Training for Gibbs Sampling on Conditional Random Fields with Neural Scoring Factors

Sida Gao
Carnegie Mellon University
sidag@alumni.cmu.edu

Matthew R. Gormley
Carnegie Mellon University
mgormley@cs.cmu.edu

Abstract

Most recent improvements in NLP come from changes to the neural network architectures modeling the text input. Yet, state-of-the-art models often rely on simple approaches to model the label space, e.g. bigram Conditional Random Fields (CRFs) in sequence tagging. More expressive graphical models are rarely used due to their prohibitive computational cost. In this work, we present an approach for efficiently training and decoding hybrids of graphical models and neural networks based on Gibbs sampling. Our approach is the natural adaptation of SampleRank (Wick et al., 2011) to neural models, and is widely applicable to tasks beyond sequence tagging. We apply our approach to named entity recognition and present a neural skip-chain CRF model, for which exact inference is impractical. The skip-chain model improves over a strong baseline on three languages from CoNLL-02/03. We obtain new state-of-the-art results on Dutch. [1]

1 Introduction

Complex probabilistic graphical models were widely adopted for NLP tasks before the prevalence of deep learning (e.g. the skip-chain CRF of Finkel et al. (2005) and Sutton and Mccallum (2004) for NER). Although modern neural architectures are able to learn much better feature representations (e.g. the contextualized word representations of Peters et al. (2018), Devlin et al. (2018), and Akbik et al. (2019)) than the hand-crafted features used classically in graphical model's log-linear potentials, these advances in feature learning do not negate the need for modeling the output label space.

Consider two contrasting approaches to structured prediction: transition-based models and graphical models. Transition-based models (e.g. the sequence-to-sequence models of Sutskever et al. (2014)) have enjoyed recent success thanks to their ability to have unbounded memory of past transitions when predicting subsequent ones; yet because no conditional independence assumptions are made, inference is typically restricted to (heuristic) greedy search and its variants. By contrast, graphical models make strong conditional independence assumptions, but enjoy a wealth of inference algorithms, both exact and approximate, as a result. Moreover, graphical models readily admit the incorporation of domain knowledge about interactions between the output variables. In this paper, we focus on this latter approach to modeling.

Specifically, we explore conditional random fields (CRFs) (Lafferty et al., 2001) with neural potential functions. Prior state-of-the-art approaches utilizing such models (e.g. CRF-LSTMs) for sequence tagging tasks like named entity recognition (NER) have focused on simple linear-chain CRFs, which only model bi-gram dependencies of adjacent labels (Lample et al., 2016; Peters et al., 2017), and the exact inference can be done in polynomial time with dynamic programming. By contrast, we are motivated by CRFs that do not admit exact inference.

We propose Neural SampleRank, a novel algorithm that is computationally efficient for approximate inference and training of complex CRFs (where exact inference is impractical) with neural factors. The main inspiration of our work is SampleRank (Wick et al., 2011; Zhang et al., 2014), a training algorithm for complex graphical models based on Gibbs sampling, that has been shown to work well with linear factors. We extend SampleRank to work with neural scoring factors. Neural SampleRank enables us to use CRFs that are far more expressive than the linear-chain structures seen in NER models. The loss does *not* require full

[1] The code is available at https://github.com/GaoSida/Neural-SampleRank.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 4999–5011,
November 16–20, 2020.

inference to compute the gradient in training, which makes it more computationally efficient. Comparing with message-passing based algorithms like loopy belief propagation (LBP), Neural SampleRank is conceptually simpler and easier to implement with modern deep learning tools like PyTorch (Paszke et al., 2019). We empirically evaluate Neural SampleRank on the CoNLL-02 and CoNLL-03 NER task on English, German and Dutch (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003). We show that for linear-chain BiLSTM-CRF model (Lample et al., 2016), training with Neural SampleRank achieves competitive F1 score compared with MLE and exact inference. With a new neural skip-chain CRF model trained with Neural SampleRank, we achieve higher F1 on English and German than all existing models that do not use contextualized word embeddings or external labeled data. With contextualized word embeddings, our skip-chain model obtains new state-of-the-art results on Dutch.

2 Related Work

Various approaches have been taken in NLP to combine graphical models and neural architectures. For sequence tagging tasks like NER, it is common to use a linear-chain CRF model (Huang et al., 2015; Lample et al., 2016), for which exact inference can be done in polynomial time with forward-backward. Malaviya et al. (2018) adopt a factorized CRF to model the output space of morphological tagging, and the exact inference is tractable with belief propagation. Ganea and Hofmann (2017) propose a fully connected binary CRF to model mention sequence for entity linking task, and they use loopy belief propagation for approximate inference.

Other approaches have been proposed to adopt expressive graphical models while keeping the inference computationally feasible, but have not been applied to deep neural networks. Steinhardt and Liang (2015) propose to select non-local contexts while keeping the model feasible for exact inference. Finkel et al. (2005) use Gibbs Sampling with simulated annealing for fast approximate inference for models with non-local factors. Sutton and Mccallum (2004) propose a skip-chain CRF for NER learned with loopy belief propagation. SampleRank (Wick et al., 2011; Zhang et al., 2014) propose a new training objective targeted for sampling-based inference which is efficient both in terms of computation cost and task performance. In prior work, Gibbs sampling has been used with deep neural networks for Bayesian posterior inference (Shi et al., 2017; Tran et al., 2016), and sampling from conditional sequence models (Lin and Eisner, 2018). Gibbs sampling was only widely applied to discriminative models before the prevalence of deep learning, and restricted to generative models when used with neural models (Das et al., 2015; Nguyen et al., 2015; Xun et al., 2017). To the best of knowledge, we are the first to use Gibbs sampling to obtain point estimation for neural network graphical model hybrids, for the task of structured prediction.

