An Exploration of Arbitrary-Order Sequence Labeling via Energy-Based Inference Networks

Lifu Tu[1] [*] **Tianyu Liu**[2] [*] **Kevin Gimpel**[1]
[1] Toyota Technological Institute at Chicago, IL, USA
[2] Institute of Computational Linguistics, Peking University, China
{lifu, kgimpel}@ttic.edu, tianyu0421@pku.edu.cn

Abstract

Many tasks in natural language processing involve predicting structured outputs, e.g., sequence labeling, semantic role labeling, parsing, and machine translation. Researchers are increasingly applying deep representation learning to these problems, but the structured component of these approaches is usually quite simplistic. In this work, we propose several high-order energy terms to capture complex dependencies among labels in sequence labeling, including several that consider the entire label sequence. We use neural parameterizations for these energy terms, drawing from convolutional, recurrent, and self-attention networks. We use the framework of learning energy-based inference networks (Tu and Gimpel, 2018) for dealing with the difficulties of training and inference with such models. We empirically demonstrate that this approach achieves substantial improvement using a variety of high-order energy terms on four sequence labeling tasks, while having the same decoding speed as simple, local classifiers. We also find high-order energies to help in noisy data conditions.[1]

1 Introduction

Conditional random fields (CRFs; Lafferty et al., 2001) have been shown to perform well in various sequence labeling tasks. Recent work uses rich neural network architectures to define the "unary" potentials, i.e., terms that only consider a single position's label at a time (Collobert et al., 2011; Lample et al., 2016; Ma and Hovy, 2016; Strubell et al., 2018). However, "binary" potentials, which consider pairs of adjacent labels, are usually quite simple and may consist solely of a parameter or parameter vector for each unique label transition. Models with unary and binary potentials are generally referred to as "first order" models.

A major challenge with CRFs is the complexity of training and inference, which are quadratic in the number of output labels for first order models and grow exponentially when higher order dependencies are considered. This explains why the most common type of CRF used in practice is a first order model, also referred to as a "linear chain" CRF.

One promising alternative to CRFs is structured prediction energy networks (SPENs; Belanger and McCallum, 2016), which use deep neural networks to parameterize arbitrary potential functions for structured prediction. While SPENs also pose challenges for learning and inference, Tu and Gimpel (2018) proposed a way to train SPENs jointly with "inference networks", neural networks trained to approximate structured $\arg\max$ inference.

In this paper, we leverage the frameworks of SPENs and inference networks to explore high-order energy functions for sequence labeling. Naively instantiating high-order energy terms can lead to a very large number of parameters to learn, so we instead develop concise neural parameterizations for high-order terms. In particular, we draw from vectorized Kronecker products, convolutional networks, recurrent networks, and self-attention. We also consider "skip-chain" connections (Sutton and McCallum, 2004) with various skip distances and ways of reducing their total parameter count for increased learnability.

Our experimental results on four sequence labeling tasks show that a range of high-order energy functions can yield performance improvements. While the optimal energy function varies by task, we find strong performance from skip-chain terms with short skip distances, convolutional networks with filters that consider label trigrams, and recurrent networks and self-attention networks that consider large subsequences of labels.

[*]Equal contribution.
[1]Code is available at https://github.com/tyliupku/Arbitrary-Order-Infnet

We also demonstrate that modeling high-order dependencies can lead to significant performance improvements in the setting of noisy training and test sets. Visualizations of the high-order energies show various methods capture intuitive structured dependencies among output labels.

Throughout, we use inference networks that share the same architecture as unstructured classifiers for sequence labeling, so test time inference speeds are unchanged between local models and our method. Enlarging the inference network architecture by adding one layer leads consistently to better results, rivaling or improving over a BiLSTM-CRF baseline, suggesting that training efficient inference networks with high-order energy terms can make up for errors arising from approximate inference. While we focus on sequence labeling in this paper, our results show the potential of developing high-order structured models for other NLP tasks in the future.

2 Background

2.1 Structured Energy-Based Learning

We denote the input space by $\mathcal{X}$. For an input $\boldsymbol{x} \in \mathcal{X}$, we denote the structured output space by $\mathcal{Y}(\boldsymbol{x})$. The entire space of structured outputs is denoted $\mathcal{Y} = \cup_{\boldsymbol{x}\in\mathcal{X}}\mathcal{Y}(\boldsymbol{x})$. We define an **energy function** (LeCun et al., 2006; Belanger and McCallum, 2016) E_Θ parameterized by Θ that computes a scalar energy for an input/output pair: $E_\Theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$. At test time, for a given input $\boldsymbol{x}$, prediction is done by choosing the output with lowest energy:

$$\hat{\boldsymbol{y}} = \arg\min_{\boldsymbol{y}\in\mathcal{Y}(\boldsymbol{x})} E_\Theta(\boldsymbol{x}, \boldsymbol{y}) \qquad (1)$$

2.2 Inference Networks

Inference. Solving equation (1) requires combinatorial algorithms because $\mathcal{Y}$ is a structured, discrete space. This becomes intractable when E_Θ does not decompose into a sum over small "parts" of $\boldsymbol{y}$. Belanger and McCallum (2016) relax this problem by allowing the discrete vector $\boldsymbol{y}$ to be continuous. Let $\mathcal{Y}_R$ denote the **relaxed output space**. They solve the relaxed problem by using gradient descent to iteratively minimize the energy with respect to $\boldsymbol{y}$.

Tu and Gimpel (2018) propose an alternative that replaces gradient descent with a neural network trained to do inference, i.e., to mimic the function performed in equation (1). This "inference network" $\mathbf{A}_\Psi : \mathcal{X} \rightarrow \mathcal{Y}_R$ is parameterized by Ψ and trained with the goal that

$$\mathbf{A}_\Psi(\boldsymbol{x}) \approx \underset{\boldsymbol{y}\in\mathcal{Y}_R(\boldsymbol{x})}{\arg\min}\, E_\Theta(\boldsymbol{x}, \boldsymbol{y}) \qquad (2)$$

Tu and Gimpel (2019) show that inference networks achieve a better speed/accuracy/search error trade-off than gradient descent given pretrained energy functions.

