Uncertainty-Aware Sequence Labeling

Jiacheng Ye, Xiang Zhou, Xiaoqing Zheng, Tao Gui, and Qi Zhang

***Abstract*—Conditional random fields (CRFs) have been widely used for sequence labeling tasks in the field of natural language processing. However, how to model both local and global dependencies among labels is not well solved yet. In this study, we introduce a novel two-stage label decoding method to better model the short- and long-term label dependencies, while being much more computationally efficient with the use of graphics processing units (GPUs). A base model is first used to propose draft labels, and then a novel two-stream self-attention model makes refinements on these draft predictions based on long-range label dependencies. Besides, in order to mitigate the side effects of incorrect draft labels, Bayesian neural networks are used to indicate the labels with high probabilities of being wrong, which helps to mitigate the error propagation. Not only can our method model sentence-level label dependencies, but it is also easily extended to document-level sequence labeling by querying and storing a key-value memory matrix with label co-occurrence relationships. The experimental results on both sentence-level and document-level sequence labeling benchmarks show that the proposed method outperforms existing label decoding methods while taking advantage of parallel computations on GPUs.**

***Index Terms*—Sequence Labeling, Uncertainty Estimation, Bayesian Neural Network, Transformer, Memory Network.**

I. Introduction

THE sequence labeling aims to assign labels to input tokens (i.e., words or characters). There are many applications of sequence labeling in natural language processing (NLP), including part-of-speech (POS) tagging, text chunking, and named entity recognition (NER). Traditional models, such as Hidden Markov Models (HMM) and Conditional Random Fields (CRFs) [1], [2], [3], have been widely used for sequence labeling tasks for a long time. In the past two decades, neural network-based approaches achieved impressive results in many sequence labeling tasks without handcrafted feature engineering [4], [5], [6], [7], [8].

Along with the successful use of neural networks to obtain text distributed representations by the encoders, how to design a label decoder that can properly model the dependencies among labels also has drawn much attention recently [9], [10], [11]. For sequence labeling tasks, it is beneficial to consider the correlations among labels and jointly decode the best chain of labels for a given input sentence [4]. For example, in POS tagging an adjective is more likely to be followed by a noun than a verb (neighboring label dependencies), and in NER

This work is supported by China National Key Research and Development Program (No. 2018YFC0830900, 2018YFB1005104, 2018YFC0831105, 2017YFB1002104), National Natural Science Foundation of China (No. 62076068, 61751201, 61976056, 61532011), Shanghai Municipal Science and Technology Major Project (No.2018SHZDZX01), Science and Technology Commission of Shanghai Municipality (No.18DZ1201000, 16JC1420401, 17JC1420200). Corresponding author: Qi Zhang.

The authors are with the School of Computer Science, Fudan University, Shanghai 200433, China. E-mail: {yejc19, zhouxiang20, zhengxq, tgui16, qz, xjhuang}@fudan.edu.cn.

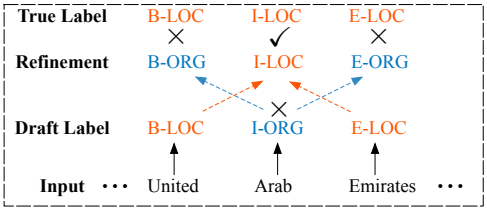


Fig. 1: Schematic diagram of label refinement framework [16]. The goal is to refine the label of "Arab" using contextual labels and words, while the refinement of other correct labels may be negatively impacted by incorrect draft labels.

the two named entities formed by coordinating conjunctions are likely to be the same type (long-term label dependencies) [12]. Therefore, it is critical for sequence label tasks to decode label sequences jointly using label dependencies, rather than of decoding each label independently [13]. Among them, the CRF layer integrated with neural encoders to capture label transition patterns has become ubiquitous in sequence labeling tasks [4], [14]. However, the linear-chain CRF usually can capture the neighboring label dependencies by using Viterbi decoding. Many of the recent methods try to introduce label embeddings to manage the dependencies at the longer ranges, such as two-stage label refinement [15], [16] and seqence-to-seqence (seq2seq) frameworks [11], [17]. Specifically, Krishnan and Manning [15] presented a two-stage approach where the second CRF uses features derived from the output of the first CRF to model long-term label dependencies. Zhang et al. [11] propose a seq2seq method and use a label long-short term memory network (LSTM) to augment the label dependencies and word-label interactions.

Although the above methods can model longer label dependencies in a sentence, they are vulnerable to error propagation: if a label is mistakenly predicted during inference, the error will be propagated and the other labels conditioned on this one will be negatively impacted [18]. We take the recently-proposed label attention network (LAN) [16] as an example, the LAN introduced a hierarchically-refined representation of marginal label distributions, which is used to predict a sequence of draft labels first and then uses the word-label interactions to refine them. As shown in Figure 1, when the LAN refine the label of "Arab" using contextual labels and words, the refinement of other correct labels may be negatively impacted by incorrect draft labels. To verify the existence of error propagation, we count the refinement results of the LAN on the CoNLL2003 test dataset, and find that there are 39 correct tokens have been incorrectly modified in all 93 entities that changed their labels after the refinement. Hence, the model should selectively correct the labels with high probabilities of being incorrect, not all of them. Fortunately, we find that

uncertainty values estimated by Bayesian neural networks [19] can effectively indicate the labels that have a high probability of being incorrect. We find that the average uncertainty[1] value of incorrect prediction is 29 times larger than that of correct predictions for the draft labels. Hence, we can easily set an uncertainty threshold to refine the potentially incorrect labels only and mitigate the side effects on the correct labels.

For document-level sequence labeling, modeling document-level label dependencies is also quite useful, because multiple occurrences of a particular token sequence are very likely to have the same entity types within a document [15]. As shown in Figure 2, previous methods only consider modeling sentence-level label dependencies and the document-level label dependencies are relatively less explored. The traditional linear-chain conditional random fields (CRFs) are insufficient for modeling the document-level relationships of labels [20], [21]. Recently, many studies have focused on the better incorporation of document-level context information [22], [20], but little attention has given to explicitly model the document-level label consistency among the same token sequences in an entire text.

In this study, we first propose a novel two-stage Uncertainty-Aware label refinement Network (UANet) to model sentence-level label dependencies. At the first stage, Bayesian neural networks take a sentence as input and yield all of the draft labels together with corresponding uncertainties. At the second stage, a two-stream self-attention model performs attention over label embeddings to explicitly model the label dependencies, and over context vectors to generate the context representations. All of these features are combined to refine the potentially incorrect draft labels. The above label refinement operations can be processed in parallel.

In addition, our method can be easily extended to further utilize document-level contextual and label information. We adopt a key-value memory component [23], which memorizes all the hidden state vectors and their corresponding label embeddings of the entire document. At the first stage, we process all the sentences in the document and store their hidden state vectors and the corresponding label embeddings into the key-value memory component, which are grouped by the unique word identifier. At the second stage, the additional document-level co-occurrence context and label vectors are retrieved by attention mechanism to provide additional document-level label dependencies, which will be combined with sentence-level context and label vectors to make the refinement.

We conduct experiments on four sequence labeling benchmarks to show the effectiveness of our model in sentence-level label dependency modeling. To further demonstrate that our model can easily incorporate document-level information, we also experiment on three document-based named entity recognition benchmarks. The experimental results on both the sentence-level and document-level sequence labeling tasks indicated that the proposed method not only outperformed state-of-the-art methods but also significantly reduced the inference time by leveraging GPUs for parallel decoding.

[1]The uncertainty was computed using the Bayesian neural network. Specifically, we used a Bayesian neural network in the first layer of the LAN to estimate the uncertainty.

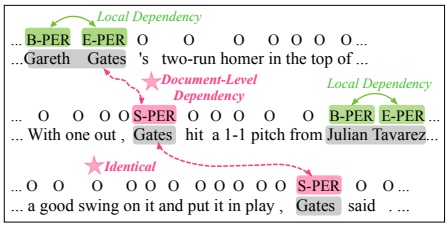


Fig. 2: Example of label dependencies in NER task. Local label dependencies (green, solid) can be learned by CRF-based methods. Although much recent work has focused on document-level context representations, the CRF decoding methods were still limited to learning document-level label dependencies (pink, dashed). In this example, "Gates" can either indicates entrances to a field or a person's name. But with the document-level label dependencies, the model can make the correct prediction.

The main contributions of this paper can be summarized as follows:

- We use Bayesian neural networks to estimate the uncertainty of predictions and indicate the potentially incorrect labels that should be refined.
- We propose a novel two-stream self-attention refining framework to better model label dependencies and word-label interactions at the short and long range.
- The proposed method can be easily extended to model document-level label dependencies by using a novel key-value memory component.
- The proposed parallel decoding process can greatly speed up the inference process by taking advantage of parallel computations on GPUs.
- The experimental results on both the sentence-level and document-level sequence labeling datasets indicate that the proposed methods perform better than existing label decoding methods. Besides, the experimental results on three document-level NER datasets indicate that the proposed method significantly outperforms the start-of-the-art methods. Our codes were released at *Github*[2].

This paper is organized as follows: Section II presents a brief overview of related work in sequence labeling, Bayesian modeling and refinement methods. Section III describes the proposed sentence-level and document-level models, together with their training and decoding details. Section IV shows the experimental results and extensive analysis on our sentence-level and document-level models. We conclude this paper in Section V.

