Semi-Supervised Structured Prediction with Neural CRF Autoencoder

Xiao Zhang[†], Yong Jiang[‡], Hao Peng[†], Kewei Tu[‡] and Dan Goldwasser[†]

[†]Department of Computer Science, Purdue University, West Lafayette, USA
{zhang923, pengh, dgoldwas}@cs.purdue.edu
[‡]School of Information Science and Technology, ShanghaiTech University, Shanghai, China
{jiangyong, tukw}@shanghaitech.edu.cn

Abstract

In this paper we propose an end-to-end neural CRF autoencoder (NCRF-AE) model for semi-supervised learning of sequential structured prediction problems. Our NCRF-AE consists of two parts: an encoder which is a CRF model enhanced by deep neural networks, and a decoder which is a generative model trying to reconstruct the input. Our model has a unified structure with different loss functions for labeled and unlabeled data with shared parameters. We developed a variation of the EM algorithm for optimizing both the encoder and the decoder simultaneously by decoupling their parameters. Our experimental results over the Part-of-Speech (POS) tagging task on eight different languages, show that the NCRF-AE model can outperform competitive systems in both supervised and semi-supervised scenarios.

1 Introduction

The recent renaissance of deep learning has led to significant strides forward in several AI fields. In Natural Language Processing (NLP), characterized by highly structured tasks, promising results were obtained by models that combine deep learning methods with traditional structured learning algorithms (Chen and Manning, 2014; Durrett and Klein, 2015; Andor et al., 2016; Wiseman and Rush, 2016). These models combine the strengths of neural models, that can score local decisions using a rich non-linear representation, with efficient inference procedures used to combine the local decisions into a coherent global decision. Among these models, neural variants of the Conditional Random Fields (CRF) model (Lafferty et al., 2001) are especially popular. By replacing the linear potentials with non-linear potential using neural networks these models were able to improve performance in several structured prediction tasks (Andor et al., 2016; Peng and Dredze, 2016; Lample et al., 2016; Ma and Hovy, 2016; Durrett and Klein, 2015).

Despite their promise, wider adoption of these algorithms for new structured prediction tasks can be difficult. Neural networks are notoriously susceptible to over-fitting unless large amounts of training data are available. This problem is exacerbated in the structured settings, as accounting for the dependencies between decisions requires even more data. Providing it through manual annotation is often a difficult labor-intensive task.

In this paper we tackle this problem, and propose an end-to-end neural CRF autoencoder (NCRF-AE) model for semi-supervised learning on sequence labeling problems.

An autoencoder is a special type of neural net, modeling the conditional probability $P(\hat{X}|X)$, where X is the original input to the model and $\hat{X}$ is the reconstructed input (Hinton and Zemel, 1994). Autoencoders consist of two parts, an *encoder* projecting the input to a hidden space, and a *decoder* reconstructing the input from it.

Traditionally, autoencoders are used for generating a compressed representation of the input by projecting it into a dense low dimensional space. In our setting the hidden space consists of discrete variables that comprise the output structure. These generalized settings are described in Figure 1a. By definition, it is easy to see that the encoder (lower half in Figure 1a) can be modeled by a discriminative model describing $P(Y|X)$ directly, while the decoder (upper half in Figure 1a) naturally fits as a generative model, describing $P(\hat{X}|Y)$, where Y is the label. In our model, illustrated in Figure 1b, the encoder is a CRF model with neural networks

as its potential extractors, while the decoder is a generative model, trying to reconstruct the input.

Our model carries the merit of autoencoders, which can exploit valuable information from unlabeled data. Recent works (Ammar et al., 2014; Lin et al., 2015) suggested using an autoencoder with a CRF model as an encoder in an unsupervised setting. We significantly expand on these works and suggest the following contributions:

1. We propose a unified model seamlessly accommodating both unlabeled and labeled data. While past work focused on unsupervised structured prediction, neglecting the discriminative power of such models, our model easily supports learning in both fully supervised and semi-supervised settings. We developed a variation of the Expectation-Maximization (EM) algorithm, used for optimizing the encoder and the decoder of our model simultaneously.

2. We increase the expressivity of the traditional CRF autoencoder model using neural networks as the potential extractors, thus avoiding the heavy feature engineering necessary in previous works.

Interestingly, our model's predictions, which unify the discriminative neural CRF encoder and the generative decoder, have led to an improved performance over the highly optimized neural CRF (NCRF) model alone, even when trained in the supervised settings over the same data.

3. We demonstrate the advantages of our model empirically, focusing on the well-known Part-of-Speech (POS) tagging problem over 8 different languages, including low resource languages. In the supervised setting, our NCRF-AE outperformed the highly optimized NCRF. In the semi-supervised setting, our model was able to successfully utilize unlabeled data, improving on the performance obtained when only using the labeled data, and outperforming competing semi-supervised learning algorithms.