State-of-the-art approaches for NER all use a simple linear-chain CRF model to model the label space. Neural architectures to learn a better representation of the text input include bi-directional LSTM (Huang et al., 2015; Lample et al., 2016), GRU (Yang et al., 2017) and character CNN (Yang et al., 2017; Peters et al., 2017). A major recent step in the field is contextualized word embedding like ELMo (Peters et al., 2018) , BERT (Devlin et al., 2018) and the character-based Flair (Akbik et al., 2019). However, none of these approaches model longer range context dependencies in the document, limited by the linear-chain structure of the CRF.

3 Neural SampleRank

3.1 CRF with Neural Factors

We use x to denote an input sentence and $y \in \mathcal{Y}(x)$ to denote a structured output for the sentence. $\mathcal{Y}(x)$ is the valid output space for input x. We denote the ground truth output as y^*. The neural CRF can be interpreted as a factor graph that defines the following conditional distribution:

$$p(y|x;\Theta) = \frac{\exp(s(x,y;\Theta))}{\sum_{y' \in \mathcal{Y}(x)} \exp(s(x,y';\Theta))} \quad (1)$$

where $s(x, y; \Theta)$ is a differentiable scoring function parameterized by Θ, given by a factor graph with arbitrary structure and factors defined with neural networks. The goal of inference is to find the output $\hat{y}$ with the highest conditional probability defined in Eq. 1, or equivalently with highest score:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}(x)} s(x,y;\Theta) \quad (2)$$

For many NLP tasks the size of the output space $\mathcal{Y}(x)$ grows exponentially as the length of x increases, which makes computation of the partition

function (i.e. the denominator in Eq. 1) and finding the maximum over $\mathcal{Y}(x)$ (i.e. Eq. 2) hard combinatorial problems. However, with Gibbs sampling, we are able to avoid computing the partition function altogether and make finding an (approximate) maximum feasible in practice.

To avoid repetitive computation for the neural networks, we decompose the scoring function s as:

$$s(x, y; \Theta) = s\big(x, y, f(x; \theta_N); \theta_G\big) \tag{3}$$

where we break down the learnable parameters as $\Theta = \{\theta_N, \theta_G\}$, where θ_N parameterizes the neural network f that constructs a representation of the input, and θ_G parameterizes the CRF that captures dependencies in the output label space. The neural function $f(x; \theta_N)$ is usually expensive to compute but only depends on the input x. On the other hand, after f is evaluated, the score $s(x, y, f; \theta_G)$ is usually very cheap to compute (e.g. look-ups in a factor table). We will leverage these properties to improve computational efficiency.

3.2 Decoding with Gibbs Sampling

To decode a neural CRF model, we find the output that maximizes the scoring function (as shown in Eq. 2) by sampling from the conditional distribution defined in Eq. 1 with Markov Chain Monte Carlo (MCMC). However, finding maximum by sampling from the original distribution is inefficient, and a common practice (Finkel et al., 2005) is to instead sample from this distribution:

$$p(y|x; \Theta, T) \propto \exp\left(\tfrac{1}{T} s(x, y; \Theta)\right) \tag{4}$$

where we introduce a temperature $T \leq 1$ to sharpen the distribution around the region with highest probability density (a smaller T will lead to a sharper peak). In practice we typically design an annealing schedule to gradually decrease T, so that we allow more exploration in the beginning of the Markov Chain, and gradually converge to the region with the highest probability density.

The decoding algorithm is shown in Alg. 1. We conduct decoding with Gibbs sampling, where the proposal distribution q is the conditional distribution of one variable (or a subset of variables) in y^t conditioned on all other variables according to p (defined in Eq. 4).

When decoding with MCMC, the output y may be stuck at a local maxima due to the annealing process, and for each run of MCMC we may end up in a different local maxima. Therefore, we run MCMC decoding for multiple times, then conduct a majority vote for each label. This simple ensemble approach is able to reduce the variance of MCMC decoding and improve prediction accuracy.

Algorithm 1: Decoding with Gibbs Sampling.

Input: $x, \Theta = \{\theta_N, \theta_G\}$
Output: $\hat{y}$

```
Initialize temperature T;
z ← f(x; θ_N);
Randomly initialize output y^0;
for t = 0, ..., M − 1 do
    y^{t+1} ← q(·|x, y^t, z, θ_G, T);
    if s(x, y^{t+1}, z; θ_G) > s(x, ŷ, z; θ_G) then
        ŷ ← y^{t+1}
    T ← anneal(T);
```

3.3 Training with Neural SampleRank

The training algorithm of Neural SampleRank is largely inspired by the SampleRank algorithm proposed by Wick et al. (2011). We adopt a max-margin loss to train the neural CRF scoring function, so that the score of a favorable output is higher than an unfavorable output by a margin. Assume $\omega(y)$ is a metric to measure the quality of a tag sequence y according to the ground truth y^* (e.g. F1 score, or negative Hamming distance). If $\omega(y) > \omega(y')$ then y is considered to have higher quality, and the ground truth $y^* = \arg\max_{y \in \mathcal{Y}} \omega(y)$. Then the margin $\Delta_\omega(y_i, y_j)$ is defined as:

$$\Delta_\omega(y_i, y_j) = \omega(y^+) - \omega(y^-) \tag{5}$$

where $y^+ = \arg\max_{y \in \{y_i, y_j\}} \omega(y)$ and $y^- = \arg\min_{y \in \{y_i, y_j\}} \omega(y)$, thus $\Delta_\omega(y_i, y_j) \geq 0$.