II. Related Work

A. Sequence Labeling

Traditional sequence labeling models use statistical approaches with handcrafted features and task-specific resources [2], [24], [3]. With advances in deep learning, neural models could achieve competitive performances without handcrafted feature engineering [25], [26]. We mainly introduce several

[2]https://github.com/jiacheng-ye/DocL-NER

previous works from three aspects, namely input representation, text encoder, and label decoder.

1) Input Representation: For word-level representations, previous methods mainly use word embeddings pre-trained over large collections of text. Commonly used word embeddings include Word2Vec[3], GloVe[4], fastText[5] and SENNA[6]. For character-level representations, Ma and Hovy [4] encodes the character-level representation with a character-level convolutional neural network (CNN). More recently, some pre-trained contextualized embeddings, such as BERT [8], GPT [27], and ELMo [7], are proposed and demonstrate a strong ability for various natural language understanding tasks. In this work, we verify that the proposed label decoding method can effectively incorporate both the pre-trained word embeddings and pre-trained contextualized embeddings in the experiments.

2) Text Encoder: There are mainly three architectures used in encoder layer for sequence labeling tasks, namely recurrent neural networks (RNN), CNN and Transformer. Huang et al. [28] utilizes a bidirectinal LSTM architecture as context encoder. For a faster inference speed, [9], [6] propose to use CNN as context encoder and [29], [30] utilize Transformer to encode context information. [31] utilize Bayesian RNN [32] as context encoder to explicitly model uncertainties in the output. They find that modeling uncertainties is useful at enhancing model performances in Named Entity Recognition tasks. However, they only model uncertainty, but do not make further use of uncertainty. In this work, we further use uncertain information through a two-stage framework and achieve better results. More details about Bayesian neural networks will be described in the next subsection.

3) Label Decoder: The simplest decoder layer can be a MLP layer with softmax activation. However, pure MLP fails to model label dependencies, which are crucial in many sequence labeling tasks (e.g. POS tagging, Named Entity Recognition and Text Chunking). Many efforts have been done on how to model label dependencies, such as using a CRF layer integrated with neural encoders to capture label transition patterns [14], [4], and introducing label embeddings to manage longer ranges of dependencies [17], [11], [16]. Our work is an extension of label embedding methods, which applies label dependencies and word-label interactions to only refine the labels with high probabilities of being incorrect. The probability of making a mistake is estimated by a Bayesian neural network encoder.

B. Bayesian Neural Networks

Traditional neural networks are parameterized by a set of model weights $\mathbf{W}$, and a point estimation of $\mathbf{W}$ is obtained by maximizing a certain objective function. Bayesian neural networks (BNNs) [33], [34], [35], [36] offer a probabilistic interpretation of deep learning models by inferring distributions over the models' weights, i.e., $P(\mathbf{W}|\mathcal{D})$, given datasets $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_i)\}_{i=1}^{n}$. Bayesian neural networks can offer some robustness against the over-fitting problem, and make it possible to train models from small datasets [37].

1) Uncertainty Types: There are two main types of uncertainty in Bayesian modeling [19]. Aleatoric uncertainty captures the noise inherent in the observations, and epistemic (model) uncertainty accounts for the uncertainty in the model parameters. Aleatoric uncertainty could be sensor noise or annotation noise, resulting in uncertainty that cannot be reduced even if more data were to be collected. Model uncertainty captures our ignorance about which model generates our collected data. Model uncertainty can be reduced given enough data. This work focused on the use of model uncertainty to indicate whether model predictions were likely to be incorrect, which could effectively prevent correct draft labels from being incorrectly refined.

2) Uncertainty Estimation: In classification models, the probability vector obtained at the end of the pipeline (the softmax output) is often erroneously interpreted as model uncertainty. Malinin et al. [38] propose a framework called Prior Networks for classification task based on BNN uncertainty estimation. A model can be uncertain in its predictions even with a high softmax output [39], [37]. [39] gives results showing that simply using the point estimation results in extrapolations with unjustified high confidence for points far from the training data. They verify that modeling a distribution over the parameters through BNNs can effectively reflect the model uncertainty, and Bernoulli Dropout (which is also applied in this study) is exactly one example of regularization techniques, which corresponds to an approximate variational distribution without changing either the models or the optimization. Specifically, they prove that the use of Dropout (and its variants) in neural networks can be interpreted as a Bayesian approximation of Gaussian Process (GP) [40]. In Dropout case, weights are modeled as Bernoulli distribution with with probability p. Some typical examples of using Bernoulli distribution to estimate uncertainty are Bayesian CNN [41] and Bayesian RNN [32].

Here, we give a formal description of using Bayesian neural networks to estimate model uncertainty. Given the dataset $\mathcal{D}$ with training inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and their corresponding outputs $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$, Bayesian inference looks for the posterior distribution of the parameters $\mathbf{W}$ given the dataset $p(\mathbf{W}|\mathcal{D})$. This makes it possible to predict an output $\mathbf{y}^*$ for a new input point $\mathbf{x}^*$ by marginalizing over all of the possible parameters, as follows:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = \int p(\mathbf{y}^*|\mathbf{W}, \mathbf{x}^*) p(\mathbf{W}|\mathcal{D}) \mathrm{d}\mathbf{W}. \quad (1)$$

Bayesian inference is intractable for many models because of the complex nonlinear structures and high dimension of the model parameters. Recent advances in variational inference introduced new techniques into the field. Among these, Monte Carlo Dropout [39] requires minimum modification to the original model. It is possible to use the variational inference approach to find an approximation $q_\theta^*(\mathbf{W})$ to the true posterior $p(\mathbf{W}|\mathcal{D})$ parameterized by a different set of weights θ, where

[3]https://code.google.com/archive/p/word2vec
[4]http://nlp.stanford.edu/projects/glove
[5]https://fasttext.cc/docs/en/english-vectors.html
[6]https://ronan.collobert.com/senna

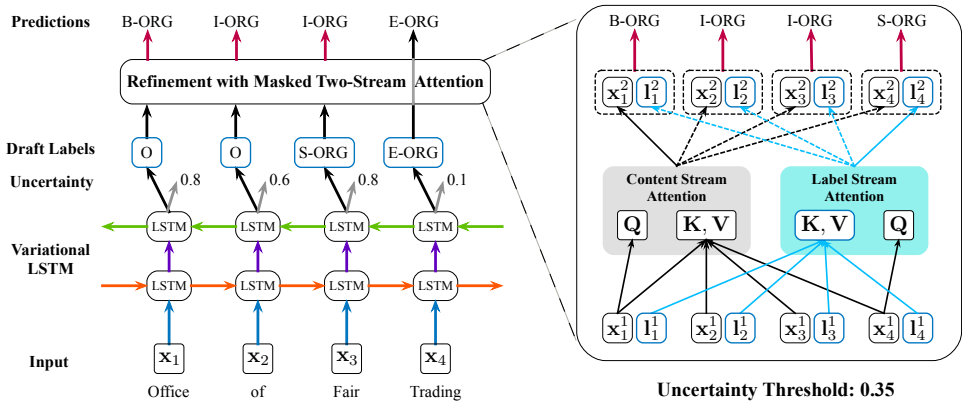


Fig. 3: Graphical illustration of architecture and inference process for the proposed UANet. The same colors between LSTM cells refer to repeating the same Dropout masks on recurrent connections at each time step. The variational LSTM outputs draft labels and model uncertainties simultaneously. The refinement works on draft labels only with threshold greater than 0.35.

the Kullback-Leibler (KL) divergence of the two distributions is minimized. The integral can be approximated as follows:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) \approx \sum_{j=1}^{T} p(\mathbf{y}^*|\mathbf{W}_j, \mathbf{x}^*) q_\theta^*(\mathbf{W}_j). \tag{2}$$

In contrast to non-Bayesian networks, at test time, Dropouts are also activated. As a result, model uncertainty can be approximately evaluated by summarizing the variance of the model outputs from multiple forward passes.

C. Refinement methods

Refinement (or reranking, rescoring) is a method to correct the draft outputs generated by the base model. A base model typically produces a set of candidates for each sentence, with associated probabilities that define an initial ranking of these outputs. A second model then improves upon the initial ranking or rewrite the candidate sentences, using additional features as evidence. Refinement approaches have given accuracy improvement on many NLP tasks including machine translation [42], [43], [44], [45], parsing [46], [47], reading comprehension [48], question answering [49], story generation [50] and named entity recognition [51], [52], [16].

Collins [51] proposed a voted perceptron algorithm for sequence labeling, which yields a significant improvement over the maximum-entropy baseline on the NER tasks by leveraging global features. Yang et al. [52] replace all entity mentions produced by a baseline NER model into their entity types and then leverage RNN to learn sentence-level patterns that involved named entity mentions. Cui et al. [16] investigate a hierarchically-refined label attention network for sequence labeling, which leverages label embeddings and uses consecutive attention layers on the label embeddings to refine the draft labels. In our work, not only do we use a novel two-stream self-attention model to generate refined labels, but we use uncertainty to determine whether to preserve the original labels or use the refined ones.

III. Uncertainty-Aware Sequence Labeling

In this work, we propose a novel two-stage label refinement framework for sequence labeling tasks. At the first stage, a variational LSTM [32] is adopted as the base model for the draft label and uncertainty predictions. At the second stage, the uncertain labels that have a high probability of being wrong can be refined by a two-stream self-attention model using long-term label dependencies and word-label interactions. The proposed sentence-level model is shown in Figure 3. In addition, the proposed method can be easily extended to further model document-level label dependencies through a key-value memory networks.