Joint training of energy functions and inference networks. Belanger and McCallum (2016) proposed a structured hinge loss for learning the energy function parameters Θ, using gradient descent for the "cost-augmented" inference step required during learning. Tu and Gimpel (2018) replaced the cost-augmented inference step in the structured hinge loss with training of a "cost-augmented inference network" $\mathbf{F}_\Phi(\boldsymbol{x})$ trained with the following goal:

$$\mathbf{F}_\Phi(\boldsymbol{x}) \approx \underset{\boldsymbol{y}\in\mathcal{Y}_R(\boldsymbol{x})}{\arg\min}\, (E_\Theta(\boldsymbol{x}, \boldsymbol{y}) - \triangle(\boldsymbol{y}, \boldsymbol{y}^*))$$

where $\triangle$ is a structured cost function that computes the distance between its two arguments. The new optimization objective becomes:

$$\min_\Theta \max_\Phi \sum_{\langle \boldsymbol{x},\boldsymbol{y}\rangle\in\mathcal{D}} [\triangle(\mathbf{F}_\Phi(\boldsymbol{x}), \boldsymbol{y}) - E_\Theta(\boldsymbol{x}, \mathbf{F}_\Phi(\boldsymbol{x})) + E_\Theta(\boldsymbol{x}, \boldsymbol{y})]_+$$

where $\mathcal{D}$ is the set of training pairs and $[h]_+ = \max(0, h)$. Tu and Gimpel (2018) alternatively optimized Θ and Φ, which is similar to training in generative adversarial networks (Goodfellow et al., 2014).

2.3 An Objective for Joint Learning of Inference Networks

One challenge with the optimization problem above is that it still requires training an inference network $\mathbf{A}_\Psi$ for test-time prediction. Tu et al. (2020a) proposed a "compound" objective that avoids this by training two inference networks jointly (with shared parameters), $\mathbf{F}_\Phi$ for cost-augmented inference and $\mathbf{A}_\Psi$ for test-time inference:

$$\min_\Theta \max_{\Phi,\Psi} \sum_{\langle \boldsymbol{x},\boldsymbol{y}\rangle\in\mathcal{D}} \underbrace{[\triangle(\mathbf{F}_\Phi(\boldsymbol{x}), \boldsymbol{y}) - E_\Theta(\boldsymbol{x}, \mathbf{F}_\Phi(\boldsymbol{x})) + E_\Theta(\boldsymbol{x}, \boldsymbol{y})]_+}_{\text{margin-rescaled hinge loss}} + \lambda \underbrace{[-E_\Theta(\boldsymbol{x}, \mathbf{A}_\Psi(\boldsymbol{x})) + E_\Theta(\boldsymbol{x}, \boldsymbol{y})]_+}_{\text{perceptron loss}}$$

As indicated, this loss can be viewed as the sum of the margin-rescaled and perceptron losses. Θ, Φ, and Ψ are alternatively optimized. The objective for the energy function parameters Θ is:

$$\hat{\Theta} \leftarrow \arg\min_{\Theta}$$
$$\big[\triangle(\mathbf{F}_\Phi(\boldsymbol{x}), \boldsymbol{y}) - E_\Theta(\boldsymbol{x}, \mathbf{F}_\Phi(\boldsymbol{x})) + E_\Theta(\boldsymbol{x}, \boldsymbol{y})\big]_+$$
$$+ \lambda\big[-E_\Theta(\boldsymbol{x}, \mathbf{A}_\Psi(\boldsymbol{x})) + E_\Theta(\boldsymbol{x}, \boldsymbol{y})\big]_+$$

The objective for the other parameters is:

$$\hat{\Psi}, \hat{\Phi} \leftarrow \arg\max_{\Psi,\Phi} \triangle(\mathbf{F}_\Phi(\boldsymbol{x}), \boldsymbol{y}) - E_\Theta(\boldsymbol{x}, \mathbf{F}_\Phi(\boldsymbol{x}))$$
$$- \lambda E_\Theta(\boldsymbol{x}, \mathbf{A}_\Psi(\boldsymbol{x})) - \tau \ell_{\text{token}}(\boldsymbol{y}, \mathbf{A}_\Psi(\boldsymbol{x}))$$

where ℓ_{token} is a supervised token-level loss which is added to aid in training inference networks. In this paper, we use the standard cross entropy summed over all positions. Like Tu et al. (2020a), we drop the zero truncation ($\max(0, .)$) when updating the inference network parameters to improve stability during training, which also lets us remove the terms that do not have inference networks. We use two independent networks but with the same architecture for the two inference networks.

3 Energy Functions

Our experiments in this paper consider sequence labeling tasks, so the input $\boldsymbol{x}$ is a length-T sequence of tokens where x_t denotes the token at position t. The output $\boldsymbol{y}$ is a sequence of labels also of length T. We use $\boldsymbol{y}_t$ to denote the output label at position t, where $\boldsymbol{y}_t$ is a vector of length L (the number of labels in the label set) and where $y_{t,j}$ is the jth entry of the vector $\boldsymbol{y}_t$. In the original output space $\mathcal{Y}(\boldsymbol{x})$, $y_{t,j}$ is 1 for a single j and 0 for all others. In the relaxed output space $\mathcal{Y}_R(\boldsymbol{x})$, $y_{t,j}$ can be interpreted as the probability of the tth position being labeled with label j. We use the following energy: $E_\Theta(\boldsymbol{x}, \boldsymbol{y}) =$

$$-\left(\sum_{t=1}^{T}\sum_{j=1}^{L} y_{t,j}\left(U_j^\top b(\boldsymbol{x}, t)\right) + E_W(\boldsymbol{y})\right) \quad (3)$$

where $U_j \in \mathbb{R}^d$ is a parameter vector for label j and $E_W(\boldsymbol{y})$ is a structured energy term parameterized by parameters W. In a linear chain CRF, W is a transition matrix for scoring two adjacent labels. Different instantiations of E_W will be detailed in the sections below. Also, $b(\boldsymbol{x}, t) \in \mathbb{R}^d$ denotes the "input feature vector" for position t. We define it to be the d-dimensional BiLSTM (Hochreiter and Schmidhuber, 1997) hidden vector at t. The full set of energy parameters Θ includes the U_j vectors, W, and the parameters of the BiLSTM.

The above energy functions are trained with the objective in Section 2.3. Table 1 shows the training and test-time inference requirements of our method compared to previous methods. For different formulations of the energy function, the inference network architecture is the same (e.g., BiLSTM). So the inference complexity is the same as the standard neural approaches that do not use structured prediction, which is linear in the label set size. However, even for the first order model (linear-chain CRF), the time complexity is quadratic in the label set size. The time complexity of higher-order CRFs grows exponentially with the order.

3.1 Linear Chain Energies

Our first choice for a structured energy term is relaxed linear chain energy defined for sequence labeling by Tu and Gimpel (2018):

$$E_W(\boldsymbol{y}) = \sum_{t=1}^{T} \boldsymbol{y}_{t-1}^\top W \boldsymbol{y}_t$$

Where $W_i \in \mathbb{R}^{L\times L}$ is the transition matrix, which is used to score the pair of adjacent labels. If this linear chain energy is the only structured energy term in use, exact inference can be performed efficiently using the Viterbi algorithm.

3.2 Skip-Chain Energies

We also consider an energy inspired by "skip-chain" conditional random fields (Sutton and McCallum, 2004). In addition to consecutive labels, this energy also considers pairs of labels appearing in a given window size $M + 1$:

$$E_W(\boldsymbol{y}) = \sum_{t=1}^{T}\sum_{i=1}^{M} \boldsymbol{y}_{t-i}^\top W_i \boldsymbol{y}_t$$

where each $W_i \in \mathbb{R}^{L\times L}$ and the max window size M is a hyperparameter. While linear chain energies allow efficient exact inference, using skip-chain energies causes exact inference to require time exponential in the size of M.