The SampleRank loss is incurred by a pair of outputs y_i, y_j when $\Delta_\omega(y_i, y_j) > 0$, defined as:

$$\ell(y_i, y_j) = [\Delta_\omega(y_i, y_j) - \big(s(y^+, x; \theta) - s(y^-, x; \theta)\big)]_+ \tag{6}$$

The training procedure for Neural SampleRank is shown in Alg. 2. The loss $\ell(\cdot, \cdot)$ is defined in Eq. 6, and $q(\cdot)$ is the proposal distribution for Gibbs sampling.

Computing the loss for Neural SampleRank does not require running full inference of the CRF model, and is instead accumulated over M Gibbs sampling steps. There are two types of loss terms: pairwise loss $\ell(y^t, y^{t-1})$, which is the max-margin loss computed with two consecutive samples y^{t-1}

Algorithm 2: Neural SampleRank Training.

Input: $D = \{(x_1, y_1^*), \ldots, (x_n, y_n^*)\}$
Output: $\Theta = \{\theta_N, \theta_G\}$
Initialize $\Theta = \{\theta_N, \theta_G\}$;
while *not converged* **do**
  foreach $x_i, y_i^* \in D$ **do**
    loss_total $\leftarrow 0$;
    $z \leftarrow f(x_i; \theta_N)$;
    $y^0 \leftarrow$ random_initialize(y_i^*);
    for $t = 1, \ldots, M$ **do**
      $y^t \leftarrow q(\cdot|x_i, y^{t-1}, z, \theta_G)$;
      if $\Delta_\omega(y^t, y^{t-1}) > 0$ **then**
        loss_total += $\ell(y^t, y^{t-1})$;
      if $\Delta_\omega(y^t, y_i^*) > 0$ **then**
        loss_total += $\ell(y^t, y_i^*)$;
    $\Theta \leftarrow$ update(Θ, ∇_Θloss_total) ;

and y^t; and gold loss $\ell(y^t, y_i^*)$, which is computed with the ground truth output y_i^* and a sample y^t. Intuitively, while gold loss helps the model to rank ground truth higher than all incorrect outputs, the pairwise loss ensures the model can correctly rank between two similar outputs. This property is helpful during the sampling based decoding: the predicted output is able to take "guided" steps to gradually move to better quality outputs, even though the initial output might be far from ground truth.

During training, for each example, the initial sample y^0 is taken from random_initialize($\cdot$) in Alg. 2, which randomly copies from the ground truth output y^*. We first uniformly randomly sample a probability u between 0 and 1, then for each label value $y^0[j]$, we copy from $y^*[j]$ with probability u, and take random value with probability $1-u$. This is to simulate different stages of MCMC decoding, in which the samples converge to a high probability density region, i.e. get closer and closer to ground truth.

In Alg. 2, the model is only updated after sampling (not during), and we reinitialize the sample after each model update, therefore we are not breaking detailed balance and the sampler is still proper MCMC. However, since full inference is not needed for training, the Markov Chains in Alg. 2 only have a small number of samples, and do not necessarily converge.

3.4 Comparison with Linear SampleRank

Comparing with the SampleRank algorithm proposed in previous work (Wick et al., 2011; Zhang et al., 2014), Neural SampleRank uses the same pairwise training objective defined on two consecutive examples on the Markov chain. Unlike Wick et al. (2011), we also adopt the gold loss term defined on the ground truth and one sample as done in Zhang et al. (2014), which has empirically shown to be important for model performance.

The key difference between Neural SampleRank and the SampleRank algorithm for CRFs with linear factors is the optimization algorithm. Wick et al. (2011) frames the optimization problem as a saddle point optimization problem and solves it with a stochastic approximation saddle point (SASP) algorithm. On the other hand, Zhang et al. (2014) frames the learning objective as a constrained optimization problem, and solves it with the MIRA algorithm (Crammer and Singer, 2003). Both algorithms rely on the fact that the scoring factors are linear functions, to derive a closed form update for each iteration in training, so neither optimization algorithm works with neural scoring factors. In Neural SampleRank, we reframe the optimization objective as a structured hinge loss (Eq. 6) without constraints, so that we are able to train the neural scoring factors with back-propagation based gradient updates.

3.5 Computational Efficiency

After the decomposition of scoring function in Eq. 3, we take a two-step approach to evaluate the scoring function. As shown in Alg. 1 and Alg. 2, for each input x, we first compute its neural representation $z = f(x; \theta_N)$ before we take any samples. Once the sampling starts, only the output y could change, leaving z, the neural representation of x, unchanged. Therefore, when we take new samples, we only need to recompute the scoring function defined by the non-neural factors of the CRF (parameterized by θ_G). In this way, for each input x, we only need to evaluate the expensive deep neural networks once, and for each additional sample we only need to evaluate the cheap non-neural factors.