A. Variational LSTM for Uncertainty Estimation

Long short-term memory (LSTM) [53] stands at the forefront of many recent developments in sequence labeling tasks. Because most of previous sequence labeling tasks are based on the LSTM framework, to facilitate comparison with the previous LSTM-based models, we adopt variational LSTMs [32] as special Bayesian neural networks to encode sentences and determine the labels with high probabilities of being wrong. In general, the uncertainty estimation methods can also be easily applied to other sequence labeling models, like the CNN and Transformer.

1) Word and Label Representation: Following [26] and [5], we use character information to enhance the word representation. Given a word sequence $\mathbf{S} = w_1, w_2, \ldots, w_n$, the product of the one-hot encoded vector with an embedding matrix then gives a word embedding: $\mathbf{w}_i = \mathbf{e}^w(w_i)$, where $\mathbf{e}^w$ denotes a word embedding lookup table. Each word is made up of a sequence of characters $c_1, c_2, \ldots, c_t$. We adopt CNNs for character encoding and $\mathbf{x}_i^c$ denotes the output of character-level encoding. Then a word is represented by concatenating its word embedding and its character-level encoding: $\mathbf{x}_i = [\mathbf{w}_i; \mathbf{x}_i^c]$. All the word representations make up an embedding matrix $\mathbf{E} \in \mathbb{R}^{V \times D}$, where D is the embedding dimensionality of $\mathbf{x}$ and V is the number of words in the vocabulary.

Given the label set $L = \{l_1, l_2, \ldots, l_K\}$, each label l_k is represented by $\mathbf{l}_k = \mathbf{e}^l(l_k) \in \mathbb{R}^{d_l}$, where $\mathbf{e}^l$ denotes a

label embedding lookup table. Label embeddings are randomly initialized and tuned during training.

2) Variational LSTM: A common practice of Dropout technique on LSTM is that the technique should be used with the inputs and outputs of the LSTM alone. In contrast, the variational LSTM additionally applies Dropout on recurrent connections by repeating the same mask at each time step. Hence, the variational LSTM can model the uncertainty more accurately. In order to show the between vanilla LSTM and variational LSTM, we first describe the vanilla LSTM as follows.

$$\begin{aligned}\begin{bmatrix}\mathbf{g}_i\\ \mathbf{i}_i\\ \mathbf{f}_i\\ \mathbf{o}_i\end{bmatrix} &= \left(\begin{bmatrix}\mathbf{W}^g\\ \mathbf{W}^i\\ \mathbf{W}^f\\ \mathbf{W}^o\end{bmatrix} \bullet \begin{bmatrix}\mathbf{x}_i'\\ \mathbf{h}_{i-1}\end{bmatrix} + \begin{bmatrix}\mathbf{b}^g\\ \mathbf{b}^i\\ \mathbf{b}^f\\ \mathbf{b}^o\end{bmatrix}\right)\\ \mathbf{c}_i &= \phi(\mathbf{g}_i) \odot \sigma(\mathbf{i}_i) + \mathbf{c}_{i-1} \odot \sigma(\mathbf{f}_i)\\ \mathbf{h}_i &= \sigma(\mathbf{o}_i) \odot \phi(\mathbf{c}_i),\end{aligned} \tag{3}$$

where ϕ denotes the `tanh` function, and σ is the `sigmoid` function. We use the following symbols to denote the four gates of LSTM: "input modulation" gate is denoted by g, "input" gate by i, "forget" gate by f, and "output" gate by o. The symbols of $\odot$ and $\bullet$ are used to denote the Hadamard product and matrix product respectively, and c, h denote the cell state and hidden state, respectively.

For variational LSTM as shown in Figure 3, we use the same Dropout vectors $\mathbf{z}_x$ and $\mathbf{z}_h$ on four gates: "input modulation", "input", "forget", and "output" as follows:

$$\begin{aligned}\begin{bmatrix}\mathbf{g}_i\\ \mathbf{i}_i\\ \mathbf{f}_i\\ \mathbf{o}_i\end{bmatrix} &= \left(\begin{bmatrix}\mathbf{W}^g\\ \mathbf{W}^i\\ \mathbf{W}^f\\ \mathbf{W}^o\end{bmatrix} \bullet \begin{bmatrix}\mathbf{x}_i' \odot \mathbf{z}_x\\ \mathbf{h}_{i-1} \odot \mathbf{z}_h\end{bmatrix} + \begin{bmatrix}\mathbf{b}^g\\ \mathbf{b}^i\\ \mathbf{b}^f\\ \mathbf{b}^o\end{bmatrix}\right)\\ \mathbf{c}_i &= \phi(\mathbf{g}_i) \odot \sigma(\mathbf{i}_i) + \mathbf{c}_{i-1} \odot \sigma(\mathbf{f}_i)\\ \mathbf{h}_i &= \sigma(\mathbf{o}_i) \odot \phi(\mathbf{c}_i),\end{aligned} \tag{4}$$

We assume that t is one of $\{g, i, f, o\}$. Then, $\theta = \{\mathbf{E}, \mathbf{W}^t\}$, where $\mathbf{E}$ is word embedding, and the Dropout rate r are the parameters of the variational LSTM.

3) Draft Labels and Uncertainty Estimation: Assuming that we have completed the training and obtained the optimized approximated posterior $q_\theta^*(\mathbf{W})$ (the optimizing method is shown in § III-D), at inference time, we can predict an output for a new input point by performing Monte Carlo integration in Eq.2 as follows:

$$\mathbf{p}_i \approx \frac{1}{M}\sum_{j=1}^{M} \texttt{Softmax}(\mathbf{h}_i|\mathbf{W}_j) \tag{5}$$

with M sampled masked model weights $\mathbf{W}_j \sim q_\theta^*(\mathbf{W})$, where $q_\theta^*(\mathbf{W})$ is the Dropout distribution. In order to fairly compare with the standard LSTM, we repeat the same input M times to form a batch. Since the computation can be done in parallel with GPUs, M samples are concurrently processed in a forward pass, leading to a constant running time that is identical to that of standard Dropout [39] (see Table V).

Similar to classic sequence labeling models, the model applies $l_i^* = \text{argmax}(\mathbf{p}_i)$ to obtain the draft label. Then the uncertainty of this probability vector $\mathbf{p}_i$ can be summarized using the entropy of the probability vector:

$$u_i = H(\mathbf{p}_i) = -\sum_{k=1}^{K} p_k \log p_k. \tag{6}$$

In this way, we can obtain the draft labels $L^* = \{l_1^*, l_2^*, \ldots, l_n^*\}$ coupled with the corresponding model uncertainties $U = \{u_1, u_2, \ldots, u_n\}$ for each input sentence. We find when the model uncertainty u_i is larger than some threshold value Γ, then the draft label l_i^* has a high probability of being wrong. Hence, we utilize a novel two-stream self-attention model to refine those uncertain labels using long-term label dependencies and word-label interactions.

B. Sentence-Level Label Refinement

Given the draft labels and corresponding model uncertainties, we seek the help of label dependencies and word-label interactions to refine the uncertain labels. In order to refine the draft labels in parallel, we use the Transformer [54] incorporating relative position encoding [55] to model the words and draft labels.

In the standard Transformer, the attention score incorporating absolute position encoding between query q_i and key vector k_j can be decomposed as

$$\begin{aligned}\mathbf{A}_{i,j}^{abs} &= \mathbf{E}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_j + \mathbf{E}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j\\ &\quad + \mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_j + \mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j,\end{aligned} \tag{7}$$

where $\mathbf{U} \in \mathbb{R}^{L_{max} \times d}$ provides a set of positional encodings. The ith row $\mathbf{U}_i$ corresponds to the ith absolute position and L_{max} prescribes the maximum possible length to be modeled.

The relative position between labels is very important for modeling the label dependencies. Inspired by [55], we modify the Eq.7 using the relative position encoding to model words and corresponding labels simultaneously, but offer a different derivation, arriving at a new form of two-stream relative positional encodings. We not only provide a word-to-word interactions but also provide a word-to-label interactions correspondence to its counterpart. The relative position encodings are reparameterized as follows:

$$\begin{aligned}\mathbf{A}_{i,j}^{x2x} &= \mathbf{E}_i^\top \mathbf{W}_{qx}^\top \mathbf{W}_{kx} \mathbf{E}_j + \mathbf{E}_i^\top \mathbf{W}_{qx}^\top \mathbf{W}_{kR} \mathbf{R}_{i-j}\\ &\quad + \mathbf{u}_x^\top \mathbf{W}_{kx} \mathbf{E}_{x_j} + \mathbf{v}_x^\top \mathbf{W}_{kR} \mathbf{R}_{i-j}\\ \mathbf{A}_{i,m}^{x2l} &= \mathbf{E}_i^\top \mathbf{W}_{ql}^\top \mathbf{W}_{kl} \mathbf{l}_m + \mathbf{E}_i^\top \mathbf{W}_{ql}^\top \mathbf{W}_{kR} \mathbf{R}_{i-m}\\ &\quad + \mathbf{u}_l^\top \mathbf{W}_{kl} \mathbf{l}_m + \mathbf{v}_l^\top \mathbf{W}_{kR} \mathbf{R}_{i-m},\end{aligned} \tag{8}$$

where $\mathbf{A}_{i,j}^{x2x}$ and $\mathbf{A}_{i,m}^{x2l}$ denotes the attention from the ith word (x_i) to the jth word (x_j) and the ith word (x_i) to the mth label (l_m^*), respectively. $\mathbf{R}_{i-j}$ is the encoding of relative distance between position i and j, and $\mathbf{R}$ is the sinusoid matrix like [55]. $\varphi = \{\mathbf{W}, \mathbf{u}, \text{and } \mathbf{v}\}$ are learnable parameters. $\mathbf{W}_k$, $\mathbf{W}_q$ are the weight matrices of key vector and query vector of attention, respectively. $\mathbf{u}$ is a trainable parameter to replace $\mathbf{U}_i^\top \mathbf{W}_q$ in the third term $\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_j$ of Eq.7. In this case, since the query vector is the same for all query positions, it suggests that the attentive bias towards different words should remain the same regardless of the query position. For the same

reason, $\mathbf{v}$ is added to substitute $\mathbf{U}_i^\top \mathbf{W}_q$ in the fourth term $\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j$ of Eq.7.