Furthermore, our newly proposed algorithm is directly applicable to other sequential learning tasks in NLP, and can be easily adapted to other structured tasks such as dependency parsing or constituent parsing by replacing the forward-backward algorithm with the inside-outside algorithm. All of these tasks can benefit from semi-supervised learning algorithms.[1]

[1]Our code and experimental set up will be available at https://github.com/cosmozhang/NCRF-AE

2 Related Work

Neural networks were successfully applied to many NLP tasks, including tagging (Ma and Hovy, 2016; Mesnil et al., 2015; Lample et al., 2016), parsing (Chen and Manning, 2014), text generation (Sutskever et al., 2011), machine translation (Bahdanau et al., 2015), sentiment analysis (Kim, 2014) and question answering (Andreas et al., 2016). Most relevant to this work are structured prediction models capturing dependencies between decisions, either by modeling the dependencies between the hidden representations of connected decisions using RNN/LSTM (Vaswani et al., 2016; Katiyar and Cardie, 2016), by explicitly modeling the structural dependencies between output predictions (Durrett and Klein, 2015; Lample et al., 2016; Andor et al., 2016), or by combining the two approaches (Socher et al., 2013; Wiseman and Rush, 2016).

In contrast to supervised latent variable models, such as the Hidden Conditional Random Fields in (Quattoni et al., 2007), which utilize additional latent variables to infer for supervised structure prediction, we do not presume any additional latent variables in our NCRF-AE model in both supervised and semi-supervised setting.

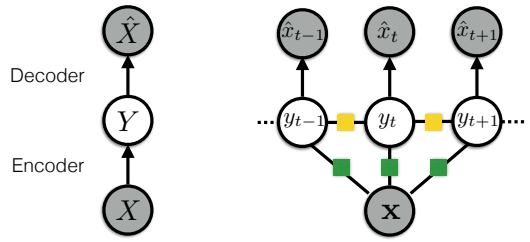
The difficulty of providing sufficient supervision has motivated work on semi-supervised and unsupervised learning for many of these tasks (McClosky et al., 2006; Spitkovsky et al., 2010; Subramanya et al., 2010; Stratos and Collins, 2015; Marinho et al., 2016; Tran et al., 2016), including several that also used autoencoders (Ammar et al., 2014; Lin et al., 2015; Miao and Blunsom, 2016; Kociský et al., 2016; Cheng et al., 2017). In this paper we expand on these works, and suggest a neural CRF autoencoder, that can leverage both labeled and unlabeled data.

3 Neural CRF Autoencoder

In semi-supervised learning the algorithm needs to utilize both labeled and unlabeled data. Autoencoders offer a convenient way of dealing with both types of data in a unified fashion.

A generalized autoencoder (Figure 1a) tries to reconstruct the input $\hat{X}$ given the original input X, aiming to maximize the log probability $P(\hat{X}|X)$ without knowing the latent variable Y explicitly. Since we focus on sequential structured prediction problems, the encoding and decoding processes are no longer for a single data point (x, y) (x if

unlabeled), but for the whole input instance and output sequence $(\boldsymbol{x}, \boldsymbol{y})$ ($\boldsymbol{x}$ if unlabeled). Additionally, as our main purpose in this study is to reconstruct the input with precision, $\hat{\boldsymbol{x}}$ is just a copy of $\boldsymbol{x}$.



(a) A generalized autoencoder.

(b) The neural CRF autoencoder model in this work.

Figure 1: On the left is a generalized autoencoder, of which the lower half is the encoder and the upper half is the decoder. On the right is an illustration of the graphical model of our NCRF-AE model. The yellow squares are interactive potentials among labels, and the green squares represent the unary potentials generated by the neural networks.

As shown in Figure 1b, our NCRF-AE model consists of two parts: the encoder (the lower half) is a discriminative CRF model enhanced by deep neural networks as its potential extractors with encoding parameters $\boldsymbol{\Lambda}$, describing the probability of a predicted sequence of labels given the input; the decoder (the upper half) is a generative model with reconstruction parameters $\boldsymbol{\Theta}$, modeling the probability of reconstructing the input given a sequence of labels. Accordingly, we present our model mathematically as follows:

$$\begin{aligned} P_{\boldsymbol{\Theta},\boldsymbol{\Lambda}}(\hat{\boldsymbol{x}}|\boldsymbol{x}) &= \sum_{\boldsymbol{y}} P_{\boldsymbol{\Theta},\boldsymbol{\Lambda}}(\hat{\boldsymbol{x}}, \boldsymbol{y}|\boldsymbol{x}) \\ &= \sum_{\boldsymbol{y}} P_{\boldsymbol{\Theta}}(\hat{\boldsymbol{x}}|\boldsymbol{y}) P_{\boldsymbol{\Lambda}}(\boldsymbol{y}|\boldsymbol{x}), \end{aligned}$$

where $P_{\boldsymbol{\Lambda}}(\boldsymbol{y}|\boldsymbol{x})$ is the probability given by the neural CRF encoder, and $P_{\boldsymbol{\Theta}}(\hat{\boldsymbol{x}}|\boldsymbol{y})$ is the probability produced by the generative decoder.

When making a prediction, the model tries to find the most probable output sequence by performing the following inference procedure using the Viterbi algorithm:

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} P_{\boldsymbol{\Theta},\boldsymbol{\Lambda}}(\hat{\boldsymbol{x}}, \boldsymbol{y}|\boldsymbol{x}).$$

To clarify, as we focus on POS tagging problems in this study, in the unsupervised setting where the true POS tags are unknown, the labels used for reconstruction are actually the POS tags being induced. The labels induced here are corespoding to the hidden nodes in a generalized autoencoder model.

