}^M$, the AE marginal log likelihood is $\sum_{m=1}^M \log p(\hat{x}^{(m)}|x^{(m)})$, where

$$\begin{aligned} p(\hat{x}|x) &= \sum_y p(\hat{x}, y|x) = \sum_y p_\lambda(y|x)p_\theta(\hat{x}|y, x) = \sum_y \frac{e^{\phi(x,y)}}{Z(x)} \cdot \prod_n p(\hat{x}_n|y_n) \\ &= \sum_y \frac{e^{\phi(x,y)+\sum_t \log p(\hat{x}_n|y_n)}}{Z(x)} := \sum_y \frac{e^{s(x,y)}}{Z(x)} = \frac{U(x)}{Z(x)} \end{aligned} \tag{3}$$

i.e. $U(x) = \sum_y e^{s(x,y)}$, $Z(x) = \sum_y e^{\phi(x,y)}$, and we minimize $loss = -(\log U - \log Z)$.

Clarification: What is $U(x)$? It is the partition function of another CRF, namely

$$\frac{e^{s(x,y)}}{\sum_y e^{s(x,y)}} = \frac{e^{s(x,y)}}{U(x)} = \frac{\frac{e^{s(x,y)}}{Z(x)}}{\sum_y \frac{e^{s(x,y)}}{Z(x)}} = \frac{\frac{e^{\phi(x,y)}}{Z(x)}\prod_n p(\hat{x}_n|y_n)}{\sum_y \frac{e^{\phi(x,y)}}{Z(x)}\prod_n p(\hat{x}_n|y_n)} = \frac{p(\hat{x}, y|x)}{\sum_y p(\hat{x}, y|x)} = p(y|\hat{x}, x) \quad . \tag{4}$$

Learning

We use Rush [2020]'s pytorch-struct library to compute the partition functions $Z(x)$ and $U(x)$ of the posteriors $p(y|x)$ and $p(y|x, \hat{x})$ respectively, and consider alternative optimization scenarios:

1 **Direct maximization of the marginal log likelihood:** Using autodiff directly on $\log p(\hat{x}|x)$ amounts to gradient optimization of the expected value of $\log p(\hat{x}, y|x)$ with respect to the posterior $p(y|\hat{x}, x)$, which is the familiar E-step lower bound on the marginal log likelihood $p(\hat{x}|x)$.

2 **Iterative EMs:**

- First, we optimize wrt to λ as in direct max of the marginal likelihood above, using autodiff on $p(\hat{x}|x)$ to obtain λ^{new}, after backpropagating $\nabla_\lambda \underset{p(y|\hat{x},x;\lambda^{old})}{\mathbb{E}} \log p(\hat{x}, y|x)$.
- Then, we optimize wrt to θ: We compute the E-step posterior $p(y|\hat{x}, x; \lambda^{new}, \theta^{old})$, and use it to solve analytically $\nabla_\theta \underset{p(y|\hat{x},x;\lambda^{new},\theta^{old})}{\mathbb{E}} \log p(\hat{x}, y|x; \lambda^{new}, \theta) = 0 \implies \underset{p(y|\hat{x},x;\lambda^{new},\theta^{old})}{\mathbb{E}} \nabla_\theta \log p(\hat{x}, |y, \theta) = 0$. For $c = 1, ..., C$, and $v = 1, ..., V$

$$\hat{\theta}_{cv} = \frac{\sum_{m=1}^M \sum_{n=1}^N p(y_n^{(m)} = c|x^{(m)}, \theta^{old}, \lambda^{new})\mathbb{1}\{x_n^{(m)} = v\}}{\sum_{m=1}^M \sum_{n=1}^N p(y_n^{(m)} = c|x^{(m)}, \theta^{old}, \lambda^{new})} \quad . \tag{5}$$

Viterbi Objective

Instead of summing over all the possible values of the latent y in (3), we can use the Viterbi likelihood $p(\hat{x}, y^*|x)$ where

$$y^* = \underset{y}{\operatorname{argmax}} \log p(\hat{x}, y|x) \tag{6}$$

and for optimization wrt θ with an analytic M-step

$$\hat{\theta}_{cv} = \frac{\sum_{m=1}^M \sum_{n=1}^N \mathbb{1}\{y_n^{*(m)} = c|x^{(m)}, \theta^{old}, \lambda^{new}\}\mathbb{1}\{x_n^{(m)} = v\}}{\sum_{m=1}^M \sum_{n=1}^N \mathbb{1}\{y_n^{*(m)} = c|x^{(m)}, \theta^{old}, \lambda^{new}\}} \quad . \tag{7}$$