Equipping the Transformer with our proposed relative positional encoding, we finally arrive at the two-stream self-attention architecture. We summarize the computational procedure for one layer with a single attention head here:

$$\begin{aligned}
\mathbf{V}_x &= \mathbf{E}_x \mathbf{W}_x, \mathbf{a}_x = \texttt{Softmax}(\mathbf{A}^{x2x})\mathbf{V}_x, \mathbf{E}_x = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \\
\mathbf{V}_l &= \mathbf{L}\mathbf{W}_l, \mathbf{a}_l = \texttt{Softmax}(\mathbf{A}^{x2l})\mathbf{V}_l, \mathbf{L} = \{\mathbf{l}_1, \ldots, \mathbf{l}_n\} \\
\mathbf{H}_x &= \texttt{FeedForward}(\texttt{LayerNorm}(\texttt{Linear}(\mathbf{a}_x) + \mathbf{E}_x)) \\
\mathbf{H}_l &= \texttt{FeedForward}(\texttt{LayerNorm}(\texttt{Linear}(\mathbf{a}_l) + \mathbf{L})).
\end{aligned} \tag{9}$$

where $\mathbf{H}_x$ and $\mathbf{H}_l$ denote the hidden representation of sequences and labels, respectively. $\mathbf{L}$ is the label embedding of the sequence.

C. Document-Level Label Refinement

Our method can be easily extended to further model document-level label consistency. The architecture of document-level UANet is shown in Figure 4. To model document-level label consistency, we introduce a key-value memory network [23] to simultaneously record the context representations (key) and corresponding label embeddings (value) for each word. Specifically, we create a document-level memory matrix $\mathbf{M} = \{\mathbf{m}_{w_1}, \ldots, \mathbf{m}_{w_m}\}$, in which the same words under different context would occupy many different slots and form a quired subset $\mathbf{m}_{w_i}$. The memory slots in one subset $\mathbf{m}_{w_i}$ are defined as pairs of vectors $(k_{i;1}, v_{i;1}), \ldots, (k_{i;m}, v_{i;m})$. In every single slot j, the key represents a certain hidden state vector $\mathbf{h}_{i;j}$, and the value is the corresponding label embedding $\mathbf{l}_{i;j}$.

1) Store into Memory: We process every sentence of a document in parallel at the first stage. Then, we store the predicted label embedding $\mathbf{l}_{i;j}$ and the hidden vector $\mathbf{h}_{i;j}$ of every word w_i in a document into memory. Thus, every slot contains all the occurrence context and label information of a word in a document. The entire slot will be used as document-level information of a word at the second stage.

2) Read from Memory: At the second stage, the draft predictions are refined based on the explicit co-occurrence relationship derived from the key-value memory network.

With the constructed memory matrix, a key addressing and value reading mechanism is designed to access the document-level context representations and label consistencies. During addressing, each slot of the corresponding subset is assigned a relevance probability by comparing the word to each key:

$$p_{h_{i;j}} = \texttt{softmax}(\mathbf{x}_i^\top W_h \mathbf{h}_{i;j}), \tag{10}$$

where W_h is a $d_w \times d_h$ matrix. We hope that the model can refer to the semantics and corresponding labels of other sentences in the document when predicting the label of a word. Hence, in the reading step, the keys and values of the memory are read by taking their weighted sum using the addressing probabilities, and the document-level representations and label embeddings are returned:

$$\mathbf{h}_i = \sum_j p_{h_{i;j}} \mathbf{h}_{i;j}; \quad \mathbf{l}_i = \sum_j p_{h_{i;j}} \mathbf{l}_{i;j}. \tag{11}$$

3) Two-Stream Self-Attention: We further use the two-stream self-attention incorporating relative position encoding to model document-level dependencies in parallel. We propose to reparameterize the relative position encoding as follows:

$$\begin{aligned}
\mathbf{A}_{i,j}^{h2h} &= \mathbf{h}_i^\top \mathbf{W}_{qh}^\top \mathbf{W}_{kh} \mathbf{h}_j + \mathbf{h}_i^\top \mathbf{W}_{qh}^\top \mathbf{W}_{kR} \mathbf{R}_{i-j} \\
&\quad + \mathbf{u}_h^\top \mathbf{W}_{kh} \mathbf{h}_j + \mathbf{v}_h^\top \mathbf{W}_{kR} \mathbf{R}_{i-j} \\
\mathbf{A}_{i,m}^{h2l} &= \mathbf{h}_i^\top \mathbf{W}_{ql}^\top \mathbf{W}_{kl} \mathbf{l}_m + \mathbf{h}_i^\top \mathbf{W}_{ql}^\top \mathbf{W}_{kR} \mathbf{R}_{i-m} \\
&\quad + \mathbf{u}_l^\top \mathbf{W}_{kl} \mathbf{l}_m + \mathbf{v}_l^\top \mathbf{W}_{kR} \mathbf{R}_{i-m},
\end{aligned} \tag{12}$$

where $\mathbf{A}_{i,j}^{h2h}$ and $\mathbf{A}_{i,m}^{h2l}$ denotes the attention from $\mathbf{h}_i$ to $\mathbf{h}_j$ and $\mathbf{h}_i$ to $\mathbf{l}_m$, respectively. $\mathbf{R}_{i-j}$ is the encoding of the relative distance between position i and j, and $\mathbf{R}$ is a sinusoid matrix. $\varphi = \{\mathbf{W}$, $\mathbf{u}$, and $\mathbf{v}\}$ are learnable parameters.

Similar to Eq.9, We summarize the computational procedure for one layer with a single attention head as follows:

$$\begin{aligned}
\mathbf{V}_h &= \mathbf{H}\mathbf{W}_h, \mathbf{a}_h = \texttt{Softmax}(\mathbf{A}^{h2h})\mathbf{V}_h, \mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_n\} \\
\mathbf{V}_l &= \mathbf{L}\mathbf{W}_l, \mathbf{a}_l = \texttt{Softmax}(\mathbf{A}^{h2l})\mathbf{V}_l, \mathbf{L} = \{\mathbf{l}_1, \ldots, \mathbf{l}_n\} \\
\mathbf{H}_x &= \texttt{FeedForward}(\texttt{LayerNorm}(\texttt{Linear}(\mathbf{a}_h) + \mathbf{H})) \\
\mathbf{H}_l &= \texttt{FeedForward}(\texttt{LayerNorm}(\texttt{Linear}(\mathbf{a}_l) + \mathbf{L})).
\end{aligned} \tag{13}$$

D. Training and Decoding

There are two networks to be optimized: one is variational LSTM for draft labels and uncertainty estimation, the other is two-stream self-attention model for label refinement. The two models are optimized jointly: $\mathcal{L}_{total} = \mathcal{L}_1(\theta, r) + \mathcal{L}_2(\varphi)$. $\theta = \{\mathbf{E}, \mathbf{W}^t\}$, where $\mathbf{E}$ is the word embedding, and $\mathbf{W}^t$ is the parameter of LSTM. r is the Drop rate. $\varphi = \{\mathbf{W}$, $\mathbf{u}$, and $\mathbf{v}\}$, where $\mathbf{W}$ are the weight matrices of attention vectors and $\mathbf{u}$, $\mathbf{v}$ are trainable parameters in Eq.8.

The variational LSTM performs approximate variational inference. We use a simple Bernoulli distribution (Dropout) $q_\theta^*(\mathbf{W})$ in a tractable family to minimize the KL divergence to the true model posterior $p(\mathbf{W}|\mathcal{D})$. The minimization objective is given by [56]:

$$\mathcal{L}_1(\theta, r) = -\frac{1}{N} \sum_{i=1}^{N} \log p(y_i|\mathbf{W}_j) + \frac{1-r}{2N} \parallel \theta \parallel^2, \tag{14}$$

where N is the number of data points, $p(y_i|\mathbf{W}_j)$ is the marginal probability of $p(\mathbf{y}^*|\mathbf{W}_j, \mathbf{x}^*)$, $\mathcal{L}_1(\theta, r)$ is the lower bound of the log likelihood of $p(\mathbf{W}|\mathcal{D})$, and r is the Dropout probability to sample $\mathbf{W}_j \sim q_\theta^*(\mathbf{W})$.

For the two-stream self-attention model, we use the concatenation of $\mathbf{H}_x$ and $\mathbf{H}_l$ for the final prediction $\hat{y}_i = f(\mathbf{H}_x, \mathbf{H}_l|\mathbf{E}_x, \mathbf{L})$, where $f(\cdot)$ is a linear transformation layer. In particular, we can optimize the model using cross entropy loss as:

$$\mathcal{L}_2(\varphi) = -\sum_{i=1}^{N} y_i \log \hat{y}_i, \tag{15}$$

where y_i is the one-hot vector of the label corresponding to w_i. When training is complete, we can obtain the draft labels $Y^* = \{y_1^*, y_2^*, \ldots, y_n^*\}$ and corresponding uncertainties $U = \{u_1, u_2, \ldots, u_n\}$ from variational LSTM, and refined labels $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n\}$ from two-stream self-attention model.