In pairwise SampleRank loss $\ell(y^t, y^{t-1})$, the two consecutive Gibbs samples usually only differ in a small subset of the variables in the CRF. We can leverage this fact to sparsify the computation of the score difference between y^t and y^{t-1} (Eq. 6) and its gradient w.r.t the factor scores, by only considering the factors in the CRF that involve the small label subset that has been re-sampled (Wick et al., 2011). This sparse property makes Neural

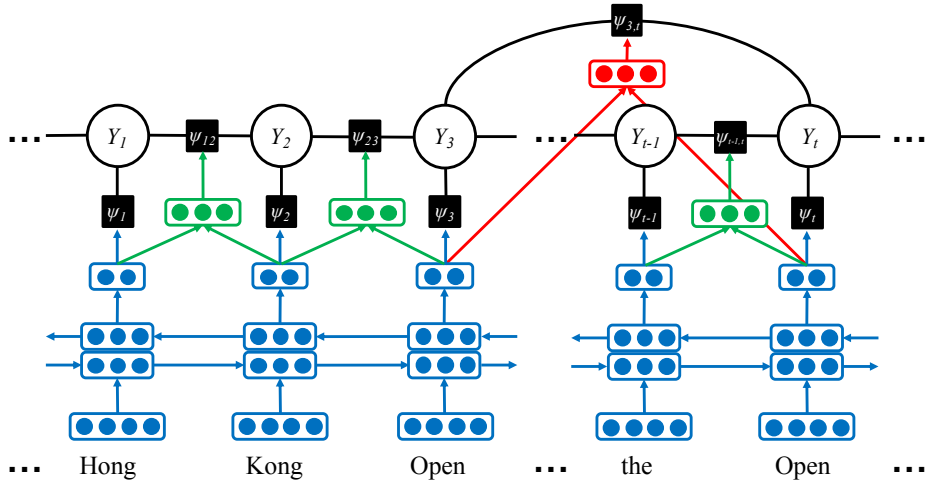


Figure 1: Neural skip-chain CRF. The BiLSTM (shown in blue) computes a vector representation for each input token. These vectors are fed into transition (green) and skip-chain (red) MLPs (i.e. feed-forward layers). The outputs of these neural scoring functions are used to compute unary, transition, and skip-chain factors (black squares). The factors provide a score for each assignment of their neighboring variables (white circles).

SampleRank efficient for complex CRFs when the degree of each node is bounded by a small constant. CRFs can usually satisfy this sparsity condition as they introduce inductive bias by making (conditional) independence assumptions. On the other hand, the gold loss $\ell(y^t, y^*)$ may require evaluating the full CRF as the sample y^t could be far from the ground truth y^*. However, as training progresses, y^t will get sufficiently close to y^* with fewer and fewer samples, resulting in fewer number of factors that need re-evaluation, thus lead to a speed-up for gold loss computation.

4 Neural Skip-Chain CRF for NER

4.1 Base Model

The base model we adopt for NER is a BiLSTM-CRF (Lample et al., 2016), which adopts a multi-layer BiLSTM to learn a representation of the text input, then use a linear-chain CRF to model the dependencies in the output label space. For token inputs of BiLSTM, we use either concatenation of pretrained word embedding and character-CNN word embedding, or contextualized word embedding.

The linear-chain CRF scoring function is:

$$s(y, x; \Theta) = \sum_{1 \le i \le d} \Psi_i(y_i, x; \Theta) + \sum_{1 \le i \le d-1} \Psi_{i,i+1}(y_i, y_{i+1}, x; \Theta) \quad (7)$$

where d is the length of the text input. The model consists of emission factors $\Psi_i(y_i, x; \Theta)$ for each token label y_i, and transition factors $\Psi_{i,i+1}(y_i, y_{i+1}, x; \Theta)$ for each pair of adjacent token labels (i.e. a bi-gram).

The hidden states of the token BiLSTM are treated as a context-aware representation for each token, and are used to parameterize the emission factors in the linear chain CRF. In previous works, the transition factors are learnable scalars shared across all bi-grams, and they do not have any dependencies on the context. In our model, we modify the transition factors to be context dependent: we use feed-forward layers to compute the transition factors with the BiLSTM hidden states of the two tokens in the bi-gram. The parameters of the feed-forward layers are shared among all bi-grams.

4.2 Skip-Chain CRF

Besides bi-gram level label dependencies modeled by the transition factors in linear-chain CRF, we introduce longer range factors that model global dependencies in a sequence tagging task. The design of global factors may be different for each task in order to model the task-specific dependency patterns. In this section we present one approach to design global factors for NER. The resulting neural skip-chain CRF is depicted in Figure 1.

We adopt the same consistency assumption and inductive bias proposed by Finkel et al. (2005) and Sutton and Mccallum (2004): different occurrences of the same token are likely to be labeled in the same way (e.g. they could be recurring references to the same named entity). However, this assump-

tion is not always true, as the labels of a token sequence are context dependent, so we introduce factors to model this uncertainty. On top of the linear-chain CRF, we introduce a skip-chain connection between every pair of recurring capitalized tokens in the same document—named entities are usually capitalized. We denote the set of skip-chain connections that satisfy these conditions as S, then the scoring function for the skip-chain CRF is:

$$s(y, x; \Theta) = \sum_{1 \le i \le d} \Psi_i(y_i, x; \Theta) + \sum_{1 \le i \le d-1} \Psi_{i,i+1}(y_i, y_{i+1}, x; \Theta) + \sum_{(i,j) \in S} \Psi_{i,j}(y_i, y_j, x; \Theta) \quad (8)$$

A skip-chain factor scores all possible labels of the token pair with feed forward layers on token representations constructed by the token BiLSTM. Although the skip-chain factors $\Psi_{i,j}(y_i, y_j, x; \Theta)$ are still second order factors like the transition factors, the token labels y_i, y_j are usually not adjacent, and in most cases are far apart in the document. We can no longer use forward-backward for exact inference. Instead, we use Gibbs sampling to do efficient approximate inference for each document. In our experiments, we employ block Gibbs sampling for token pairs that have a skip-chain connection, so that the model can better leverage long-range context dependency.