3.3 High-Order Energies

We also consider Mth-order energy terms. We use the function F to score the $M + 1$ consecutive

	Training		Inference	
	Time	Number of Parameters	Time	Number of Parameters
BiLSTM	$\mathcal{O}(T*L)$	$\mathcal{O}(\|\Psi\|)$	$\mathcal{O}(T*L)$	$\mathcal{O}(\|\Psi\|)$
CRF	$\mathcal{O}(T*L^2)$	$\mathcal{O}(\|\Theta\|)$	$\mathcal{O}(T*L^2)$	$\mathcal{O}(\|\Theta\|)$
Energy-Based Inference Networks	$\mathcal{O}(T*L)$	$\mathcal{O}(\|\Psi\|+\|\Phi\|+\|\Theta\|)$	$\mathcal{O}(T*L)$	$\mathcal{O}(\|\Psi\|)$

Table 1: Time complexity and number of parameters of different methods during training and inference, where T is the sequence length, L is the label set size, Θ are the parameters of energy function, and Φ, Ψ are the parameters of two energy-based inference networks. For arbitrary-order energy functions or different parameterizations, the size of Θ can be different.

labels $\boldsymbol{y}_{t-M}, \ldots, \boldsymbol{y}_t$, then sum over positions:

$$E_W(\boldsymbol{y}) = \sum_{t=M}^{T} F(\boldsymbol{y}_{t-M}, \ldots, \boldsymbol{y}_t) \qquad (4)$$

We consider several different ways to define the function F, detailed below.

Vectorized Kronecker Product (VKP): A naive way to parameterize a high-order energy term would involve using a parameter tensor $W \in \mathbb{R}^{L^{M+1}}$ with an entry for each possible label sequence of length $M+1$. To avoid this exponentially-large number of parameters, we define a more efficient parameterization as follows. We first define a label embedding lookup table $\in \mathbb{R}^{L\times n_l}$ and denote the embedding for label j by e_j. We consider $M=2$ as an example. Then, for a tensor $W \in \mathbb{R}^{L\times L\times L}$, its value $W_{i,j,k}$ at indices (i,j,k) is calculated as

$$\mathbf{v}^\top \text{LayerNorm}([e_i;e_j;e_k] + \text{MLP}([e_i;e_j;e_k]))$$

where $\mathbf{v} \in \mathbb{R}^{(M+1)n_l}$ is a parameter vector and ; denotes vector concatenation. MLP expects and returns vectors of dimension $(M+1)\times n_l$ and is parameterized as a multilayer perceptron. Then, the energy is computed:

$$F(\boldsymbol{y}_{t-M}, \ldots, \boldsymbol{y}_t) = \text{VKP}(\boldsymbol{y}_{t-M}, \ldots, \boldsymbol{y}_{t-1}) W \boldsymbol{y}_t$$

where W is reshaped as $\in \mathbb{R}^{L^M\times L}$. The operator VKP is somewhat similar to the Kronecker product of the k vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$[2]. However it will return a vector, not a tensor:

$$\text{VKP}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k) = \begin{cases} \boldsymbol{v}_1 & k=1 \\ \mathbf{vec}(\boldsymbol{v}_1\boldsymbol{v}_2^\top) & k=2 \\ \mathbf{vec}(\text{VKP}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k-1})\boldsymbol{v}_k^\top) & k>2 \end{cases}$$

[2]There are some work (Lei et al., 2014; Srikumar and Manning, 2014; Yu et al., 2016) that use Kronecker product for higher order feature combinations with low-rank tensors. Here we use this form to express the computation when scoring the consecutive labels.

Where **vec** is the operation that vectorizes a tensor into a (column) vector.

CNN: Convolutional neural networks (CNN) are frequently used in NLP to extract features based on words or characters (Collobert et al., 2011; Kim, 2014). We apply CNN filters over the sequence of $M+1$ consecutive labels. The F function is computed as follows:

$$F(\boldsymbol{y}_{t-M}, \ldots, \boldsymbol{y}_t) = \sum_n f_n(\boldsymbol{y}_{t-M}, \ldots, \boldsymbol{y}_t)$$

$$f_n(\boldsymbol{y}_{t-M}, \ldots, \boldsymbol{y}_t) = g(W_n[\boldsymbol{y}_{t-M}; \ldots; \boldsymbol{y}_t] + \mathbf{b}_n)$$

where g is a ReLU nonlinearity and the vector $W_n \in \mathbb{R}^{L(M+1)}$ and scalar $b_n \in \mathbb{R}$ are the parameters for filter n. The filter size of all filters is the same as the window size, namely, $M+1$. The F function sums over all CNN filters. When viewing this high-order energy as a CNN, we can think of the summation in Eq. 4 as corresponding to sum pooling over time of the feature map outputs.

Tag Language Model (TLM): Tu and Gimpel (2018) defined an energy term based on a pretrained "tag language model", which computes the probability of an entire sequence of labels. We also use a TLM, scoring a sequence of $M+1$ consecutive labels in a way similar to Tu and Gimpel (2018); however, the parameters of the TLM are trained in our setting:

$$F(\boldsymbol{y}_{t-M}, \ldots, \boldsymbol{y}_t) = -\sum_{t'=t-M+1}^{t} \boldsymbol{y}_{t'}^\top \log(\text{TLM}(\langle \boldsymbol{y}_{t-M}, \ldots, \boldsymbol{y}_{t'-1}\rangle))$$

where $\text{TLM}(\langle \boldsymbol{y}_{t-M}, \ldots, y_{t'-1}\rangle)$ returns the softmax distribution over tags at position t' (under the tag language model) given the preceding tag vectors. When each $y_{t'}$ is a one-hot vector, this energy reduces to the negative log-likelihood of the tag sequence specified by $\boldsymbol{y}_{t-M}, \ldots, \boldsymbol{y}_t$.

Self-Attention (S-Att): We adopt the multi-head self-attention formulation from Vaswani et al. (2017). Given a matrix of the $M+1$ consecutive labels $Q = K = V = [\boldsymbol{y}_{t-M}; \ldots; \boldsymbol{y}_t] \in \mathbb{R}^{(M+1)\times L}$:

$$H = \mathrm{attention}(Q, K, V)$$
$$F(\boldsymbol{y}_{t-M}, \ldots, \boldsymbol{y}_t) = \sum H$$

where attention is the general attention mechanism: the weighted sum of the value vectors V using query vectors Q and key vectors K (Vaswani et al., 2017). The energy on the $M+1$ consecutive labels is defined as the sum of entries in the feature map $H \in \mathbb{R}^{L\times(M+1)}$ after the self-attention transformation.