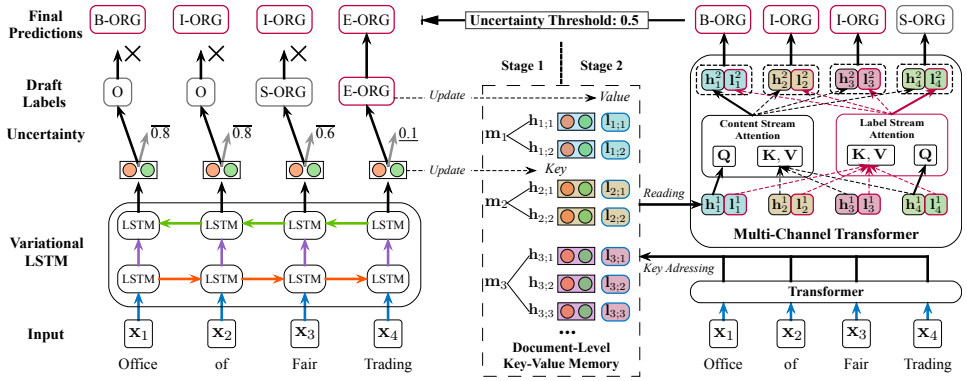


Fig. 4: Architecture of Doc-UANet. The refinement works only on draft labels with uncertainty greater than the threshold. For example, the threshold is set to 0.5 in the figure, and the final predictions are those labels in red blocks.

To avoid the correct labels being incorrectly modified, we set an uncertainty threshold Γ to distinguish which labels should be used, i.e., we use refined labels when $u_i > \Gamma$ and vice versa (as an example, given $u_1 > \Gamma$, $u_2 \leq \Gamma$, and $u_n > \Gamma$, decoding labels will become $\{\hat{y}_1, y_2^*, \ldots, \hat{y}_n\}$).

IV. Experimental Setup

In this section, we will first detail the datasets we used. Then, we will describe several baseline methods, including a number of sentence-level taggers and a series of document-level sequence labeling methods. We also detail the hyperparameter configuration of the proposed model.

A. Datasets

We conduct sentence-level experiments on CoNLL2003, OntoNotes 5.0, WSJ, and CHEMDNER datasets. In addition, the CoNLL2003, OntoNotes 5.0, and CHEMDNER datasets can also be used in document-level experiments. The statistics of datasets are listed in Table I.

WSJ. Wall Street Journal portion of Penn Treebank [57] contains 45 types of part-of-speech tags. We adopt standard splits following previous works [58], [59], selecting section 0-18 for training, section 19-21 for validation and section 22-24 for test. We conduct sentence-level experiments on this dataset.

CoNLL2003. The shared task of CoNLL2003 dataset [60] for named entity recognfition is collected from Reuters Corpus. The dataset divide name entities into four different types: persons, locations, organizations, and miscellaneous entities. We use the BIOES tag scheme instead of standard BIO2, which is the same as [4]. In addition to sentence-level experiments, the original files use -DOCSTART- as document separator and we keep this to conduct our document-level experiments.

OntoNotes 5.0. English NER dataset OntoNotes 5.0 [61] is a large corpus consists of various genres, such as newswire, broadcast, and telephone speech. Named entities are labeled in eleven types and values are specifically divided into seven types, like DATE, TIME, ORDINAL. In addition to sentence-level experiments, we regard *Part number* as document indicator in our experiments. For simplicity, we will use OntoNotes to represent OntoNotes 5.0 in the following description.

Dataset	Type	Train	Dev	Test
WSJ	#doc	-	-	-
	#sent	38,219	5,527	5,462
CoNLL2003	#doc	946	216	231
	#sent	14,041	3,250	3,453
OntoNotes 5.0	#doc	2,483	319	322
	#sent	59,924	8,528	8,262
CHEMDNER	#doc	3,500	3,500	3,000
	#sent	30,802	30,807	26,435

TABLE I: Statistics of WSJ (POS tagging), CoNLL2003 (NER), OntoNotes 5.0 (NER) and CHEMDNER (NER) datasets.

CHEMDNER. The CHEMDNER corpus consists of 10,000 PubMed abstracts published in the top journals from various chemistry-related disciplines, which contains a total of 84,355 chemical entity mentions labeled manually by expert chemistry literature curators [62]. The corpus is tagged with only one chemical entity type. We regard each abstract as a document in our experiments.

B. Sentence-Level Comparison Methods

In this work, we mainly focus on improving decoding efficiency and enhancing label dependencies. Thus, we make comparisons with the classic methods that have different decoding layers, such as Softmax, CRF, and LAN frameworks. We also compare some recent competitive methods, such as Transformer, IntNet [63], and BERT [8].

BiLSTM-Softmax. This baseline uses bidirectional LSTM to reprensent a sequence. The BiLSTM concatenates the forward hidden state $\overrightarrow{\mathbf{h}}_i$ and backward hidden state $\overleftarrow{\mathbf{h}}_i$ to form an integral representation $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$. Finally, sentence representation $\mathbf{H} = \{\mathbf{h}_i, \cdots, \mathbf{h}_n\}$ is fed to softmax layer for predicting.

BiLSTM-CRF. A CRF layer is used on top of the hidden vectors $\mathbf{H}$ [4]. The CRF can model bigram interactions between two successive labels [5] instead of making independent labeling decisions for each output. In the decoding time, the Viterbi algorithm is used to find the highest scored label sequence over an input word sequence.

BiLSTM-Seq2seq. To model longer label dependencies, [11] predicts a sequence of labels as a sequence to sequence problem.

BiLSTM-LAN. The label attention network (LAN) [16] introduces label embedding, and uses consecutive attention layers on the label embeddings to refine the draft labels. The representation for each label-attention layers is the concatenation of primitive BiLSTM hidden states and label embedding. In the last layer, the model looks up for the corresponding label with maximum attention weights for each word. It achieves the state-of-the-art results on several sequence labeling tasks.

Rel-Transformer. This baseline model adopts self-attention mechanism with relative position representations [54], [55]. Specifically, the model processes word representations with relative positional representations by a list of self-attention layers and feed-forward neural network, end up giving labels with a softmax layer.

C. Document-Level Comparison Methods

For document-level experiments, the proposed method compares favourably with state-of-the-art methods, such as GraphIE [20] and Hier-NER [21].

GraphIE. GraphIE [20] utilizes a co-occurrence graph to incorporate document-level contextual information and CRF to model sentence-level label dependency. This methods achieves great performances in many information extraction tasks, including textual, social media and visual information extraction.

Hier-NER. Hier-NER [21] introduces a corpus-level memory mechanism to utilize the global contextual information of a token. Moreover, a sentence-level encoder is used to enhance the sentence representation learned from an independent BiLSTM. A CRF layer is used as the final decoder in Hier-NER. With two-level hierarchical representations, Hier-NER established state-of-the-art results on several NER tasks.

D. Hyper-parameter Settings

1) Sentence-Level Experiments' Settings: Following [4], we use the same 100-dimensional GloVe embeddings[7] as initialization. We use 1-layer variational LSTM with a hidden size of 400 to create draft labels. We use 2 layers of multi-head Transformer for WSJ and CoNLL2003 and 3 for OntoNotes dataset to refine the label. The number of heads is chosen from {5, 7, 9}, and the dimension of each head is chosen from {80, 120, 160} via grid search. In our experiments, we set the number of heads and the dimension of each head to {7, 80}, {5, 160}, and {5, 160} for CoNLL2003, OntoNotes, and WSJ datasets, respectively.

2) Document-Level Experiments' Settings: Following [4], [20], [21], we use the same 100-dimensional Glove embeddings for the CoNLL2003 and OntoNotes datasets. For CHEMDNER, we use 50-dimensional pretrained word2vec embeddings [64], which is the same as [20]. For the first stage, we use 1 layer of variational LSTM with 200 dimensions for CoNLL2003 and 2 layers for the other datasets. For the second stage, we use 3 layers of Transformer for CoNLL2003 dataset, and 4 layers for the other datasets.

For CharCNN, we use 32-dimensional character embeddings and 32 filters of width 3 for CoNLL2003, OntoNotes, and WSJ datasets. We use 128-dimensional character embeddings and 128 filters of width 2 to 4 for CHEMDNER. For all the experiments, we use 400-dimensional label embedding with randomly initialization. The vanilla dropout after the embedding layer and the variational dropout is set to 0.5 and 0.25, respectively. We use Adam [65] as the optimizer for Transformer and SGD for variational LSTM. Learning rates are set to 0.015 for SGD and 0.0001 for Adam, respectively. The number of samples is set to 32. We use the BIOES tag scheme instead of standard BIO2, as previous studies have reported a meaningful improvement with this scheme [4], [5]. F1 score and accuracy are used for NER and POS tagging, respectively. All parameters are initialized by the default methods in Pytorch[8]. All experiments are implemented in NCRF++ [66] and conducted using a GeForce GTX 1080Ti with 11GB memory.