3.1 Neural CRF Encoder

In a CRF model, the probability of predicted labels $\boldsymbol{y}$, given sequence $\boldsymbol{x}$ as input is modeled as

$$P_{\boldsymbol{\Lambda}}(\boldsymbol{y}|\boldsymbol{x}) = \frac{e^{\Phi(\boldsymbol{x},\boldsymbol{y})}}{Z},$$

where $Z = \sum_{\tilde{\boldsymbol{y}}} e^{\Phi(\boldsymbol{x},\tilde{\boldsymbol{y}})}$ is the partition function that marginalize over all possible assignments to the predicted labels of the sequence, and $\Phi(\boldsymbol{x}, \boldsymbol{y})$ is the scoring function, which is defined as:

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = \sum_t \phi(\boldsymbol{x}, y_t) + \psi(y_{t-1}, y_t).$$

The partition function Z can be computed efficiently via the forward-backward algorithm. The term $\phi(\boldsymbol{x}, y_t)$ corresponds to the score of a particular tag y_t at position t in the sequence, and $\psi(y_{t-1}, y_t)$ represents the score of transition from the tag at position $t-1$ to the tag at position t. In our NCRF-AE model, $\phi(\boldsymbol{x}, y_t)$ is described by deep neural networks while $\psi(y_{t-1}, y_t)$ by a transition matrix. Such a structure allows for the use of distributed representations of the input, for instance, the word embeddings on a continuous vector space (Mikolov et al., 2013).

Typically in our work, $\phi(\boldsymbol{x}, y_t)$ is modeled jointly by a multi-layer perceptron (MLP) that utilizes the word-level information, and a bi-directional long-short term memory (LSTM) neural network (Hochreiter and Urgen Schmidhuber, 1997) that captures the character level information within each word. A bi-directional structure can extract character level information from both directions, with which we expect to catch the prefix and suffix information of words in an end-to-end system, rather than using hand-engineered features. The bi-directional LSTM neural network consumes character embeddings $\boldsymbol{e}_c \in \mathbb{R}^{k_1}$ as input, where k_1 is the dimensionality of the character embeddings. A normal LSTM can be denoted

as:

$$
\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_{ei}\boldsymbol{e}_{c_t} + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i),\\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_{ef}\boldsymbol{e}_{c_t} + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f),\\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_{eo}\boldsymbol{e}_{c_t} + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o),\\
\boldsymbol{g}_t &= Relu(\boldsymbol{W}_{ec}\boldsymbol{e}_{c_t} + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c),\\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t,\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot tanh(\boldsymbol{c}_t),
\end{aligned}
$$

where $\odot$ denotes element-wise multiplication. Then a bi-directional LSTM neural network extends it as follows, by denoting the procedure of generating $\boldsymbol{h}_t$ as $\mathcal{H}$:

$$
\begin{aligned}
\overrightarrow{\boldsymbol{h}}_t &= \mathcal{H}(\boldsymbol{W}_{e\overrightarrow{h}}\boldsymbol{e}_{c_t} + \boldsymbol{W}_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_{\overrightarrow{h}}),\\
\overleftarrow{\boldsymbol{h}}_t &= \mathcal{H}(\boldsymbol{W}_{e\overleftarrow{h}}\boldsymbol{e}_{c_t} + \boldsymbol{W}_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_{\overleftarrow{h}}),
\end{aligned}
$$

where $\boldsymbol{e}_{c_t}$ here is the character embedding for character c in position t in a word.

The inputs to the MLP are word embeddings $\boldsymbol{e}_v \in \mathbb{R}^{k_2}$ for each word v, where k_2 is the dimensionality of the vector, concatenated with the final representation generated by the bi-directional LSTM over the characters of that word: $\boldsymbol{u} = [\boldsymbol{e}_v; \overrightarrow{\boldsymbol{h}}_v; \overleftarrow{\boldsymbol{h}}_v]$. In order to leverage the capacity of the CRF model, we use a word and its context together to generate the unary potential. More specifically, we adopt a concatenation $\boldsymbol{v}_t = [\boldsymbol{u}_{t-(w-1)/2}; \cdots; \boldsymbol{u}_{t-1}; \boldsymbol{u}_t; \boldsymbol{u}_{t+1}; \cdots; \boldsymbol{u}_{t+(w-1)/2}]$ as the inputs to the MLP model, where t denotes the position in a sequence, and w being an odd number indicates the context size. Further, in order to enhance the generality of our model, we add a dropout layer on the input right before the MLP layer as a regularizer. Notice that different from a normal MLP, the activation function of the last layer is no more a softmax function, but a linear function generates the log-linear part $\phi_t(\boldsymbol{x}, y_t)$ of the CRF model:

$$
\begin{aligned}
\boldsymbol{h}_t &= Relu(\boldsymbol{W}\boldsymbol{v}_t + \boldsymbol{b})\\
\phi_t &= \boldsymbol{w}_y^\intercal \boldsymbol{h}_t + b_y.
\end{aligned}
$$

The transition score $\psi(y_{t-1}, y_t)$ is a single scalar representing the interactive potential. We use a transition matrix $\boldsymbol{\Psi}$ to cover all the transitions between different labels, and $\boldsymbol{\Psi}$ is part of the encoder parameters $\boldsymbol{\Lambda}$.

All the parameters in the neuralized encoder are updated when the loss function is minimized via error back-propagation through all the structures of the neural networks and the transition matrix.

The detailed structure of the neural CRF encoder is demonstrated in Fig 2. Note that the MLP layer is also interchangeable with a recurrent neural network (RNN) layer or LSTM layer. But in our pilot experiments, we found a single MLP structure yields better performance, which we conjecture is due to over-fitting caused by the high complexity of those alternatives.