3.4 Fully-Connected Energies

We can simulate a "fully-connected" energy function by setting a very large value for M in the skip-chain energy (Section 3.2). For efficiency and learnability, we use a low-rank parameterization for the many translation matrices W_i that will result from increasing M. We first define a matrix $S \in \mathbb{R}^{L\times d}$ that all W_i will use. Each i has a learned parameter matrix $D_i \in \mathbb{R}^{L\times d}$ and together S and D_i are used to compute W_i:

$$W_i = SD_i^\top$$

where d is a tunable hyperparameter that affects the number of learnable parameters.

4 Related Work

Linear chain CRFs (Lafferty et al., 2001), which consider dependencies between at most two adjacent labels or segments, are commonly used in practice (Sarawagi and Cohen, 2005; Lample et al., 2016; Ma and Hovy, 2016).

There have been several efforts in developing efficient algorithms for handling higher-order CRFs. Qian et al. (2009) developed an efficient decoding algorithm under the assumption that all high-order features have non-negative weights. Some work has shown that high-order CRFs can be handled relatively efficiently if particular patterns of sparsity are assumed (Ye et al., 2009; Cuong et al., 2014). Mueller et al. (2013) proposed an approximate CRF using coarse-to-fine decoding and early updating. Loopy belief propagation (Murphy et al., 1999) has been used for approximate inference in high-order CRFs, such as skip-chain CRFs (Sutton and McCallum, 2004), which form the inspiration for one category of energy function in this paper.

CRFs are typically trained by maximizing conditional log-likelihood. Even assuming that the graph structure underlying the CRF admits tractable inference, it is still time-consuming to compute the partition function. Margin-based methods have been proposed (Taskar et al., 2004; Tsochantaridis et al., 2004) to avoid the summation over all possible outputs. Similar losses are used when training SPENs (Belanger and McCallum, 2016; Belanger et al., 2017), including in this paper. The energy-based inference network learning framework has been used for multi-label classification (Tu and Gimpel, 2018), non-autoregressive machine translation (Tu et al., 2020b), and previously for sequence labeling (Tu and Gimpel, 2019).

Moving beyond CRFs and sequence labeling, there has been a great deal of work in the NLP community in designing non-local features, often combined with the development of approximate algorithms to incorporate them during inference. These include n-best reranking (Och et al., 2004), beam search (Lowerre, 1976), loopy belief propagation (Sutton and McCallum, 2004; Smith and Eisner, 2008), Gibbs sampling (Finkel et al., 2005), stacked learning (Cohen and de Carvalho, 2005; Krishnan and Manning, 2006), sequential Monte Carlo algorithms (Yang and Eisenstein, 2013), dynamic programming approximations like cube pruning (Chiang, 2007; Huang and Chiang, 2007), dual decomposition (Rush et al., 2010; Martins et al., 2011), and methods based on black-box optimization like integer linear programming (Roth and Yih, 2004). These methods are often developed or applied with particular types of non-local energy terms in mind. By contrast, here we find that the framework of SPEN learning with inference networks can support a wide range of high-order energies for sequence labeling.

5 Experimental Setup

We perform experiments on four tasks: Twitter part-of-speech tagging (POS), named entity recognition (NER), CCG supertagging (CCG), and semantic role labeling (SRL).

5.1 Datasets

POS. We use the annotated data from Gimpel et al. (2011) and Owoputi et al. (2013) which contains 25 POS tags. We use the 100-dimensional skip-gram embeddings from Tu et al. (2017) which were trained on a dataset of 56 million English tweets using `word2vec` (Mikolov et al., 2013). The evaluation metric is tagging accuracy.

NER. We use the CoNLL 2003 English data (Tjong Kim Sang and De Meulder, 2003). We use the BIOES tagging scheme, so there are 17 labels. We use 100-dimensional pretrained GloVe (Pennington et al., 2014) embeddings. The task is evaluated with micro-averaged F1 score.

CCG. We use the standard splits from CCGbank (Hockenmaier and Steedman, 2002). We only keep sentences with length less than 50 in the original training data during training. We use only the 400 most frequent labels. The training data contains 1,284 unique labels, but because the label distribution has a long tail, we use only the 400 most frequent labels, replacing the others by a special tag $*$. The percentages of $*$ in train/development/test are 0.25/0.23/0.23$\%$. When the gold standard tag is $*$, the prediction is always evaluated as incorrect. We use the same GloVe embeddings as in NER. The task is evaluated with per-token accuracy.

SRL. We use the standard split from CoNLL 2005 (Carreras and Màrquez, 2005). The gold predicates are provided as part of the input. We use the official evaluation script from the CoNLL 2005 shared task for evaluation. We again use the same GloVe embeddings as in NER. To form the inputs to our models, an embedding of a binary feature indicating whether the word is the given predicate is concatenated to the word embedding.[3]

5.2 Training

Local Classifiers. We consider local baselines that use a BiLSTM trained with the local loss ℓ_{token}. For POS, NER and CCG, we use a 1-layer BiLSTM with hidden size 100, and the word embeddings are fixed during training. For SRL, we use a 4-layer BiLSTM with hidden size 300 and the word embeddings are fine-tuned.

BiLSTM-CRF. We also train BiLSTM-CRF models with the standard conditional log-likelihood objective. A 1-layer BiLSTM with hidden size 100 is used for extracting input features. The CRF part uses a linear chain energy with a single tag transition parameter matrix. We do early stopping based on development sets. The usual dynamic programming algorithms are used for training and inference, e.g., the Viterbi algorithm is used for inference. The same pretrained word embeddings as for the local classifiers are used.

Inference Networks. When defining architectures for the inference networks, we use the same architectures as the local classifiers. However, the objective of the inference networks is different, which is shown in Section 2.3. $\lambda = 1$ and $\tau = 1$ are used for training. We do early stopping based on the development set.

Energy Terms. The unary terms are parameterized using a one-layer BiLSTM with hidden size 100. For the structured energy terms, the VKP operation uses $n_l = 20$, the number of CNN filters is 50, and the tag language model is a 1-layer LSTM with hidden size 100. For the fully-connected energy, $d = 20$ for the approximation of the transition matrix and $M = 20$ for the approximation of the fully-connected energies.

Hyperparameters. For the inference network training, the batch size is 100. We update the energy function parameters using the Adam optimizer (Kingma and Ba, 2014) with learning rate 0.001. For POS, NER, and CCG, we train the inference networks parameter with stochastic gradient descent with momentum as the optimizer. The learning rate is 0.005 and the momentum is 0.9. For SRL, we train the inference networks using Adam with learning rate 0.001.

6 Results

Parameterizations for High-Order Energies. We first compare several choices for energy functions within our inference network learning framework. In Section 3.3, we considered several ways to define the high-order energy function F. We compare performance of the parameterizations on three tasks: POS, NER, and CCG. The results are shown in Table 2.