5 Experiments

5.1 Dataset and Model Configuration

We evaluate Neural SampleRank for sequence tagging models on CoNLL-02 Dutch (Tjong Kim Sang, 2002), and CoNLL-03 English and German NER datasets (Tjong Kim Sang and De Meulder, 2003). [2] Summary statistics of the training set for each language is shown in Table 1. We are unable to evaluate our skip-chain CRF model on CoNLL-02 Spanish due to lack of labels for document boundaries. We use the BIOES tagging scheme, and report the F1 score in its standard definition for NER.

	English	German	Dutch
#document	946	553	287
#sentence	14,987	12,705	15,806
#token	204,567	207,484	202,931
#skip-chain	29,309	31,683	44,309

Table 1: Training sets statistics of CoNLL-03 English and German, and CoNLL-02 Dutch.

For pretrained word embeddings, we use GLoVE (Pennington et al., 2014) for English, and Fasttext (Bojanowski et al., 2017) for German and Dutch. For contextualized word embedding, we use Flair (Akbik et al., 2019) in its recommended settings for each language. For training, we use negative Hamming distance for the metric in the SampleRank loss and Adam optimizer (Kingma and Ba, 2014). For Gibbs sampling, at training time we take 10 cycles of samples for each update. (We resample the full label sequence in each cycle.) At decoding time, we set the initial temperature to 10, the annealing rate to 0.95 and take 120 cycles of samples. We ensemble model predictions over 3 runs with majority vote. Additional hyperparameter settings can be found in Appendix A.

Following the convention for NER tasks (Peters et al., 2017; Akbik et al., 2019), we train the model using both training and development sets when reporting test set results. For analysis, we train the model with training set only and report on development set. We use paired permutation test (Yeh, 2000) for significance testing in result comparisons.

5.2 NER Results

We present the NER results of our neural skip-chain CRF model with Flair embedding in Table 2. The skip-chain CRF has context dependent transition and skip-chain factors, and is trained with Neural SampleRank (NSR), while all other models are trained with standard MLE. On English, we are able to achieve comparable F1 scores as other contextualized embedding models, yet unable to match Akbik et al. (2019). When trained with Flair embedding, our neural skip-chain CRF model does not improve over baseline for English and German. The F1 score difference between baseline and neural skip-chain CRF on German is not statistically significant. Our skip-chain neural CRF model sig-

[2] Following Akbik et al. (2019), we use the 2006 revised ground truth labels for German NER. Clarifications from the author can be found at https://github.com/flairNLP/flair/issues/1102. In our paper, we use † to denote results for which we are not sure about the label version, so they may or may not be comparable to our results. In Appendix C, we discuss more about the label version change, and show results of our models with the original 2003 ground truth labels. In general, models trained and evaluated on the 2006 label set get higher F1 scores than the 2003 label set. The magnitudes of improvements brought by our skip-chain model are comparable on the two label set versions.

Model	Learning	English F1	German F1	Dutch F1
ELMo (Peters et al., 2018)	MLE	92.22	—	—
BERT (Devlin et al., 2018)	MLE	92.80	—	—
Flair + BiLSTM-CRF (Akbik et al., 2019)	MLE	**93.18**	88.27	90.44
Our baseline Flair + BiLSTM-CRF	MLE	92.58	**88.30**	90.63
+ context transition + skip-chain CRF	NSR	92.56	87.97	**91.44***

Table 2: NER F1 score comparison on CoNLL-03 English and German, and CoNLL-02 Dutch dataset, with contextualized embeddings. Bold indicates the highest score, "*" indicates statistical significance compared with baseline.

Model		F1
CRF+linking (Luo et al., 2015)		91.20
BiLSTM-CRF (Lample et al., 2016)		90.94
BiGRU-CRF (Yang et al., 2016)		91.20
BiLSTM-CRF (Ma and Hovy, 2016)		91.21
Our baseline BiLSTM-CRF	MLE	91.01
+ skip-chain	NSR	91.19
+ context transition	MLE	91.18
+ context transition + skip-chain	NSR	**91.68***

Table 3: NER F1 score comparison for English, without contextualized word embeddings.

Model		F1
BiLSTM-CRF (Lample et al., 2016)		78.76†
BiLSTM (Riedl and Padó, 2018)		82.99†
Our baseline BiLSTM-CRF	MLE	83.55
+ context transition + skip-chain	NSR	**84.50***

Table 4: NER F1 score comparison for German, without contextualized word embeddings.

nificantly improves the Flair model's performance on Dutch ($p < 0.01$), achieving new state-of-the-art. According to Table 1, the Dutch dataset has significantly longer documents compared with the other languages, and significantly more skip-chain connections, which could explain why the skip-chain model performs exceptionally well on Dutch.

We further evaluate our neural skip-chain model trained without contextualized word embedding on English and German NER. As shown in Table 3 and Table 4, we are able to significantly improve F1 over baseline on both languages ($p < 0.05$). On English, we also present results when the context dependent transition factors and skip-chain factors are separately added to baseline. We show that the context-dependent transition factors and skip-chain factors can separately improve on NER performance of the base model, and some synergy exists between the two types of factors when used together. Compared with previous approaches that do not use contextualized word embedding or external labeled data, our neural skip-chain CRF model trained with Neural SampleRank achieves the highest F1 on both CoNLL-03 English and German.