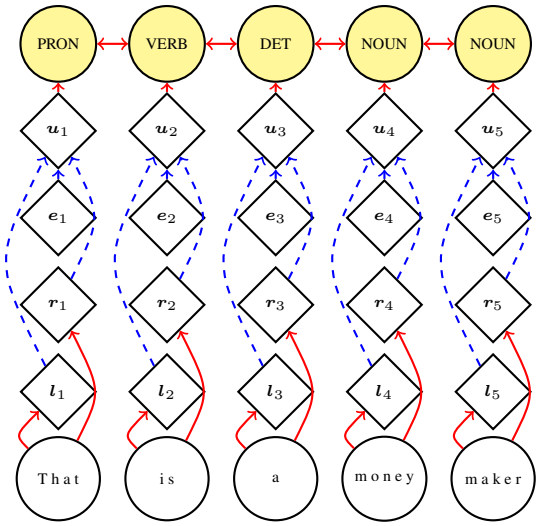


Figure 2: A demonstration of the neural CRF encoder. $\boldsymbol{l}_t$ and $\boldsymbol{r}_t$ are the output of the forward and backward character-level LSTM of the word at position t in a sentence, and $\boldsymbol{e}_t$ is the word-level embedding of that word. $\boldsymbol{u}_t$ is the concatenation of $\boldsymbol{e}_t, \boldsymbol{l}_t$ and $\boldsymbol{r}_t$, denoted by blue dashed arrows.

3.2 Generative Decoder

In our NCRF-AE, we assume the generative process follows several multinomial distributions: each label y has the probability $\theta_{y\to x}$ to reconstruct the corresponding word x, i.e., $P(x|y) = \theta_{y\to x}$. This setting naturally leads to a constraint $\sum_x \theta_{y\to x} = 1$. The number of parameters of the decoder is $|\mathcal{Y}| \times |\mathcal{X}|$. For a whole sequence, the reconstruction probability is $P_{\boldsymbol{\Theta}}(\hat{\boldsymbol{x}}|\boldsymbol{y}) = \prod_t P(\hat{x}_t|y_t)$.

4 A Unified Learning Framework

We first constructed two loss functions for labeled and unlabeled data using the same model. Our model is trained in an on-line fashion: given a labeled or unlabeled sentence, our NCRF-AE optimizes the loss function by choosing the corresponding one. In an analogy to coordinate descent, we optimize the loss function of the NCRF-

AE by alternatively updating the parameters $\mathbf{\Theta}$ in the decoder and the parameters $\mathbf{\Lambda}$ in the encoder. The parameters $\mathbf{\Theta}$ in the decoder are updated via a variation of the Expectation-Maximization (EM) algorithm, and the the parameters $\mathbf{\Lambda}$ in the encoder are updated through a gradient-based method due to the non-convexity of the neuralized CRF. In contrast to the early autoencoder models (Ammar et al., 2014; Lin et al., 2015), our model has two distinctions: First, we have two loss functions to model labeled example and unlabeled examples; Second, we designed a variant of EM algorithm to alternatively learn the parameters of the encoder and the decoder at the same time.

4.1 Unified Loss Functions for Labeled and unlabeled Data

For a sequential input with labels, the complete data likelihood given by our NCRF-AE is

$$\begin{aligned} P_{\mathbf{\Theta},\mathbf{\Lambda}}(\hat{\boldsymbol{x}}, \boldsymbol{y}|\boldsymbol{x}) &= P_{\mathbf{\Theta}}(\hat{\boldsymbol{x}}|\boldsymbol{y}) P_{\mathbf{\Lambda}}(\boldsymbol{y}|\boldsymbol{x}) \\ &= \left[\prod_t P(\hat{x}_t|y_t)\right] \frac{e^{\Phi(\boldsymbol{x},\boldsymbol{y})}}{Z} \\ &= \frac{e^{\sum_t s_t(\boldsymbol{x},\boldsymbol{y})}}{Z}, \end{aligned}$$

where

$$s_t(\boldsymbol{x}, \boldsymbol{y}) = \log P(x_t|y_t) + \phi(\boldsymbol{x}, y_t) + \psi(y_{t-1}, y_t).$$

If the input sequence is unlabeled, we can simply marginalize over all the possible assignment to labels. The probability is formulated as

$$\begin{aligned} P_{\mathbf{\Theta},\mathbf{\Lambda}}(\hat{\boldsymbol{x}}|\boldsymbol{x}) &= \sum_{\boldsymbol{y}} P(\hat{\boldsymbol{x}}, \boldsymbol{y}|\boldsymbol{x}) \\ &= \frac{U}{Z}, \end{aligned}$$

where $U = \sum_{\boldsymbol{y}} e^{\sum_t s_t(\boldsymbol{x},\boldsymbol{y})}$.