For VKP high-order energies, there are small differences between 2nd and 3rd order models, however, 4th order models are consistently worse. The CNN high-order energy is best when M=2 for the three tasks. Increasing M does not consistently help. The tag language model (TLM) works best when scoring the entire label sequence. In

[3]Our SRL baseline is most similar to Zhou and Xu (2015), though there are some differences. We use GloVe embeddings while they train word embeddings on Wikipedia. We both use the same predicate context features.

		POS	NER	CCG
Linear Chain		89.5	90.6	92.8
VKP	$M = 2$	**89.9**	91.1	**93.1**
	$M = 3$	89.8	**91.2**	92.9
	$M = 4$	89.5	90.8	92.8
CNN	$M = 1$	89.7	91.1	93.0
	$M = 2$	**90.0**	**91.3**	**93.0**
	$M = 3$	89.9	91.2	92.9
	$M = 4$	89.7	91.0	**93.0**
TLM	$M = 2$	89.7	90.8	92.4
	$M = 3$	89.8	91.0	92.7
	$M = 4$	89.8	91.3	92.7
	all	**90.0**	**91.4**	**92.9**
S-Att	$M = 2$	89.7	90.7	92.6
	$M = 4$	89.8	90.8	92.8
	$M = 6$	**89.9**	90.9	92.8
	$M = 8$	**89.9**	**91.0**	93.0
	all	89.7	90.8	**93.1**

Table 2: Development results for different parameterizations of high-order energies when increasing the window size M of consecutive labels, where "all" denotes the whole relaxed label sequence. The inference network architecture is a one-layer BiLSTM. We ran t-tests for the mean performance (over five runs) of our proposed energies (the settings in bold) and the linear-chain energy. All differences are significant at $p < 0.001$ for NER and $p < 0.005$ for other tasks.

the following experiment with TLM energies, we always use it with this "all" setting. Self-attention (S-Att) also shows better performance with larger M. However, the results for NER are not as high overall as for other energy terms.

Overall, there is no clear winner among the four types of parameterizations, indicating that a variety of high-order energy terms can work well on these tasks, once appropriate window sizes are chosen. We do note differences among tasks: NER benefits more from larger window sizes than POS.

Comparing Structured Energy Terms. Above we compared parameterizations of the high-order energy terms. In Table 3, we compare instantiations of the structured energy term $E_W(\boldsymbol{y})$: linear-chain energies, skip-chain energies, high-order energies, and fully-connected energies.[4] We also compare to local classifiers (BiLSTM). The models with structured energies typically improve over the local classifiers, even with just the linear chain energy.

The richer energy terms tend to perform better than linear chain, at least for most tasks and energies. The skip-chain energies benefit from relatively large M values, i.e., 3 or 4 depending on the task. These tend to be larger than the optimal VKP M values. We note that S-Att high-order energies work well on SRL. This points to the benefits of self-attention on SRL, which has been found in recent work (Tan et al., 2018; Strubell et al., 2018).

Both the skip-chain and high-order energy models achieve substantial improvements over the linear chain CRF, notably a gain of 0.8 F1 for NER. The fully-connected energy is not as strong as the others, possibly due to the energies from label pairs spanning a long range. These long-range energies do not appear helpful for these tasks.

		POS	NER	CCG	SRL	
					WSJ	Brown
BiLSTM		88.7	85.3	92.8	81.8	71.8
Linear Chain		89.7	85.9	93.0	81.7	72.0
Skip-Chain		90.0	**86.7**	**93.3**	82.1	**72.4**
High-Order	VKP	**90.1**	**86.7**	**93.3**	81.8	72.0
	CNN	**90.1**	86.5	93.2	81.9	72.2
	TLM	90.0	86.6	93.0	81.8	72.1
	S-Att	**90.1**	86.5	**93.3**	**82.2**	72.2
Fully-Connected		89.8	86.3	92.9	81.4	71.4

Table 3: Test results on all tasks for local classifiers (BiLSTM) and different structured energy functions. POS/CCG use accuracy while NER/SRL use F1. The architecture of inference networks is one-layer BiLSTM. More results are shown in the appendix.

		POS	NER	CCG
2-layer BiLSTM		88.8	86.0	93.4
BiLSTM-CRF		89.2	87.3	93.1
Linear Chain		90.0	86.6	93.7
Skip-Chain		**90.2**	**87.5**	**93.8**
High-Order	VKP	**90.2**	87.2	**93.8**
	CNN	**90.2**	87.3	93.6
	TLM	90.1	87.1	93.6
	S-Att	90.0	87.3	93.7
Fully-Connected		90.0	87.2	93.3

Table 4: Test results when inference networks have 2 layers (so the local classifier baseline also has 2 layers).

Comparison using Deeper Inference Networks. Table 4 compares methods when using 2-layer BiLSTMs as inference networks.[5] The deeper inference networks reach higher performance across all tasks compared to 1-layer inference networks.

We observe that inference networks trained with skip-chain energies and high-order energies achieve better results than BiLSTM-CRF on the three datasets (the Viterbi algorithm is used for

[4] M values are tuned based on dev sets. Tuned M values for POS/NER/CCG/SRL: Skip-Chain: 3/4/3/3; VKP: 2/3/2/2; CNN: 2/2/2/2; TLM: whole sequence; S-Att: 8/8/8/8.

[5] M values are retuned based on dev sets when using 2-layer inference networks. Tuned M values for POS/NER/CCG: Skip-Chain: 3/4/3; VKP: 2/3/2; CNN: 2/2/2; TLM: whole sequence; S-Att: 8/8/8.

	α=0.1	α=0.2	α=0.3
BiLSTM	75.0	67.2	58.8
Linear Chain	75.2	67.4	59.1
Skip-Chain (M=4)	75.5	67.9	59.5
VKP (M=3)	75.3	67.7	59.3
CNN (M=0)	75.7	67.9	59.4
CNN (M=2)	76.3	68.6	60.2
CNN (M=4)	**76.7**	**69.8**	**60.4**
TLM	76.0	67.8	59.9
S-Att (M=8)	75.6	67.6	59.7

Table 5: UnkTest setting for NER: words in the test set are replaced by the unknown word symbol with probability α. For CNN energies (the settings in bold) and linear-chain energy, they differ significantly with $p < 0.001$.

	α=0.1	α=0.2	α=0.3
BiLSTM	80.1	76.0	70.6
Linear Chain	80.4	76.3	70.9
Skip-Chain (M=4)	81.2	76.7	71.2
VKP (M=3)	81.4	76.8	71.4
CNN (M=0)	81.1	76.7	71.5
CNN (M=2)	81.8	77.0	**71.8**
CNN (M=4)	**82.0**	**77.1**	71.7
TLM	80.9	76.3	71.1
S-Att (M=8)	81.4	76.9	71.4

Table 6: UnkTrain setting for NER: training on noisy text, evaluating on noisy test sets. Words are replaced by the unknown word symbol with probability α. For CNN energies (the settings in bold) and linear-chain energy, they differ significantly with $p < 0.001$.

exact inference for BiLSTM-CRF). This indicates that adding richer energy terms can make up for approximate inference during training and inference. Moreover, a 2-layer BiLSTM is much cheaper computationally than Viterbi, especially for tasks with large label sets.