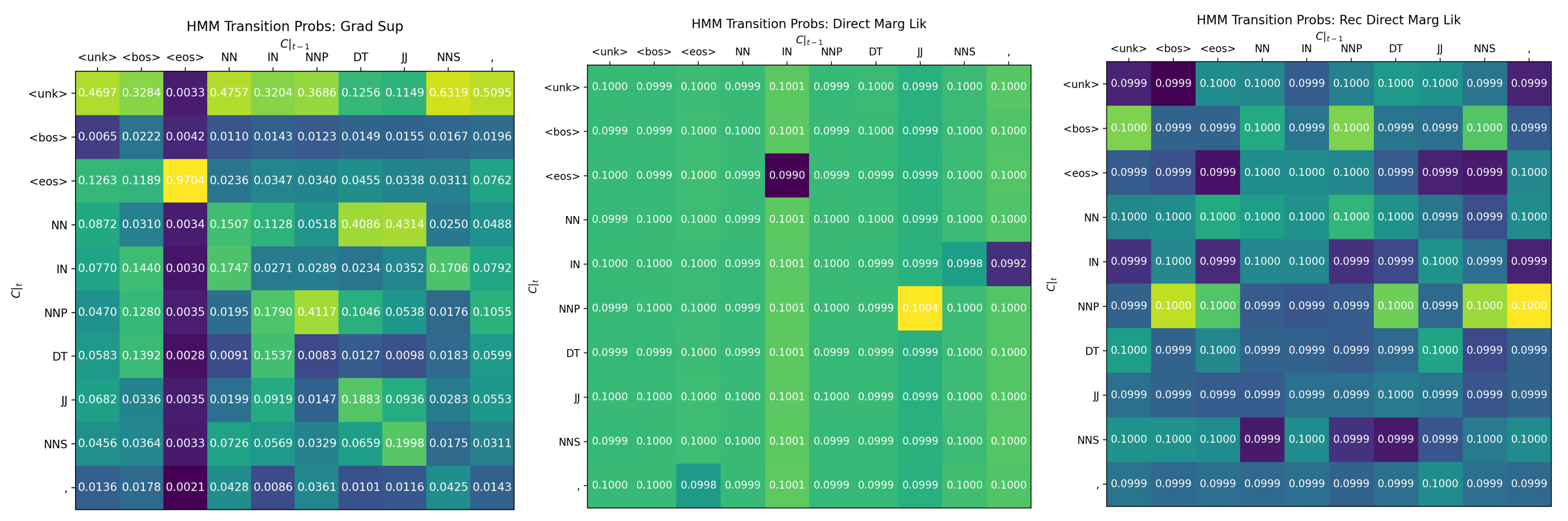
Preliminary Results

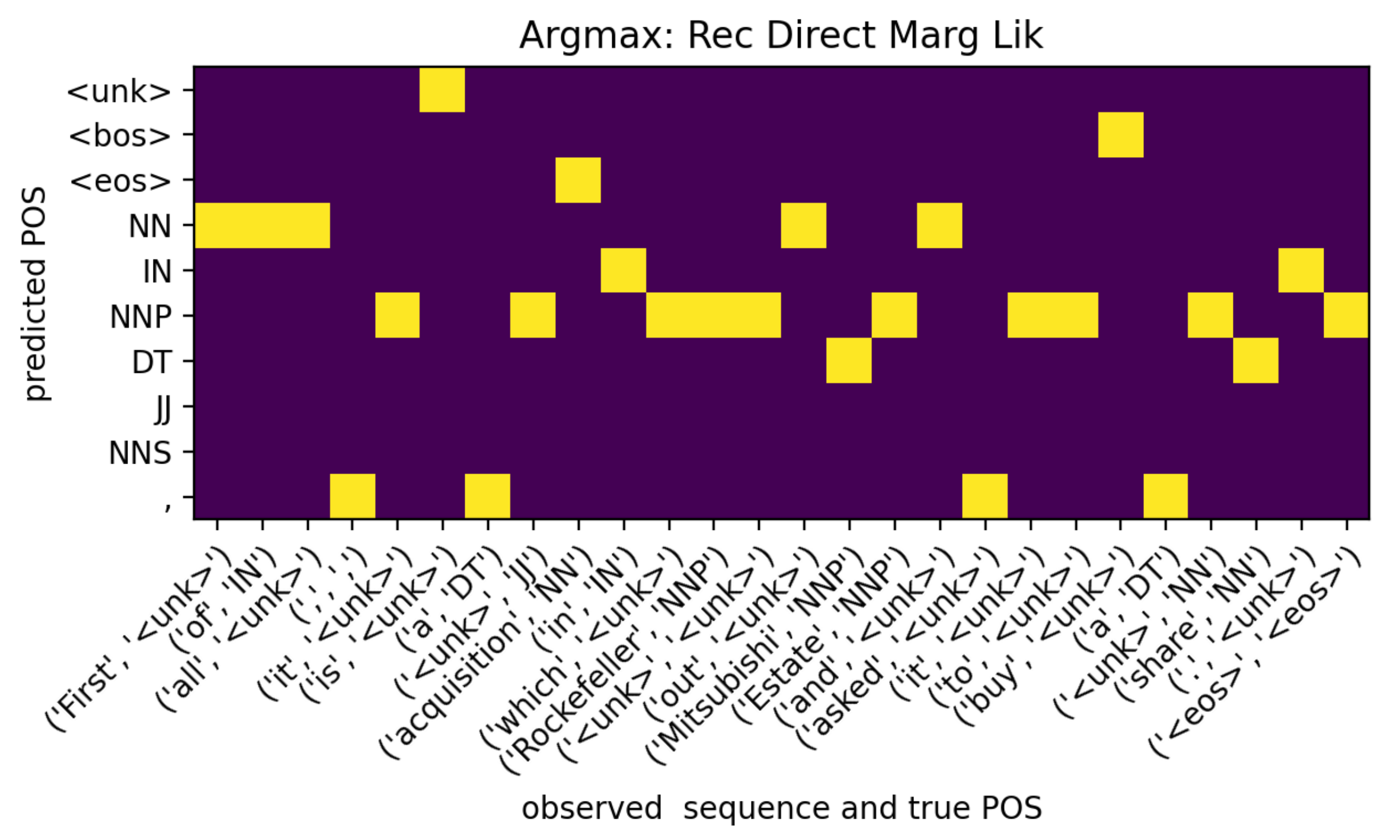
Model	Supervised	Direct Marg (EM)	Rec Direct Marg	Rec EMs
HMM-anal	0.1614	(0.7920)	na	na
HMM-grad	0.2621	0.7569	0.6781	tbd
lin-chain-CRF-1hot	0.3666	0.7778	**0.5109**	?0.8623
lin-chain-CRF-bert	?0.2203	?0.8870	?0.8813	tbd

For reconstruction models we compare $\operatorname{argmax} p(y|x, \hat{x})$ with labeled test data y. For supervised and simple latent variable models we compare $p(y|x)$ with labels. We report *inaccuracy* rate i.e. # incorrect edges predicted/ # total predicted. Hyperparameter settings are variable (? indicates lack of tunning), and these are the best scores from preliminary experiments on CPU with toy-size data: 1174 train sentences, 45 held out test sentences from WSJ English. The format is transferable to GPU. Code: `https://github.com/peithous/nlp-pytorch-struct`.

- We replicate a version of Ammar et al. [2014] (underlined) where if $p_\lambda(y|x)$ is a CRF instead of an HMM parsing accuracy is better.
- Neuralization in the sense of using pretrained embeddings is a straightforward extension. Zhang et al. [2020] use native BiLSTM states for word and character embeddings, which is a pre-training independent neural parametrization.

Images





References

W. Ammar, C. Dyer, and N. A. Smith. Conditional random field autoencoders for unsupervised structured prediction. *Advances in Neural Information Processing Systems*, 27, 2014.

J. Cai, Y. Jiang, and K. Tu. CRF autoencoder for unsupervised dependency parsing. *arXiv preprint arXiv:1708.01018*, 2017.

W. Han, Y. Jiang, H. T. Ng, and K. Tu. A survey of unsupervised dependency parsing. *arXiv preprint arXiv:2010.01535*, 2020.

A. M. Rush. Torch-struct: Deep structured prediction library. *arXiv preprint arXiv:2002.00876*, 2020.

S. Yang, Y. Jiang, W. Han, and K. Tu. Second-order unsupervised neural dependency parsing. *arXiv preprint arXiv:2010.14720*, 2020.

X. Zhang, Y. Jiang, H. Peng, K. Tu, D. Goldwasser, et al. Semi-supervised structured prediction with neural crf autoencoder. Association for Computational Linguistics (ACL), 2017.

ssklavi1@jh.edu