Our formulation have two advantages. First, term U is different from but in a similar form as term Z, such that to calculate the probability $P(\hat{\boldsymbol{x}}|\boldsymbol{x})$ for an unlabeled sequence, the forward-backward algorithm to compute the partition function Z can also be applied to compute U efficiently. Second, our NCRF-AE highlights a unified structure of different loss functions for labeled and unlabeled data with shared parameters. Thus during training, our model can address both labeled and unlabeled data well by alternating the loss functions. Using negative log-likelihood as our loss function, if the data is labeled, the loss function is:

$$\begin{aligned} loss_l &= -\log P_{\mathbf{\Theta},\mathbf{\Lambda}}(\hat{\boldsymbol{x}}, \boldsymbol{y}|\boldsymbol{x}) \\ &= -(\sum_t s_t(\boldsymbol{x}, \boldsymbol{y}) - \log Z) \end{aligned}$$

If the data is unlabeled, the loss function is:

$$\begin{aligned} loss_u &= -\log P_{\mathbf{\Theta},\mathbf{\Lambda}}(\hat{\boldsymbol{x}}|\boldsymbol{x}) \\ &= -(\log U - \log Z). \end{aligned}$$

Thus, during training, based on whether the encountered data is labeled or unlabeled, our model can select the appropriate loss function for learning parameters. In practice, we found for labeled data, using a combination of $loss_l$ and $loss_u$ actually yields better performance.

5 Mixed Expectation-Maximization Algorithm

The Expectation-Maximization (EM) algorithm (Dempster et al., 1977) was applied to a wide range of problems. Generally, it establishes a lower-bound of the objective function by using Jensen's Inequality. It first tries to find the posterior distribution of the latent variables, and then based on the posterior distribution of the latent variables, it maximizes the lower-bound. By alternating expectation (E) and maximization (M) steps, the algorithm iteratively improves the lower-bound of the objective function.

In this section we describe the mixed Expectation-Maximization (EM) algorithm used in our study. Parameterized by the encoding parameters $\mathbf{\Lambda}$ and the reconstruction parameters $\mathbf{\Theta}$, our NCRF-AE consists of the encoder and the decoder, which together forms the log-likelihood a highly non-convex function. However, a careful observation shows that if we fix the encoder, the lower bound derived in the E step, is convex with respect to the reconstruction parameters $\mathbf{\Theta}$ in the M step. Hence, in the M step we can analytically obtain the global optimum of $\mathbf{\Theta}$. In terms of the reconstruction parameters $\mathbf{\Theta}$ by fixing $\mathbf{\Lambda}$, we describe our EM algorithm in iteration t as follows:

In the E-step, we let $Q(\boldsymbol{y}_i) = P(\boldsymbol{y}_i|\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i)$, and treat $\boldsymbol{y_i}$ the latent variable as it is not observable in unlabeled data. We derive the lower-bound of the

log-likelihood using $Q(\boldsymbol{y}_i)$:

$$\sum_i \log P(\hat{\boldsymbol{x}}_i|\boldsymbol{x}_i) = \sum_i \log \sum_{\boldsymbol{y}_i} Q(\boldsymbol{y}_i) \frac{P(\hat{\boldsymbol{x}}_i, \boldsymbol{y}_i|\boldsymbol{x}_i)}{Q(\boldsymbol{y}_i)}$$
$$\geq \sum_i \sum_{\boldsymbol{y}_i} Q(\boldsymbol{y}_i) \log \frac{P(\hat{\boldsymbol{x}}_i, \boldsymbol{y}_i|\boldsymbol{x}_i)}{Q(\boldsymbol{y}_i)},$$

where $Q(\boldsymbol{y}_i)$ is computed using parameters $\boldsymbol{\Theta}^{(t-1)}$ in the previous iteration $t-1$.

In the M-step, we try to improve the aforementioned lower-bound using all examples:

$$\arg\max_{\boldsymbol{\Theta}^{(t)}} \sum_i \sum_{\boldsymbol{y}_i} Q(\boldsymbol{y}_i) \log \frac{P_{\boldsymbol{\Theta}^{(t)}}(\hat{\boldsymbol{x}}_i|\boldsymbol{y}_i) P_{\boldsymbol{\Lambda}}(\boldsymbol{y}_i|\boldsymbol{x}_i)}{Q(\boldsymbol{y}_i)}$$
$$\arg\max_{\boldsymbol{\Theta}^{(t)}} \sum_i \sum_{\boldsymbol{y}_i} Q(\boldsymbol{y}_i) \log P_{\boldsymbol{\Theta}^{(t)}}(\hat{\boldsymbol{x}}_i|\boldsymbol{y}_i) + const$$
$$\arg\max_{\boldsymbol{\Theta}^{(t)}} \sum_{y \to x} \log \theta^{(t)}_{y \to x} \sum_y Q(y) C(y, x)$$
$$\arg\max_{\boldsymbol{\Theta}^{(t)}} \sum_{y \to x} \log \theta^{(t)}_{y \to x} \mathrm{E}_{y \sim Q}[C(y, x)]$$
$$s.t. \sum_x \theta^{(t)}_{y \to x} = 1.$$

In this formulation, $const$ is a constant with respect to the parameters we are updating. $Q(y)$ is the distribution of a label y at any position by marginalizing labels at all other positions in a sequence. By denoting $C(y, x)$ as the number of times that (x, y) co-occurs, $\mathrm{E}_{y \sim Q_{\boldsymbol{\Theta}^{(t-1)}}}[C(y, x)]$ is the expected count of a particular reconstruction at any position, which can also be calculated using Baum-Welch algorithm (Welch, 2003), and can be summed over for all examples in the dataset (In the labeled data, it is just a real count). The algorithm we used to calculate the expected count is described in Algorithm 1. Therefore, it can be shown that the aforementioned global optimum can be calculated by simply normalizing the expected counts. In terms of the encoder's parameters $\boldsymbol{\Lambda}$, they are first updated via a gradient-based optimization before each EM iteration. Based on the above discussion, our Mixed EM Algorithm is presented in Algorithm 2.