6.1 Results on Noisy Datasets

We now consider the impact of our structured energy terms in noisy data settings. Our motivation for these experiments stems from the assumption that structured energies will be more helpful when there is a weaker relationship between the observations and the labels. One way to achieve this is by introducing noise into the observations.

So, we create new datasets: for any given sentence, we randomly replace a token x with an unknown word symbol "UNK" with probability α. From previous results, we see that NER shows more benefit from structured energies, so we focus on NER and consider two settings: **UnkTest**: train on clean text, evaluate on noisy text; and **UnkTrain**: train on noisy text, evaluate on noisy text.

Baselines:	
BERT (local loss)	92.13
BERT-CRF	92.34
Energy-based inference networks:	
Linear Chain	92.14
Skip-Chain (M=3)	92.46

Table 7: Test results for NER when using BERT. When using energy-based inference networks (our framework), BERT is used in both the energy function and as the inference network architecture.

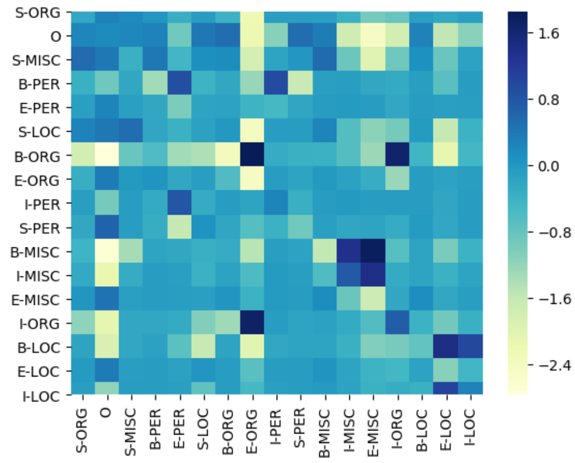
Table 5 shows results for UnkTest. CNN energies are best among all structured energy terms, including the different parameterizations. Increasing M improves F1, showing that high-order information helps the model recover from the high degree of noise. Table 6 shows results for UnkTrain. The CNN high-order energies again yield large gains: roughly 2 points compared to the local classifier and 1.8 compared to the linear chain energy.

7 Incorporating BERT

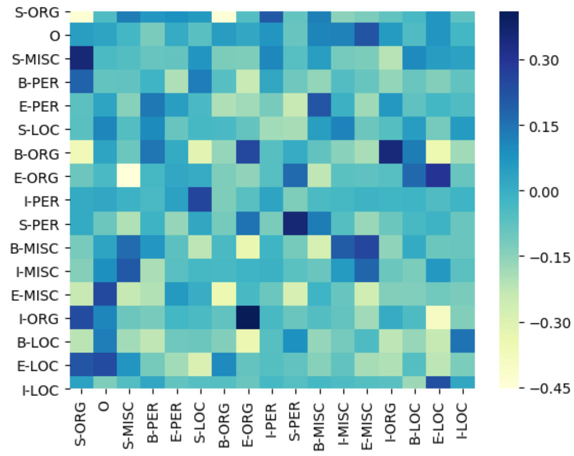
Researchers have recently been applying large-scale pretrained transformers like BERT (Devlin et al., 2019) to many tasks, including sequence labeling. To explore the impact of high-order energies on BERT-like models, we now consider experiments that use BERT$_{\text{BASE}}$ in various ways. We use two baselines: (1) BERT finetuned for NER using a local loss, and (2) a CRF using BERT features ("BERT-CRF"). Within our framework, we also experiment with using BERT in both the energy function and inference network architecture. That is, the "input feature vector" in Equation 3 is replaced by the features from BERT. The energy and inference networks are trained with the objective in Section 2.3. For the training of energy function and inference networks, we use Adam with learning rate $5\text{e}-5$, a batch size of 32, and L2 weight decay of $1\text{e}-5$. The results are shown in Table 7.[6]

There is a slight improvement when moving from BERT trained with the local loss to using BERT within the CRF (92.13 to 92.34). There is little difference (92.13 vs. 92.14) between the locally-trained BERT model and when using the linear-chain energy function within our framework. However, when using the higher-order energies, the difference is larger (92.13 to 92.46).

[6]Various high-order energies were explored. We found the skip-chain energy (M=3) to achieve the best performance (96.28) on the dev set, so we use it when reporting the test results.



(a) Skip-chain energy matrix W_1.



(b) Skip-chain energy matrix W_3.

Figure 1: Learned pairwise potential matrices W_1 and W_3 for NER with skip-chain energy. The rows correspond to earlier labels and the columns correspond to subsequent labels.

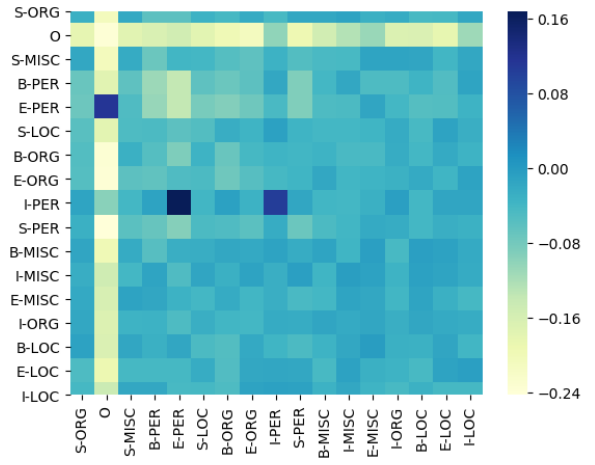


Figure 2: Learned 2nd-order VKP energy matrix beginning with B-PER in NER dataset.

filter 26	B-MISC	I-MISC	E-MISC
filter 12	B-LOC	I-LOC	E-LOC
filter 15	B-PER	I-PER	I-PER
filter 5	B-MISC	E-MISC	O
filter 6	O	B-LOC	I-LOC
filter 16	S-LOC	B-ORG	I-ORG
filter 44	B-PER	I-PER	I-PER
filter 3	B-MISC	I-MISC	E-MISC
filter 2	I-LOC	E-LOC	O
filter 45	O	B-LOC	E-LOC

Table 8: Top 10 CNN filters with high inner product with 3 consecutive labels for NER.

8 Analysis of Learned Energies

In this section, we visualize our learned energy functions for NER to see what structural dependencies among labels have been captured.