5.3 Qualitative Analysis

For all analysis, we investigate the neural skip-chain CRF model without Flair embedding for English, trained without development set. In Figure 2, we show an example of improvements on NER brought by skip-chain factors, from a document in the English development set. We look at two mentions of the English cricketer Peter Such: while the first mention uses his full name, the second mention only uses his last name. From the emission factors, we can see that the local context for the first mention is clear enough for the model to give a high score to label it as a Person type. However, since the last name "Such" is also a common stopword, the model confuses the second mention as a non-entity context. The skip-chain factors are especially helpful in this case, in which long-range contexts can help with disambiguation. From the skip-chain factor, we can see that when looking at both contexts, the model is confident that both mentions are referring to a Person type entity.

5.4 Ablation Study

To compare Neural SampleRank and MLE with exact inference for training, we train the base BiLSTM-CRF model with Neural SampleRank as well. At evaluation time we still use Viterbi decoding for a fair comparison of the training algorithm. As shown in Table 5, the F1 score regressed after switching from exact inference to approximate inference with Neural SampleRank. However, the performance is still comparable.

We conduct ablation study on the neural skip-chain CRF model to see the effectiveness of each component. Results reported in Table 5 are best of

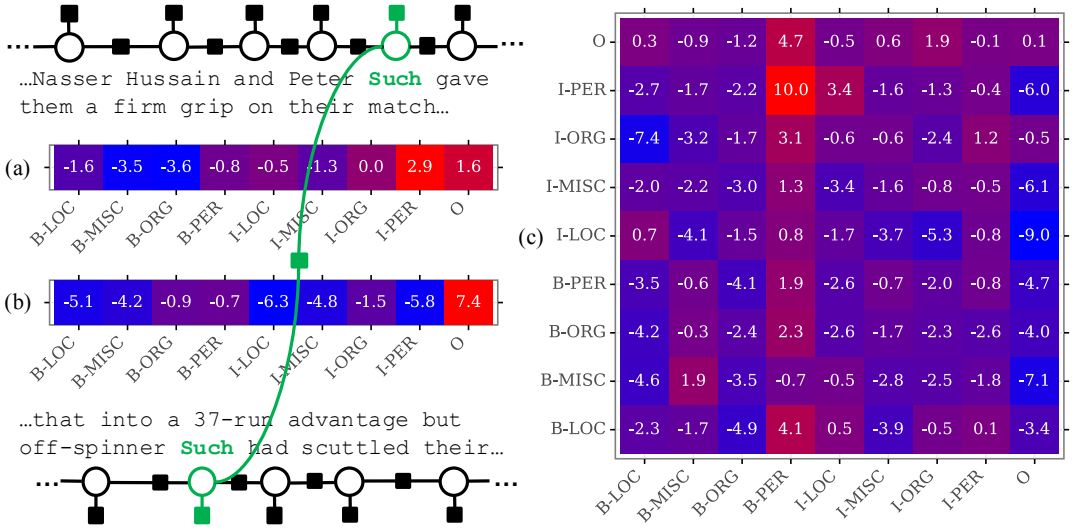


Figure 2: Scores from three factors: (a) unary factor of "Such" in top sentence, (b) unary factor of "Such" in bottom sentence, (c) skip-chain factor connecting the two—column gives the top sentence tag, and row the bottom.

	dev F1
base model (MLE)	94.89
w/ Neural SampleRank	94.50
best model (NSR)	**95.22**
w/o pairwise loss	89.79
w/o gold loss	95.02
w/o block sampling	94.96
w/o context transition	94.89

Table 5: Ablation results on the development set for English. Each row changes one component while keeping all of the others.

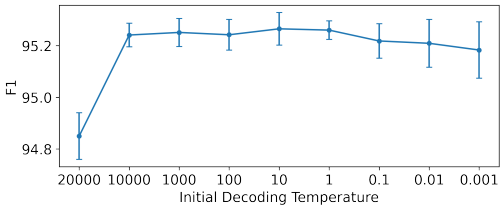


Figure 3: Decoding results with different initial temperature (mean and standard deviation over 10 runs).

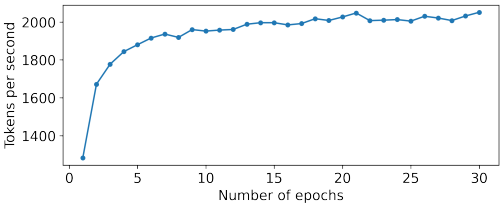


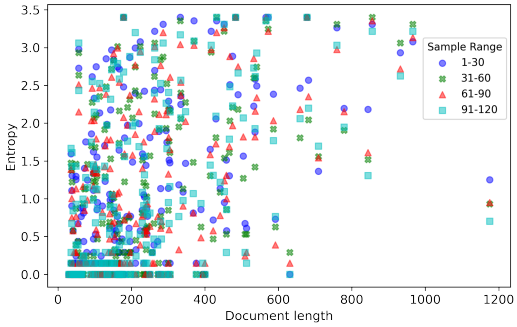
Figure 4: Training speed (tokens per second) for the first 30 epochs of one run.

5 training runs with different random seeds. The mean and standard deviation for the base model setting is 94.67 ± 0.12, while our best skip-chain model setting is 94.96 ± 0.16. This shows that Neural SampleRank does not bring much additional variance in the training process. As for the variance in MCMC decoding, in Figure 3 we show how various initial temperatures affect decoding results. We can see that as long as the initial temperature is high enough for exploration in the beginning, and the temperature anneals sufficiently close to 0 in the end, the decoding achieves optimal performance, with a lower standard deviation compared with training variations.