Algorithm 1 Obtain Expected Count (T_e)

Require: the expected count table T_e

1: **for** an unlabeled data example $\boldsymbol{x}_i$ **do**
2: Compute the forward messages: $\alpha(y, t) \quad \forall y, t.$ ▷ t is the position in a sequence.
3: Compute the backward messages: $\beta(y, t) \quad \forall y, t.$
4: Calculate the expected count for each x in $\boldsymbol{x}_i$: $P(y_t|x_t) \propto \alpha(y, t) \times \beta(y, t)$.
5: $T_e(x_t, y_t) \leftarrow T_e(x_t, y_t) + P(y_t|x_t)$ ▷ T_e is the expected count table.
6: **end for**

Algorithm 2 Mixed Expectation-Maximization

1: Initialize expected count table T_e using labeled data $\{\boldsymbol{x}, \boldsymbol{y}\}^l_i$ and use it as $\boldsymbol{\Theta}^{(0)}$ in the decoder.
2: Initialize $\boldsymbol{\Lambda}^{(0)}$ in the encoder randomly.
3: **for** t in $epochs$ **do**
4: Train the encoder on labeled data $\{\boldsymbol{x}, \boldsymbol{y}\}^l$ and unlabeled data $\{\boldsymbol{x}\}^u$ to update $\boldsymbol{\Lambda}^{(t-1)}$ to $\boldsymbol{\Lambda}^{(t)}$.
5: Re-initialize expected count table T_e with **0**s.
6: Use labeled data $\{\boldsymbol{x}, \boldsymbol{y}\}^l$ to calculate real counts and update T_e.
7: Use unlabeled data $\{\boldsymbol{x}\}^u$ to compute the expected counts with parameters $\boldsymbol{\Lambda}^{(t)}$ and $\boldsymbol{\Theta}^{(t-1)}$ and update T_e.
8: Obtain $\boldsymbol{\Theta}^{(t)}$ globally and analytically based on T_e.
9: **end for**

This mixed EM algorithm is a combination of the gradient-based approach to optimize the encoder by minimizing the negative log-likelihood as the loss function, and the EM approach to update the decoder's parameters by improving the lower-bound of the log-likelihood.

6 Experiments

6.1 Experimental Settings

Dataset We evaluated our model on the POS tagging task, in both the supervised and semi-supervised learning settings, over eight different languages from the UD (Universal Dependencies) 1.4 dataset (Mcdonald et al., 2013). The task is defined over 17 different POS tags, used across the different languages. We followed the original

	English	French	German	Italian	Russian	Spanish	Indonesian	Croatian
Tokens	254830	391107	293088	272913	99389	423346	121923	139023
Training	12543	14554	14118	12837	4029	14187	4477	5792
Development	2002	1596	799	489	502	1552	559	200
Testing	2077	298	977	489	499	274	297	297

Table 1: Statistics of different UD languages used in our experiments, including the number of tokens, and the number of sentences in training, development and testing set respectively.

UD division for training, development and testing in our experiments. The statistics of the data used in our experiments are described in table 1. The UD dataset includes several low-resource languages which are of particular interest to our semi-supervised model.

Input Representation and Neural Architecture Our model uses word embeddings as input. In our pilot experiments, we compared the performance on the English dataset of the pre-trained embedding from Google News (Mikolov et al., 2013) and the embeddings we trained directly on the UD dataset using the skip-gram algorithm (Mikolov et al., 2013). We found these two types of embeddings yield very similar performance on the POS tagging task. So in our experiments, we used embeddings of different languages directly trained on the UD dataset as input to our model, whose dimension is 200. For the MLP neural network layer, the number of hidden nodes in the hidden layer is 20, which is the same for the hidden layer in the character-level LSTM. The dimension of the character-level embeddings sent into the LSTM layer is 15, which is randomly initialized. In order to incorporate the global information of the input sequence, we set the context window size to 3. The dropout rate for the dropout layer is set to 0.5.

Learning We used ADADELTA (Zeiler, 2012) to update parameters $\mathbf{\Lambda}$ in the encoder, as ADADELTA dynamically adapts learning rate over time using only first order information and has minimal computational overhead beyond vanilla stochastic gradient descent (SGD). The authors of ADADELTA also argue this method appears robust to noisy gradient information, different model architecture choices, various data modalities and selection of hyper-parameters. We observed that ADADELTA indeed had faster convergence than vanilla SGD optimization. In our experiments, we include word embeddings and character embeddings as parameters as well. We used Theano to implement our algorithm, and all the experiments were run on NVIDIA GPUs. To prevent over-fitting, we used the "early-stop" strategy to determine the appropriate number of epochs during training. We did not take efforts to tune those hyper-parameters and they remained the same in both our supervised and semi-supervised learning experiments.

6.2 Supervised Learning

In these settings our Neural CRF autoencoder model had access to the full amount of annotated training data in the UD dataset. As described in Section 5, the decoder's parameters $\mathbf{\Theta}$ were estimated using real counts from the labeled data.

We compared our model with existing sequence labeling models including HMM, CRF, LSTM and neural CRF (NCRF) on all the 8 languages. Among these models, the NCRF can be most directly compared to our model, as it is used as the base of our model, but without the decoder (and as a result, can only be used for supervised learning).