Figure 1 visualizes two matrices in the skip-chain energy with $M = 3$. We can see strong associations among labels in neighborhoods from W_1. For example, B-ORG and I-ORG are more likely to be followed by E-ORG. The W_3 matrix shows a strong association between I-ORG and E-ORG, which implies that the length of organization names is often long in the dataset.

For the VKP energy with M=3, Figure 2 shows the learned matrix when the first label is B-PER, showing that B-PER is likely to be followed by "I-PER E-PER", "E-PER O", or "I-PER I-PER".

In order to visualize the learned CNN filters, we calculate the inner product between the filter weights and consecutive labels. For each filter, we select the sequence of consecutive labels with the highest inner product. Table 8 shows the 10 filters with the highest inner product and the corresponding label trigram. All filters give high scores for structured label sequences with a strong local dependency, such as "B-MISC I-MISC E-MISC" and "B-LOC I-LOC E-LOC", etc. Figure 3 in the appendix shows these inner product scores of 50 CNN filters on a sampled NER label sequence. We can observe that filters learn the sparse set of label trigrams with strong local dependency.

9 Conclusion

We explore arbitrary-order models with different neural parameterizations on sequence labeling tasks via energy-based inference networks. This approach achieve substantial improvement using high-order energy terms, especially in noisy data conditions, while having same decoding speed as simple local classifiers.

Acknowledgments

We would like to thank Sam Wiseman for help regarding high-order CRFs and the reviewers for insightful comments. This research was supported in part by an Amazon Research Award to K. Gimpel.

References

David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *Proceedings of the 33rd International Conference on Machine Learning*.

David Belanger, Bishan Yang, and Andrew McCallum. 2017. End-to-end learning for structured prediction energy networks. In *Proceedings of the 34th International Conference on Machine Learning*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

William W. Cohen and Vitor Rocha de Carvalho. 2005. Stacked sequential learning. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 671–676.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*.

Nguyen Viet Cuong, Nan Ye, Wee Sun Lee, and Hai Leong Chieu. 2014. Conditional random field with high-order dependencies for sequence labeling and segmentation. *Journal of Machine Learning Research*, 15(28):981–1009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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 of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan. Association for Computational Linguistics.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 42–47.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain. European Language Resources Association (ELRA).

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Vijay Krishnan and Christopher D. Manning. 2006. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1121–1128, Sydney, Australia. Association for Computational Linguistics.

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 ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 260–270.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu-Jie Huang. 2006. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1381–1391, Baltimore, Maryland. Association for Computational Linguistics.

Bruce T. Lowerre. 1976. *The HARPY Speech Recognition System*. Ph.D. thesis, Pittsburgh, PA, USA.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074.

André Martins, Noah Smith, Mário Figueiredo, and Pedro Aguiar. 2011. Dual decomposition with many overlapping components. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 238–249, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA. Association for Computational Linguistics.

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *UAI*.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 161–168, Boston, Massachusetts, USA. Association for Computational Linguistics.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Xian Qian, Xiaoqian Jiang, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Sparse higher order conditional random fields for improved sequence labeling. In *ICML*.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 1–8, Boston, Massachusetts, USA. Association for Computational Linguistics.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Cambridge, MA. Association for Computational Linguistics.

Sunita Sarawagi and William W Cohen. 2005. Semi-Markov conditional random fields for information extraction. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1185–1192. MIT Press.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 145–156, Honolulu, Hawaii. Association for Computational Linguistics.

Vivek Srikumar and Christopher D Manning. 2014. Learning distributed representations for structured output prediction. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3266–3274. Curran Associates, Inc.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Charles Sutton and Andrew McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. In *ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of AAAI*.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin Markov networks. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32.

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.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-first International Conference on Machine Learning*.

Lifu Tu and Kevin Gimpel. 2018. Learning approximate inference networks for structured prediction. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Lifu Tu and Kevin Gimpel. 2019. Benchmarking approximate inference methods for neural structured prediction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3313–3324, Minneapolis, Minnesota. Association for Computational Linguistics.

Lifu Tu, Kevin Gimpel, and Karen Livescu. 2017. Learning to embed words in context for syntactic tasks. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 265–275.

Lifu Tu, Richard Yuanzhe Pang, and Kevin Gimpel. 2020a. Improving joint training of inference networks and structured prediction energy networks. *Proceedings of the 4th Workshop on Structured Prediction for NLP*.

Lifu Tu, Richard Yuanzhe Pang, Sam Wiseman, and Kevin Gimpel. 2020b. ENGINE: Energy-based inference networks for non-autoregressive machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Yi Yang and Jacob Eisenstein. 2013. A log-linear model for unsupervised text normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 61–72, Seattle, Washington, USA. Association for Computational Linguistics.

Nan Ye, Wee S. Lee, Hai L. Chieu, and Dan Wu. 2009. Conditional random fields with high-order features for sequence labeling. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 2196–2204. Curran Associates, Inc.

Mo Yu, Mark Dredze, Raman Arora, and Matthew R. Gormley. 2016. Embedding lexical features via low-rank tensors. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1019–1029, San Diego, California. Association for Computational Linguistics.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. 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 1127–1137, Beijing, China. Association for Computational Linguistics.

A Appendices

	POS		NER		CCG		SRL		
	Dev	Test	Dev	Test	Dev	Test	Dev	WSJ	Brown
BiLSTM	88.6	88.7	90.4	85.3	92.6	92.8	80.2	81.8	71.8
BiLSTM + CRF	89.1	89.2	91.6	87.3	93.0	93.1	-	-	-
Linear Chain	89.5	89.7	90.6	85.9	92.8	93.0	80.3	81.7	72.0
VKP (M=2)	89.9	90.1	91.1	86.5	93.1	93.3	80.3	81.8	72.0
VKP (M=3)	89.7	89.8	91.2	86.7	92.9	93.0	80.1	81.6	71.6
VKP (M=4)	89.4	89.5	90.8	86.3	92.8	93.0	79.9	81.2	71.3
VKP (M=[2,3,4])	89.8	89.9	91.0	86.5	93.0	93.3	80.3	81.9	71.8
Skip Chain (M=2)	89.7	89.8	90.8	86.2	92.8	93.1	80.3	81.8	71.8
Skip Chain (M=3)	89.9	90.0	91.2	86.5	93.0	93.3	80.4	82.1	72.4
Skip Chain (M=4)	89.8	89.9	91.3	86.7	92.7	92.8	80.2	81.6	71.7
Skip-Chain (M=5)	89.5	89.6	91.0	86.2	92.5	92.7	80.2	81.7	71.7
Fully Connect (M=20)	89.7	89.8	91.1	86.3	92.8	92.9	80.0	81.4	71.4
CNN (M=0)	89.6	89.8	90.9	86.2	92.6	92.8	80.0	81.7	71.8
CNN (M=1)	89.7	89.8	91.1	86.4	92.8	93.0	80.1	81.8	72.0
CNN (M=2)	90.0	90.1	91.3	86.5	93.0	93.2	80.3	81.9	72.2
CNN (M=3)	89.9	89.9	91.2	86.4	92.9	93.0	80.0	81.7	71.9
CNN (M=4)	89.7	89.8	91.0	86.2	93.0	93.1	80.2	81.7	72.2
CNN (M=1,2,3)	90.0	90.0	91.3	86.6	93.1	93.3	80.3	82.0	72.2
TLM (M=1)	89.6	89.7	90.9	86.3	92.4	92.6	79.8	81.3	71.3
TLM (M=2)	89.7	89.8	90.8	86.3	92.4	92.7	80.0	81.6	71.7
TLM (M=3)	89.8	89.8	91.0	86.4	92.7	92.9	80.1	81.7	71.9
TLM (M=4)	89.8	90.0	91.3	86.5	92.7	92.8	80.0	81.6	71.8
TLM	90.0	90.0	91.4	86.6	92.9	93.0	80.2	81.8	72.1
S-Att(M=2)	89.7	89.8	90.7	86.3	92.6	92.8	80.0	81.6	71.8
S-Att(M=4)	89.8	89.9	90.8	86.4	92.8	93.0	80.0	81.7	71.8
S-Att(M=6)	89.9	90.0	90.9	86.4	92.8	93.1	80.2	81.9	72.0
S-Att(M=8)	89.9	90.1	91.0	86.5	93.0	93.3	80.4	82.2	72.2
S-Att	89.7	89.9	90.8	86.4	93.1	93.3	80.3	82.0	72.2