Models	CoNLL2003	OntoNotes	WSJ	CHEMDNER
Chiu and Nichols (2016) [25]	90.91	86.28	-	-
Strubell et al. (2017) [6]	90.54	86.84	-	-
Liu et al. (2018) [67]	91.24	-	97.53	-
Chen et al. (2019) [68]	91.44	87.67	-	-
BiLSTM-CRF [4]	91.21	86.99	97.55	89.45
BiLSTM-Softmax [69]	90.77	83.76	97.51	88.15
BiLSTM-Seq2seq [11]	91.22	-	97.59	89.50
Rel-Transformer [55]	90.70	87.45	97.49	87.89
BiLSTM-LAN [16]	90.77*	88.16	97.58	86.94
BiLSTM-UANet ($M = 8$)	**91.60**	**88.39**	**97.62**	**90.15**
GraphIE †	91.74	87.43*	-	89.71
Hier-NER †	91.96	87.98	-	89.53*
Doc-BiLSTM-UANet †	**92.13**	**88.49**	-	**90.72**

TABLE II: Main results on four sequence labeling datasets. The results with † are the document-level experiments. * indicates the results by running [16]'s released code.

V. Results and Analysis

In this section, we present the experimental results of the proposed and baseline models. We show that the proposed method not only can achieve the better performance but also is friendly to GPU architectures which leads to a great increase in the decoding speed in a parallel way. Since our contribution is mainly focused on the label decoding layer, the proposed model can also be combined with the latest pre-trained model to further improve performance.

A. Main Results

Table II reports model performances on CoNLL2003, OntoNotes, WSJ, and CHEMDNER datasets, which shows that the proposed method not only can achieve state-of-the-art results on NER task but also is effective on other sequence labeling tasks, like POS tagging. The previous methods leverage rich handcrafted features [28], [25], CRF decoding [6], and longer range label dependencies [11], [16]. Compared with these methods, our UANet model gives better results.

[7]http://nlp.stanford.edu/projects/glove/

[8]https://pytorch.org/docs/1.2.0/

Models	F_1
BERT-Softmax [8]	91.62
BERT-CRF	91.71
BERT + UANet	**92.02**
BERT + Doc-BiLSTM-UANet	**92.92**
ELMo-softmax [7]	92.22
ELMo-CRF	92.40
ELMo-UANet	**92.83**
ELMo + Doc-BiLSTM-UANet	**93.05**

TABLE III: Results on CoNLL2003 test set. We implement BERT for NER task without document-level information. Original result of BERT in [8] was not achieved with the current version of the library. See a discussion in [70] and the reported results at [71].

The results show that the UANet outperforms models with the CRF inference layer by a large margin, we speculate that's benefited from the strong capability of modeling long-term label dependencies. And it also outperforms LAN and seq2seq models on all of the four datasets, we speculate that's because our UANet model integrates model uncertainty into the refinement stage to avoid side effects on correct draft labels.

For document-level sequence labeling tasks, we also compare the document-level BiLSTM-UANet (Doc-BiLSTM-UANet) with state-of-the-art document-level methods. The models incorporating document-level context information (GraphIE and Hier-NER) can outperform those without. Because of the novel design of our document-level refinement networks, Doc-BiLSTM-UANet can utilize not only document-level context information, but also document-level label consistency. Hence, Doc-BiLSTM-UANet obtains significant improvement compared with the BiLSTM-CRF, GraphIE and Hier-NER models. Moreover, Doc-BiLSTM-UANet also outperforms the models that exploit additional task-specific resources or annotated corpora [25].

We also use the pretrained models (BERT, ELMo) to replace the default embeddings. We fine-tuned the BERT and ELMo in our experiments. The results are shown in Table III. We find that by adding our UANet on the BERT and ELMo embeddings, the F1 score on the CoNLL2003 dataset further improves 0.40% and 0.61%, respectively. By our speculation, the insufficient performance of BERT and ELMo may due to the inability to model document-level label consistency. If we further allow our method to model document-level label dependencies (Doc-BiLSTM-UANet), the BERT+Doc-BiLSTM-UANet can even exceed the BERT-Softmax by 1.3%.

B. Ablation Study

To study the contribution of each component in BiLSTM-UANet and Doc-BiLSTM-UANet, we conducted ablation experiments on the four datasets and display the results in Table IV. The experimental results on the sentence-level sequence labeling tasks show that the performance of the model will be degraded if the draft label information is not taken into account, indicating that label dependencies are useful in the refinement. We also find that both the variational LSTM and two-stream self-attention play an important role in label refinement. Even though we replace any component with the CRF layer, the performance will be seriously hurt.

Sentence-Level Models	CoNLL2003	OntoNotes	WSJ
BiLSTM-UANet	**91.60**	**88.39**	**97.62**
- Label information	91.23	87.84	97.57
- Variational LSTM [1]	90.70	87.45	97.49
- Two-stream self-attention [2]	90.83	87.11	97.46
Rel-Transformer-CRF	91.22	87.77	97.56
Variational LSTM-CRF	91.20	87.63	97.55
Document-Level Models	**CoNLL2003**	**OntoNotes**	**CHEMDNER**
Doc-BiLSTM-UANet	**92.13**	**88.49**	**90.72**
- document-level label	91.81	88.20	90.36
- document-level context	91.46	88.00	90.12
- both	91.36	87.76	89.83
- sentence-level label	91.63	88.05	90.21
+ CRF	92.05	88.28	90.69
- refinement stage	90.83	87.09	89.43

[1] This is equivalent to Variational LSTM-Softmax.
[2] This is equivalent to Rel-Transformer-Softmax.

TABLE IV: Ablation study of the proposed model. "+" means the information or component is added in the models and "-" means the information or component is removed.

	CoNLL2003	OntoNotes	WSJ
Average Sentence Length	13	18	24
BiLSTM-CRF	1,433	950	801
BiLSTM-LAN	949	773	943
BiLSTM-Seq2seq	1,084	842	751
BiLSTM-UANet ($M = 1$)	1,630	1,262	1,192
BiLSTM-UANet ($M = 8$)	1,474	1,129	1,044

TABLE V: Comparison of sentence-level inference speed. We show how many sentences the model can process per second with the use of GPUs.

For document-level experiments, the results show that the model's performance degraded when the document-level label information is not used, indicating that the previous methods that only incorporate document-level contextual information are insufficient to model the dependency between labels. More notably, we discover that results can be improved more when using document-level label and document-level context together than simply adding up the separate increases, which verifies the effectiveness of the proposed model. Since we also use the Transformer to model sentence-level label dependencies instead of CRF, we further investigate its advantage. As shown in the second part of Table IV, our method outperforms CRF since we utilize Transformer with relative position embedding to capture long-term label dependency, while CRF only considers the neighboring label dependencies. Moreover, an accompanying additional benefit is a faster training and decoding speed, as we have mentioned earlier. Furthermore, when we remove the second stage, the performance drops by 1.30%, 1.31%, and 1.29% for CoNLL2003, OntoNotes, and CHEMDNER, respectively, indicating that our refinement method is useful and can yield significant improvements.

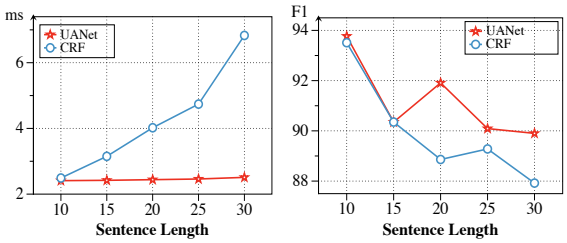


Fig. 5: Speed and F1 against sentence length.

Models	Train	Inference	Co-Acc
Hier-NER	1.00x	1.00x	92.74
GraphIE	1.78x	2.86x	93.05
BiLSTM-CRF	2.22x	4.76x	91.33
Doc-BiLSTM-UANet	**2.64x**	**5.48x**	**93.36**

TABLE VI: Document-level speed and Co-Acc comparison on CoNLL2003 datasets. Co-Acc refers to the accuracy of co-occurrence tokens.

C. Efficiency Advantage

Table V shows a comparison of inference speeds. BiLSTM-UANet processes 1,630, 1,262, and 1,192 sentences per second on the CoNLL2003, OntoNotes, and WSJ development data, respectively, outperforming BiLSTM-CRF, which is also implemented on GPUs, by 13.7%, 32.8% and 48.8%, respectively. We can see that for the dataset with a longer average length, the speed of inference will be more advantageous. Because the model calculates uncertainties through parallel sampling the same input multiple times, the inference time of the BiLSTM-UANet ($M = 8$) only slightly increases.

To further investigate the influence of the different sentence lengths, we analyze the inference speed of the UANet and CRF on the CoNLL2003 development set, which is split into five parts according to sentence length. We rule out the influence of the text encoder and only counted the time of label decoding. The left subfigure in Figure 5 shows the decoding speed on the different sentence lengths. The results reveal that as the sentence length increases, the speed of the UANet is relatively stable, while the speed of the CRF decreases substantially. Due to the UANet's parallelism, the words in an input sentence can be processed in parallel, which means every word can be labeled concurrently by taking advantage of parallel computations on GPUs, while the CRF decoding algorithm processes one word after another in a left-to-right fashion. When processing the sentence longer than 30, the UANet is nearly 3 times faster than the CRF. In addition, we exhibit the F1 score of the sentences with different lengths in right subfigure. It is worth noting that the UANet outperforms the CRF by a large margin when the length of the sentence is greater than 15, verifying the UANet's superiority in long-term label dependencies.

We also investigate the training and inference speeds of document-level models, and probe whether the performance can be improved through modeling the document-level label consistency. The results are shown in Table VI. In term of speed, Doc-BiLSTM-UANet outperforms baseline models by a large margin. In term of advantage on co-occurrence entities, For the baseline models, they use CRF as decoder. In contrast, our method performs attention over document-level label embeddings derived from the memory network to explicitly model the co-occurrence relationship. Hence, our model can achieve better performance on the co-occurrence entity recognition.