The results, summarized in Table 2, show that our NCRF-AE consistently outperformed all other systems, on all the 8 languages, including Russian, Indonesian and Croatian which had considerably less data compared to other languages. Interestingly, the NCRF consistently came second to our model, which demonstrates the efficacy of the expressivity added to our model by the decoder, together with an appropriate optimization approach.

To better understand the performance difference by different models, we performed error analysis, using an illustrative example, described in Figure 3.

In this example, the LSTM incorrectly predicted the POS tag of the word "*search*" as a verb, instead of a noun (part of the NP "*nice search engine*"), while predicting correctly the preceding word, "*nice*", as an adjective. We attribute the error to LSTM lacking an explicit output transition scoring function, which would penalize the ungrammatical transition between "ADJ" and "VERB".

Models	English	French	German	Italian	Russian	Spanish	Indonesian	Croatian
HMM	86.28%	91.23%	85.59%	92.03%	79.82%	91.31%	89.40%	86.98%
CRF	89.96%	93.40%	86.83%	94.07%	83.38%	91.47%	88.63%	86.90%
LSTM	90.50%	94.16%	88.40%	94.96%	84.87%	93.17%	89.42%	88.95%
NCRF	91.52%	95.07%	90.27%	96.20%	93.37%	93.34%	92.32%	93.85%
NCRF-AE	**92.50%**	**95.28%**	**90.50%**	**96.64%**	**93.60%**	**93.86%**	**93.96%**	**94.32%**

Table 2: Supervised learning accuracy of POS tagging on 8 UD languages using different models

Models	English	French	German	Italian	Russian	Spanish	Indonesian	Croatian
NCRF (OL)	88.01%	93.38%	90.43%	91.75%	86.63%	91.22%	88.35%	86.11%
NCRF-AE (OL)	88.41%	93.69%	90.75%	92.17%	87.82%	91.70%	89.06%	87.92%
HMM-EM	79.92%	88.15%	77.01%	84.57%	72.96%	86.77%	83.61%	77.20%
NCRF-AE (HEM)	86.79%	92.83%	89.78%	90.68%	86.39%	91.30%	88.86%	86.55%
NCRF-AE	**89.43%**	**93.89%**	**90.99%**	**92.85%**	**88.93%**	**92.17%**	**89.41%**	**89.14%**

Table 3: Semi-supervised learning accuracy of POS tagging on 8 UD languages. HEM means hard-EM, used as a self-training approach, and OL means only 20% of the labeled data is used and no unlabeled data is used.

Text	*Google*	*is*	*a*	*nice*	*search*	*engine*	.
Gold	PROPN	VERB	DET	ADJ	NOUN	NOUN	PUNCT
NCRF-AE	PROPN	VERB	DET	ADJ	NOUN	NOUN	PUNCT
NCRF	NOUN	VERB	DET	ADJ	NOUN	NOUN	PUNCT
LSTM	PROPN	VERB	DET	ADJ	VERB	NOUN	PUNCT

Figure 3: An example from the test set to compare the predicted results of our NCRF-AE model, the NCRF model and the LSTM model.

The NCRF, which does score such transitions, correctly predicted that word. However, it incorrectly predicted "*Google*" as a noun rather than a proper-noun. This is a subtle mistake, as the two are grammatically and semantically similar. This mistake appeared consistently in the NCRF results, while NCRF-AE predictions were correct.

We attribute this success to the superior expressivity of our model: The prediction is done jointly by the encoder and the decoder, as the reconstruction decision is defined over all output sequences, picking the jointly optimal sequence. From another perspective, our NCRF-AE model is a combination of discriminative and generative models, in that sense the decoder can be regarded as a soft constraint that supplements the encoder. Such that, the decoder performs as a regularizer to check-balance the choices made by the encoder.

6.3 Semi-supervised Learning

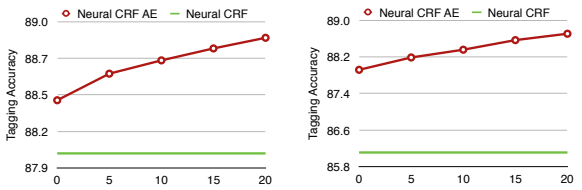
In the semi-supervised settings we compared our models with other semi-supervised structured prediction models. In addition, we studied how varying the amount of unlabeled data would change the performance of our model.

As described in Sec. 5, the decoder's parameters $\mathbf{\Theta}$ are initialized by the labeled dataset using real counts and updated in training.

6.3.1 Varying Unlabeled Data Proportion

We first experimented with varying the proportion of unlabeled data, while fixing the amount of labeled data. We conducted these experiments over two languages, English and low-resource language Croatian. We fixed the proportion of labeled data at 20%, and gradually added more unlabeled data from 0% to 20% (from full supervision to semi-supervision). The unlabeled data was sampled from the same dataset (without overlapping with the labeled data), with the labels removed. The results are shown in Figure 4.

The left most point of both sub-figures is the accuracy of fully supervised learning with 20% of the whole data. As we can observe, the tagging accuracy increased as the proportion of unlabeled data increased.



(a) English (b) Croatian

Figure 4: UD English and Croatian POS tagging accuracy versus increasing proportion of unlabeled sequences using 20% labeled data. The green straight line is the performance of the neural CRF, trained over the labeled data.