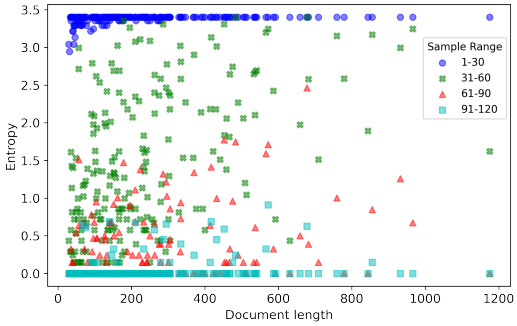
From Table 5, We can see that among the two types of SampleRank loss, the pairwise loss has a much bigger impact on the F1 score than the gold loss. This shows that the training signal introduced by pairwise loss is necessary for efficient Gibbs sampling. The pairwise loss pushes the model locally to a better output structure even when it is far from the gold output. Over time this should push the model towards faster convergence. We also observe that block Gibbs sampling can improve the performance of the skip-chain model, which effectively leverages long-range context dependencies.

5.5 Training speed

As discussed in Section 3.5, the gold loss term can be very dense at the beginning of training, but will become sparse as we train the model and get samples closer to the gold label. The increase in training speed brought by this sparsity is shown in Figure 4. While the first epoch runs at 1283 tokens



(a) MCMC without annealing.



(b) MCMC decoding with annealing.

Figure 5: Entropy of samples against document length, at different stages of MCMC, measured on 30-sample ranges.

per second on average, the first 3 update steps runs at 367 tokens per second. After 30 epochs, the training speed stabilizes at 2,000 tokens per second. As a comparison, the linear-chain model runs at 8,000 tokens per second for training with MLE. See Appendix D for details about the profiling environment.

5.6 Mixing of MCMC

In order to evaluate the mixing of the Markov chain defined by the neural skip-chain CRF, we measure the entropy of samples for each document at different stages of MCMC. Following Keith et al. (2018), we approximate the probability of each sample (i.e. tag sequence) with its frequency when calculating the entropy, then plot this empirical entropy against the length of document. We take 120 samples, by collecting the sample at the end of each cycle (i.e. resampling of the full tag sequence), then split the 120 samples into four 30-sample stages. In Figure 5, we compare the sample entropy of standard MCMC (i.e. without annealing), and MCMC decoding with annealing. From Figure 5a, we observe that the entropy distributions at different stages stay roughly the same, which suggests that the Markov chain is well-mixed, even after a small number of samples. Figure 5b shows how annealing affects sample mixing: Initially, the high temperature leads to samples with high entropy and better exploration. Then, annealing of the temperature drives down the entropy, such that the chain gradually converges to a high probability density region.

6 Conclusion

In this work, we have proposed Neural SampleRank (NSR), an efficient algorithm for approximate inference and training for CRF models with neural network factors. With a novel skip-chain CRF model that models long range context dependencies, NSR can significantly improve NER performance over the linear-chain CRF on multiple datasets. NSR is computationally efficient for arbitrarily complex graphical models, thus applicable to a wide range of structured prediction tasks. Graphical models with task specific inductive bias have been successful for tasks like NER, coreference resolution, relation extraction, and parsing. Our proposed method paves the way for new neural graphical models to be designed for these tasks.

References

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, page 724728.

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

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

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 795–804, Beijing, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629, Copenhagen, Denmark. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Katherine Keith, Su Lin Blodgett, and Brendan O'Connor. 2018. Monte Carlo syntax marginals for exploring and using dependency parses. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 917–928, New Orleans, Louisiana. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Chu-Cheng Lin and Jason Eisner. 2018. Neural particle smoothing for sampling from conditional sequence models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 929–941.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Chaitanya Malaviya, Matthew R Gormley, and Graham Neubig. 2018. Neural factor graph models for cross-lingual morphological tagging. *arXiv preprint arXiv:1805.04570*.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics*, 3:299–313.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1756–1765.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Martin Riedl and Sebastian Padó. 2018. A named entity recognition shootout for German. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 120–125, Melbourne, Australia. Association for Computational Linguistics.

Jiaxin Shi, Jianfei Chen, Jun Zhu, Shengyang Sun, Yucen Luo, Yihong Gu, and Yuhao Zhou. 2017. Zhusuan: A library for bayesian deep learning. *arXiv preprint arXiv:1709.05870*.

Jacob Steinhardt and Percy Liang. 2015. Reified context models. In *International Conference on Machine Learning*, pages 1043–1052.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Charles Sutton and Andrew Mccallum. 2004. Mccallum: Collective segmentation and labeling of distant entities in information extraction. In *In ICML Workshop on Statistical Relational Learning and Its Connections*. Citeseer.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Dustin Tran, Alp Kucukelbir, Adji B Dieng, Maja Rudolph, Dawen Liang, and David M Blei. 2016. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*.

Michael L Wick, Khashayar Rohanimanesh, Kedar Bellare, Aron Culotta, and Andrew McCallum. 2011. Samplerank: Training factor graphs with atomic gradients. In *ICML*, pages 777–784.

Guangxu Xun, Yaliang Li, Wayne Xin Zhao, Jing Gao, and Aidong Zhang. 2017. A correlated topic model using word embeddings. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI17, page 42074213. AAAI Press.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345*.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*.

Yuan Zhang, Tao Lei, Regina Barzilay, Tommi Jaakkola, and Amir Globerson. 2014. Steps to excellence: Simple inference with refined scoring of dependency trees. Association for Computational Linguistics.