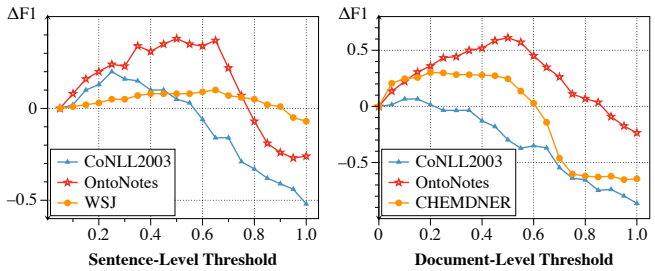


Fig. 6: F1 variation under different uncertainty thresholds. The results are evaluated on the development sets. ΔF1 represents the F1 scores at different steps minus the initial results.

D. Discussion

In this section, we study the influence of four important hyper-parameters (i.e. uncertainty threshold, memory subset size, and number of sampling) and then present a case study to show how our model refines the wrong draft labels.

Uncertainty Threshold. In order to investigate the influence of uncertainty threshold Γ, we evaluate the performance with different uncertainty thresholds on the four datasets, as shown in Figure 6. We can see that as the threshold increases, the sentence-level and document-level curves show the same trend. $\Gamma = 0$ represents that the model uses all of the refined labels as final predictions. As the threshold gets larger, the performance of UANet can improve by reducing the negative effects on correct draft labels. However, when Γ is too large, the model mainly uses draft labels as final predictions, resulting in performance degradation, which verifies our motivation that a reasonable uncertainty threshold can avoid side effects on correct draft labels.

As compared with sentence-level experiments, we find that on the CoNLL2003 dataset, the peak of $\Delta F1$ curve in Figure 6 shifted to the left in document-level experiment, while there is no significant change on OntoNotes dataset. The possible reason is that when we inject document-level knowledge, the ability of the refinement stage becomes stronger, while the ability of first stage is basically unchanged, which leads to more preference to use the output of the refinement stage when setting the threshold (i.e. corresponding a small threshold). After adding document-level knowledge to OntoNotes dataset, the performance improvement is relatively small, so the threshold curve does not change much.

Number of Sampling. We also investigate the influence of the number of sampling in the variational LSTM as shown in Figure 7. The results meet our expectation that a larger number of sampling can lead to better performance because a larger number of sampling can make the model better approximate the posterior $p(\mathbf{W}|\mathcal{D})$.

Text	...	striker	Viorel	Ion	of	Otelul	Galati	and	defender	Liviu	Ciobotariu	of	National	Bucharest	...	...	University	of	Yangon	...
BiLSTM-CRF	...	O	B-PER	E-PER	O	B-PER	E-PER	O	O	B-PER	E-PER	O	B-LOC	E-LOC	...	...	O	O	S-LOC	...
Draft Label	...	O	B-PER	E-PER	O	B-PER	E-PER	O	O	B-PER	E-PER	O	B-ORG	E-ORG	...	...	B-ORG	I-ORG	E-LOC	...
Refinement	...	O	B-PER	E-PER	O	B-ORG	E-ORG	O	O	B-PER	E-PER	O	B-ORG	E-ORG	...	...	B-LOC	I-ORG	E-ORG	...
Uncertainty	...	0.001	0.005	0.047	0.004	0.532	0.605	0.000	0.000	0.001	0.014	0.001	0.818	0.927	...	...	0.302	0.816	0.800	...
Final Prediction	...	O	B-PER	E-PER	O	B-ORG	E-ORG	O	O	B-PER	E-PER	O	B-ORG	E-ORG	...	...	B-ORG	I-ORG	E-ORG	...

TABLE VII: NER cases analysis. The first example shows the necessity of long-range dependencies: UANet can learn the label consistency of two phrases `Otelul Galati` and `Liviu Ciobotariu` are connected by the coordinating conjunctions word `and`. The second example shows the effectiveness of the uncertainty threshold in mitigating the side effect of incorrect refinement: the uncertainty of the word `University` is under the uncertainty threshold, so it will not be refined. Contents with blue and red colors represent correct and incorrect entities, respectively. Draft labels with uncertainty greater than 0.35 will be refined.

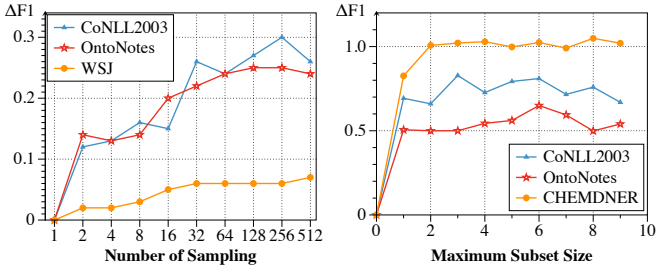


Fig. 7: F_1 variation with respect to the number of sampling in variational LSTM and maximum size of queried subset m. The results are evaluated on the development sets. ΔF_1 represents the F_1 scores at different steps minus the initial results.

Memory Size The right part of figure 7 illustrates the performance of our model with respect to the maximum size of queried subset m for each word. The maximum size 10 is derived from the observation that 97.47% of the tokens belonging to an entity appeared less than 10 times in an document for the development set of CoNLL2003, with 91.65% for OntoNotes and 81.78% for CHEMDNER, respectively. We find that incorporating the first co-occurrence key-value information of an document can provide a performance improvement of at least 0.5% for all the three datasets. Note that different from [21], we don't randomly select memories for a unique word. We let the memories in the original order as they appear in the document, since we believe that the contextual information for the place where an entity first appears in an document should be more adequate, and would be more useful in the future predictions. We further calculate the average number of co-occurrence tokens of each token belonging to an entity. Regardless of the current token, the statistics are 1.46, 1.72, and 2.36 for CoNLL2003, OntoNotes and CHEMDNER, respectively. It's reasonable that a larger group size can bring more improvements for CHEMDNER dataset due to a larger average number of reference tokens.

Case Study. Table VII shows two cases from CoNLL2003 NER dataset. The first case reflects the necessity of modeling higher-order dependencies in the NER task. UANet can learn the label consistency of two phrases near the word `and`. Moreover, seq2seq decoding model [11] refines the labels in a left-to-right way and can't refine the previous labels in this case. The second case shows the effectiveness of the uncertainty threshold in mitigating the side effect of incorrect refinement. In this case, the refinement model is affected by the incorrect label of `Yangon` (*E-LOC*) when predicting the word `University`. Since the uncertainty value of `University` is lower than the threshold, our model can get the correct results.

VI. Conclusions

In this study, we introduce a novel two-stage sequence labeling framework that incorporates Bayesian neural networks to estimate the uncertainty in the model's predictions. We found that such uncertainty can effectively indicate the labels with a high probability of being wrong. The proposed method can selectively refine the uncertain labels to mitigate the side effect of the refinement on correct labels. In addition, the proposed model can capture the label dependencies and word-label interactions with different ranges in parallel by taking advantage of GPUs for a faster prediction. We also show that this framework can be easily extended to the document-level sequence labeling cases. The experimental results on both the sentence-level and document-level tasks across four sequence labeling datasets demonstrated that the proposed method significantly outperform the previous methods.

References

[1] L. Ratinov and D. Roth, "Design challenges and misconceptions in named entity recognition," in Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009), 2009, pp. 147–155.

[2] A. Passos, V. Kumar, and A. McCallum, "Lexicon infused phrase embeddings for named entity resolution," in Proceedings of the Eighteenth Conference on Computational Natural Language Learning, 2014, pp. 78–86.

[3] G. Luo, X. Huang, C.-Y. Lin, and Z. Nie, "Joint entity recognition and disambiguation," in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 879–888.

[4] X. Ma and E. Hovy, "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), 2016, pp. 1064–1074.

[5] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in Proceedings of NAACL-HLT, 2016, pp. 260–270.

[6] E. Strubell, P. Verga, D. Belanger, and A. McCallum, "Fast and accurate entity recognition with iterated dilated convolutions," in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 2670–2680.

[7] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in Proceedings of NAACL-HLT, 2018, pp. 2227–2237.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "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), 2019, pp. 4171–4186.

[9] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," Journal of machine learning research, vol. 12, no. Aug, pp. 2493–2537, 2011.

[10] Z. Ye and Z.-H. Ling, "Hybrid semi-markov crf for neural sequence labeling," in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2018, pp. 235–240.

[11] Y. Zhang, H. Chen, Y. Zhao, Q. Liu, and D. Yin, "Learning tag dependencies for sequence tagging." in IJCAI, 2018, pp. 4581–4587.

[12] M. Popel, D. Mareček, J. Štěpánek, D. Zeman, and Z. Žabokrtskỳ, "Coordination structures in dependency treebanks," in Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2013, pp. 517–527.

[13] J. D. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in Proceedings of the Eighteenth International Conference on Machine Learning, 2001, pp. 282–289.

[14] J. Zhou and W. Xu, "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), 2015, pp. 1127–1137.

[15] V. Krishnan and C. D. Manning, "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 the 44th annual meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2006, pp. 1121–1128.

[16] L. Cui and Y. Zhang, "Hierarchically-refined label attention network for sequence labeling," in EMNLP-IJCNLP, 2019.

[17] A. Vaswani, Y. Bisk, K. Sagae, and R. Musa, "Supertagging with lstms," in NAACL-HIT, 2016, pp. 232–237.