6.3.2 Semi-supervised POS Tagging on Multiple Languages

We compared our NCRF-AE model with other semi-supervised learning models, including the HMM-EM algorithm and the hard-EM version of our NCRF-AE. The hard EM version of our model can be considered as a variant of self-training, as it infers the missing labels using the current model in the E-step, and uses the real counts of these labels to update the model in the M-step. To contextualize the results, we also provide the results of the NCRF model and the supervised version our NCRF-AE model trained on 20% of the data. We set the proportion of labeled data to 20% for each language and set the proportion of unlabeled data to 50% of the dataset. There was no overlap between labeled and unlabeled data.

The results are summarized in Table 3. Similar to the supervised experiments, the supervised version of our NCRF-AE, trained over 20% of the labeled data, outperforms the NCRF model. Our model was able to successfully use the unlabeled data, leading to improved performance in all languages, over both the supervised version of our model, as well as the HMM-EM and Hard-EM models that were also trained over both the labeled and unlabeled data.

6.3.3 Varying Sizes of Labeled Data on English

As is known to all, semi-supervised approaches tend to work well when given a small size of labeled training data. But with the increase of labeled training data size, we might get diminishing effectiveness. To verify this conjecture, we conducted additional experiments to show how varying sizes of labeled training data affect the effectiveness of our NCRF-AE model. In these experiments, we gradually increased the proportion of labeled data, and in accordance decreased the proportion of unlabeled data.

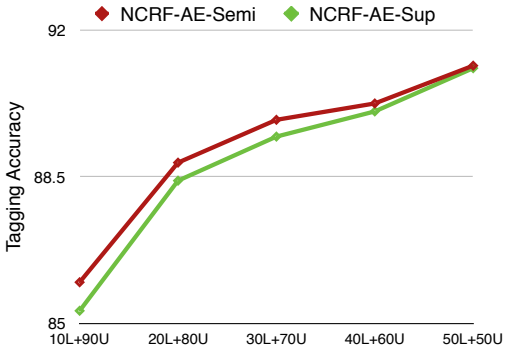


Figure 5: Performance of the NCRF-AE model on different proportion of labeled and unlabeled data. The green line shows the results on only labeled data, and the red line on both labeled and unlabeled data. The difference between the red line and the green line are gradually vanishing.

The results of these experiments are demonstrated in Figure 5. As we speculated, we observed diminishing effectiveness when increasing the proportion of labeled data in training.

7 Conclusion

We proposed an end-to-end neural CRF autoencoder (NCRF-AE) model for semi-supervised sequence labeling. Our NCRF-AE is an integration of a discriminative model and generative model which extends the generalized autoencoder by using a neural CRF model as its encoder and a generative decoder built on top of it. We suggest a variant of the EM algorithm to learn the parameters of our NCRF-AE model.

We evaluated our model in both supervised and semi-supervised scenarios over multiple languages, and show it can outperform other supervised and semi-supervised methods. Additional experiments suggest how varying sizes of labeled training data affect the effectiveness of our model.

These results demonstrate the strength of our model, as it was able to utilize the small amount of labeled data and exploit the hidden information from the large amount of unlabeled data, without additional feature engineering which is often needed in order to get semi-supervised and weakly-supervised systems to perform well. The superior performance on the low resource language also suggests its potential in practical use.

References

Waleed Ammar, Chris Dyer, and Noah A Smith. 2014. Conditional random field autoencoders for unsupervised structured prediction. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 1–12.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 2442–2452.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 1545–1554.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. International Conference on Learning Representation (ICLR)*.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Yong Cheng, Yang Liu, and Wei Xu. 2017. Maximum reconstruction estimation for generative latent-variable models. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B*, 39(1):1–38.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. pages 302–312.

Geoffrey E Hinton and Richard S Zemel. 1994. Autoencoders, minimum description length and helmholtz free energy. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 3–10.

Sepp Hochreiter and J Urgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Arzoo Katiyar and Claire Cardie. 2016. Investigating lstms for joint extraction of opinion entities and relations. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 919–929, Berlin, Germany.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 1746–1751.

Tomás Kociský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 1078–1087.

John D Lafferty, Andrew McCallum, and Fernando C N Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 282–289, San Francisco, CA, USA.

Guillaume Lample, Miguel Ballesteros, Kazuya Kawakami, Sandeep Subramanian, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 1–10.

Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. 2015. Unsupervised pos induction with word embeddings. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 1311–1316.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 1064–1074, Berlin, Germany.

Z. Marinho, A. F. T. Martins, S. B. Cohen, and N. A. Smith. 2016. Semi-supervised learning of sequence models with the method of moments. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 152–159, Stroudsburg, PA, USA.

Ryan Mcdonald, Joakim Nivre, Yvonne Quirmbach-brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Tckstrm, Claudia Bedini, Nria Bertomeu, and Castell Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu, and G. Zweig. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 319–328, Austin, Texas.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 1–9.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for chinese social media with word segmentation representation learning. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 149–155.

Ariadna Quattoni, Sybor Wang, Louis-Philippe Morency, Michael Collins, and Trevor Darrell. 2007. Hidden conditional random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(10):1848–1852.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 455–465, Sofia, Bulgaria.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How less is more in unsupervised dependency parsing. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 751–759.

Karl Stratos and Michael Collins. 2015. Simple semi-supervised pos tagging. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

Amarnag Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 1017–1024, New York, NY, USA.

Ke M. Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. Unsupervised neural hidden markov models. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 63–71, Austin, TX.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with lstms. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 232–237.

Lloyd R Welch. 2003. Hidden markov models and the baum-welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4):1,10–13.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*, pages 1296–1306, Austin, Texas.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701.