Table 9: Results on all tasks for local classifiers and different structured energy functions: linear-chain energy, Kronecker Product high-order energies, skip-chain energy and fully-connected energies. The metrics of the four tasks POS, NER, CCG, SRL are accuracy, F1, accuracy and F1. The architecture of inference networks is one-layer BiLSTM.

	POS		NER		CCG	
	Dev	Test	Dev	Test	Dev	Test
2-layer BiLSTM	88.7	88.8	90.9	86.0	93.2	93.4
Linear Chain	89.9	90.0	91.2	86.6	93.3	93.7
Skip-Chain	90.0	90.2	91.7	87.5	93.5	93.8
VKP	89.9	90.2	91.5	87.2	93.6	93.8
CNN	90.0	90.2	91.5	87.3	93.5	93.6
TLM	89.9	90.1	91.4	87.1	93.3	93.6
S-Att (M=8)	89.9	90.0	91.6	87.3	93.5	93.7
Fully Connected	89.8	90.0	91.4	87.2	93.2	93.3

Table 10: Results when inference networks use 2-layer BiLSTMs (so the local classifier baseline also has 2 layers).

	α=0.1		α=0.2		α=0.3	
	Dev	Test	Dev	Test	Dev	Test
BiLSTM	80.0	75.0	70.1	67.2	62.4	58.8
Linear Chain	80.2	75.2	70.3	67.4	62.7	59.1
Skip-Chain (M=4)	80.6	75.5	70.9	67.9	63.2	59.5
VKP (M=3)	80.5	75.3	70.5	67.7	62.8	59.3
CNN (M=0)	80.8	75.7	71.3	67.9	63.3	59.4
CNN (M=2)	81.4	76.3	72.4	68.6	64.0	60.2
CNN (M=4)	81.9	76.7	73.0	69.8	64.5	60.4
TLM	81.0	76.0	71.3	67.8	63.8	59.9
S-Att (M=8)	80.6	75.6	71.5	67.6	63.2	59.7

Table 11: UnkTest setting for NER: Words in the test set are randomly replaced by the unknown word symbol with probability α.

	α=0.1		α=0.2		α=0.3	
	Dev	Test	Dev	Test	Dev	Test
BiLSTM	85.0	80.1	80.0	76.0	75.0	70.6
Linear Chain	85.4	80.4	80.5	76.3	75.2	70.9
Skip-Chain (M=4)	85.7	81.2	80.7	76.7	75.4	71.2
VKP (M=3)	85.9	81.4	81.0	76.8	75.5	71.4
CNN (M=0)	85.6	81.1	80.8	76.7	75.6	71.5
CNN (M=2)	86.0	81.8	81.2	77.0	76.1	71.8
CNN (M=4)	86.1	82.0	81.2	77.1	75.9	71.7
TLM	85.6	80.9	80.6	76.3	75.3	71.1
S-Att (M=8)	85.8	81.4	81.0	76.9	75.6	71.4

Table 12: UnkTrain setting for NER: training on noisy text, evaluating on noisy test sets. Words are randomly replaced by the unknown word symbol with probability α.

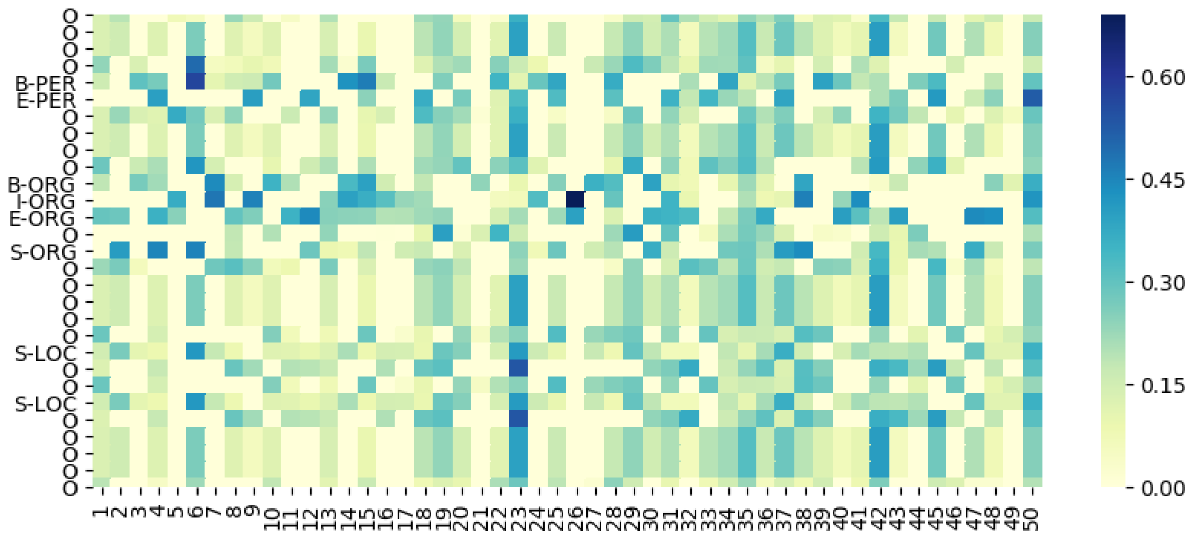


Figure 3: Visualization of the scores of 50 CNN filters on a sampled label sequence. We can observe that filters learn the sparse set of label trigrams with strong local dependency.