[18] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in Advances in Neural Information Processing Systems, 2015, pp. 1171–1179.

[19] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in NeurIPS, 2017, pp. 5574–5584.

[20] Y. Qian, E. Santus, Z. Jin, J. Guo, and R. Barzilay, "Graphie: A graph-based framework for information extraction," in NAACL-HIT, 2019, pp. 751–761.

[21] Y. Luo, F. Xiao, and H. Zhao, "Hierarchical contextualized representation for named entity recognition," in AAAI, 2020.

[22] A. Hu, Z. Dou, J.-Y. Nie, and J.-R. Wen, "Leveraging multi-token entities in document-level named entity recognition," in AAAI, 2019.

[23] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," in EMNLP, 2016, pp. 1400–1409.

[24] N. V. Cuong, N. Ye, W. S. Lee, and H. L. Chieu, "Conditional random field with high-order dependencies for sequence labeling and segmentation," The Journal of Machine Learning Research, vol. 15, no. 1, pp. 981–1009, 2014.

[25] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," Transactions of the Association for Computational Linguistics, vol. 4, pp. 357–370, 2016.

[26] C. D. Santos and B. Zadrozny, "Learning character-level representations for part-of-speech tagging," in Proceedings of the 31st International Conference on Machine Learning (ICML-14), 2014, pp. 1818–1826.

[27] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[28] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," arXiv preprint arXiv:1508.01991, 2015.

[29] H. Yan, B. Deng, X. Li, and X. Qiu, "Tener: Adapting transformer encoder for name entity recognition," arXiv preprint arXiv:1911.04474, 2019.

[30] X. Li, H. Yan, X. Qiu, and X. Huang, "Flat: Chinese ner using flat-lattice transformer," arXiv preprint arXiv:2004.11795, 2020.

[31] Y. Xiao and W. Y. Wang, "Quantifying uncertainties in natural language processing tasks," in Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI 2019). AAAI, 2019.

[32] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in Advances in neural information processing systems, 2016, pp. 1019–1027.

[33] J. S. Denker and Y. LeCun, "Transforming neural-net output levels to probability distributions," in Advances in neural information processing systems, 1991, pp. 853–859.

[34] W. L. Buntine and A. S. Weigend, "Bayesian back-propagation," Complex systems, vol. 5, no. 6, pp. 603–643, 1991.

[35] D. J. MacKay, "A practical bayesian framework for backpropagation networks," Neural computation, vol. 4, no. 3, pp. 448–472, 1992.

[36] R. M. Neal, Bayesian learning for neural networks. Springer Science & Business Media, 2012, vol. 118.

[37] Y. Gal, "Uncertainty in Deep Learning," p. 174, 2016, zSCC: 0001676.

[38] M. G. Andrey Malinin, "Predictive uncertainty estimation via prior networks," arXiv preprint arXiv:1802.10501, 2018.

[39] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in ICML, 2016, pp. 1050–1059.

[40] C. E. Rasmussen, "Gaussian processes in machine learning," in Summer School on Machine Learning. Springer, 2003, pp. 63–71.

[41] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with bernoulli approximate variational inference," arXiv preprint arXiv:1506.02158, 2015.

[42] L. Shen, A. Sarkar, and F. J. Och, "Discriminative reranking for machine translation," in Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004, 2004, pp. 177–184.

[43] L. Huang and D. Chiang, "Forest rescoring: Faster decoding with integrated language models," in Proceedings of the 45th annual meeting of the association of computational linguistics, 2007, pp. 144–151.

[44] F. T. Yingce Xia, J. L. Lijun Wu, N. Y. Tao Qin, and T.-Y. Liu, "Deliberation networks: Sequence generation beyond one-pass decoding," in Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, Long Beach, CA, USA, dec 2017, pp. 1782—1792.

[45] J. Lee, E. Mansimov, and K. Cho, "Deterministic non-autoregressive neural sequence modeling by iterative refinement," arXiv preprint arXiv:1802.06901, 2018.

[46] M. Collins and T. Koo, "Discriminative reranking for natural language parsing," Computational Linguistics, vol. 31, no. 1, pp. 25–70, 2005.

[47] P. Yin and G. Neubig, "Reranking for neural semantic parsing," in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019.

[48] S. Min, V. Zhong, L. Zettlemoyer, and H. Hajishirzi, "Multi-hop reading comprehension through question decomposition and rescoring," arXiv preprint arXiv:1906.02916, 2019.

[49] S. Wang, M. Yu, J. Jiang, W. Zhang, X. Guo, S. Chang, Z. Wang, T. Klinger, G. Tesauro, and M. Campbell, "Evidence aggregation for answer re-ranking in open-domain question answering," arXiv preprint arXiv:1711.05116, 2017.

[50] S. Goldfarb-Tarrant, T. Chakrabarty, R. Weischedel, and N. Peng, "Content planning for neural story generation with aristotelian rescoring," arXiv preprint arXiv:2009.09870, 2020.

[51] M. Collins, "Ranking algorithms for named entity extraction: Boosting and the votedperceptron," in Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 2002, pp. 489–496.

[52] J. Yang, Y. Zhang, and F. Dong, "Neural reranking for named entity recognition," arXiv preprint arXiv:1707.05127, 2017.

[53] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in NeurIPS, 2017, pp. 5998–6008.

[55] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in ACL, 2019.

[56] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," Machine learning, vol. 37, no. 2, pp. 183–233, 1999.

[57] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," Computational Linguistics, vol. 19, no. 2, pp. 313–330, 1993. [Online]. Available: https://www.aclweb.org/anthology/J93-2004

[58] M. Collins, "Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms," in Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002). Association for

Computational Linguistics, Jul. 2002, pp. 1–8. [Online]. Available: https://www.aclweb.org/anthology/W02-1001

[59] C. D. Manning, "Part-of-speech tagging from 97% to 100%: Is it time for some linguistics?" in Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I, ser. CICLing'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 171–189. [Online]. Available: http://dl.acm.org/citation.cfm?id=1964799.1964816

[60] E. F. Tjong Kim Sang and F. De Meulder, "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, 2003, pp. 142–147.

[61] R. Weischedel, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue, A. Taylor, J. Kaufman, M. Franchini et al., "Ontonotes release 5.0 ldc2013t19," Linguistic Data Consortium, Philadelphia, PA, vol. 23, 2013.

[62] M. Krallinger, F. Leitner, O. Rabal, M. Vazquez, J. Oyarzabal, and A. Valencia, "Chemdner: The drugs and chemical names extraction challenge," Journal of cheminformatics, vol. 7, no. 1, p. S1, 2015.

[63] Y. Xin, E. Hart, V. Mahajan, and J.-D. Ruvini, "Learning better internal structure of words for sequence labeling," in EMNLP, 2018.

[64] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in NeurIPS, 2013.

[65] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[66] J. Yang and Y. Zhang, "NCRF++: An open-source neural sequence labeling toolkit," in Proceedings of ACL 2018, System Demonstrations. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 74–79. [Online]. Available: https://www.aclweb.org/anthology/P18-4013

[67] L. Liu, J. Shang, X. Ren, F. F. Xu, H. Gui, J. Peng, and J. Han, "Empower sequence labeling with task-aware neural language model," in Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[68] H. Chen, Z. Lin, G. Ding, J.-G. Lou, Y. Zhang, and B. F. Karlsson, "Grn: Gated relation network to enhance convolutional neural network for named entity recognition," in AAAI, 2019.

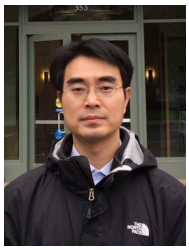
[69] J. Yang, S. Liang, and Y. Zhang, "Design challenges and misconceptions in neural sequence labeling," in Proceedings of the 27th International Conference on Computational Linguistics. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 3879–3889. [Online]. Available: https://www.aclweb.org/anthology/C18-1327

[70] T. Stanislawek, A. Wróblewska, A. Wójcicka, D. Ziembicki, and P. Biecek, "Named entity recognition-is there a glass ceiling?" in Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL), 2019, pp. 624–633.

[71] Z. Zhang, B. Tang, Z. Li, and H. Zhao, "Modeling named entity embedding distribution into hypersphere," arXiv preprint arXiv:1909.01065, 2019.

Jiacheng Ye received his bachelor degree in information management and information systems from Sun Yat-sen University. He is a master student at Fudan University. His research interests include natural language processing and information extraction.

Xiang Zhou is a post-graduate student at the School of Computer Science, Fudan University under the supervision of Professor Xiaoqing Zheng. He received his bachelor degree in chemistry from Fudan University. His research interests include Natural Language Processing and Machine Learning.



Xiaoqing Zheng is an Associate Professor of the School of Computer Science, Fudan University, Shanghai, China. He received his Ph.D. degree in Computer Science from Zhejiang University in 2007. He has been doing research on Semantic Data Integration during his visit at the Information Technology Group, Massachusetts Institute of Technology (MIT) as an International Faculty Fellow. He also visited the natural language processing and machine learning groups, University of California, Los Angeles (UCLA) as visiting researcher from 2019 to 2020. His research interests include Natural Language Processing, Machine Learning, and Semantic Web.

Tao Gui received his bachelor degree in computer science from National University of Defense Technology. He is a Ph.D. student at Fudan University. His research interests include natural language processing and information extraction.

Qi Zhang received the Ph.D. degree in computer science from Fudan University. He is a professor of computer science at Fudan University, Shanghai, China. His research interests include natural language processing and information retrieval.