Appendices

A Model Hyperparameters

For our base BiLSTM-CRF model, we use a two-layer BiLSTM with hidden state dimension set to 200. For each token, we train a character CNN with 25-dimensional dense character embedding input and 100 filters with size 3, then concatenate with pretrained word embeddings. For English we use the 100 dimensional GLoVE embedding (Pennington et al., 2014). For Dutch and German we use the 300 dimensional Fasttext embedding trained on Wikipedia (Bojanowski et al., 2017). Alternatively, we use Flair embedding (Akbik et al., 2019) in its recommended setup for each language to represent tokens. The emission factors of the CRF are computed by a feed-forward network with a 200 dimensional hidden layer. The transition and skip-chain factors use feed-forward networks with a hidden layer of 500 dimensions, which takes the concatenation, element-wise sum and maximum of the token hidden states as input.

For training, we use negative Hamming distance for the metric in the SampleRank loss. We use Adam optimizer (Kingma and Ba, 2014) with 0.001 initial learning rate, and an annealing rate of 0.5 and patience of 3. We clip the gradients at 1.0, and apply dropout to BiLSTM outputs and feed-forward layers with 0.5 dropout rate. Each mini batch contains 2 documents. For Gibbs sampling, at training time we take 10 cycles of samples for each update. (We resample the full label sequence in each cycle.) At decoding time, we set the initial temperature to 10, the annealing rate to 0.95 and take 120 cycles of samples. We ensemble model predictions over 3 runs with majority vote. Following the convention for NER tasks (Peters et al., 2017; Akbik et al., 2019), we train the model using both training and development sets when reporting test set results. For analysis we train the model with training set only and report on development set. We use paired permutation test (Yeh, 2000) for significance testing in result comparisons.

When training with both train and development sets, the early stopping is determined by the progress of learning rate annealing. The optimal point of learning rate value is determined by our experiments that only use train set for training.

B Evaluation Metrics

For all NER results we report the F1 score in its standard definition for the task. To compute the F1 scores, we directly reuse the perl script released alongside the CoNLL-02/03 shared task [3] .

C CoNLL-03 German Results

For the CoNLL-03 German NER task, there seems to be some discrepancy in the NLP community about the version of ground truth labels being used. Besides the original 2003 ground truth labels, a revised set of labels was released in 2006, with updated annotation guidelines that should lead to higher label quality [4]. The most prominent difference between the two label versions is MISC type entity, where the 2006 version has significantly fewer mentions than the 2003 version, as a result of major changes in the annotation guideline. Statistics of each entity type, in each of the training, development and test sets, is shown in Table 6.

Among the models that we compare our methods against, only Akbik et al. (2019) made clarifications on the label version in an issue in their Github repository for Flair embedding [5]. We are not sure about the label version used in Lample et al. (2016) or Riedl and Padó (2018), thus their results may or may not be comparable to ours. For our models we report results on both label versions in Table 7. We can see that the F1 scores are significantly lower on 2003 labels than on 2006 labels. However, for both versions of data, we get similar trends in the results: while our neural skip-chain CRF model trained with Neural SampleRank is not able to improve over the Flair baseline, it brings statistically significant improvements ($p < 0.05$) for models trained without contextualized word embedding.

We note that our baseline Flair results on 2003 labels match the results reported by Flair users in the Github issue (one user reported 83.22, another reported 83.78). While we can not be certain about the label version used in other works, we speculate that Lample et al. (2016) used the 2003 label version, while Riedl and Padó (2018) used the 2006 version. This speculation is made solely based on

[3] https://www.clips.uantwerpen.be/conll2003/ner/

[4] Both versions of ground truth labels are available here https://www.clips.uantwerpen.be/conll2003/ner/.

[5] https://github.com/flairNLP/flair/issues/1102

Dataset	train		testa (dev)		testb (test)	
Label Version	2003	2006	2003	2006	2003	2006
#PER	2,773	2,801	1,401	1,409	1,195	1,210
#LOC	4,363	4,273	1,181	1,216	1,035	1,051
#ORG	2,427	2,154	1,241	1,090	773	584
#MISC	2,228	780	1,010	216	670	206

Table 6: Number of entities of each type in the 2003 and 2006 version of ground truth labels for CoNLL-03 German.

Model	Contextualized Embeddings	Learning	Label Version	F1
BiLSTM-CRF (Lample et al., 2016)		MLE	Unknown	78.76
BiLSTM (Riedl and Padó, 2018)		MLE	Unknown	82.99
Our baseline BiLSTM-CRF		MLE	2003	78.90
+ context transition + skip-chain CRF		NSR	2003	79.85*
Our baseline Flair + BiLSTM-CRF	✓	MLE	2003	83.20
+ context transition + skip-chain CRF	✓	NSR	2003	83.20
Flair + BiLSTM-CRF (Akbik et al., 2019)	✓	MLE	2006	88.27
Our baseline BiLSTM-CRF		MLE	2006	83.55
+ context transition + skip-chain CRF		NSR	2006	84.50*
Our Flair + linear-chain CRF	✓	MLE	2006	**88.30**
+ context transition + skip-chain CRF	✓	NSR	2006	87.97

Table 7: NER F1 score comparisons on CoNLL-03 German dataset, between 2003 and 2006 ground truth label versions. Bold indicates the highest score, "*" indicates statistical significance compared with baseline.

comparisons with our baseline results, as our model settings are otherwise very similar to theirs.

D Profiling Configurations

Our model is implemented with PyTorch (Paszke et al., 2019), and the Neural SampleRank loss is implemented as a PyTorch C++ extension. The profiling is run with an NVIDIA RTX 2080 Ti GPU, and an Intel Core i9-9900K CPU (8 core 16 threads, 3.6 GHz).