SEGMENTING NATURAL LANGUAGE SENTENCES VIA LEXICAL UNIT ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this work, we present **L**exical **U**nit **A**nalysis (**LUA**), a framework for general sequence segmentation tasks. Given a natural language sentence, LUA scores all the valid segmentation candidates and utilizes dynamic programming (DP) to extract the maximum scoring one. LUA enjoys a number of appealing properties such as inherently guaranteeing the predicted segmentation to be valid and facilitating globally optimal training and inference. Besides, the practical time complexity of LUA can be reduced to linear time, which is very efficient. We have conducted extensive experiments on 5 tasks, including syntactic chunking, named entity recognition (NER), slot filling, Chinese word segmentation, and Chinese part-of-speech (POS) tagging, across 15 datasets. Our models have achieved the state-of-the-art performances on 13 of them. The results also show that the F1 score of identifying long-length segments is notably improved.

1 INTRODUCTION

Sequence segmentation is essentially the process of partitioning a sequence of fine-grained lexical units into a sequence of coarse-grained ones. In some scenarios, each composed unit is assigned a categorical label. For example, Chinese word segmentation splits a character sequence into a word sequence (Xue, 2003). Syntactic chunking segments a word sequence into a sequence of labeled groups of words (i.e., constituents) (Sang & Buchholz, 2000).

Currently, there are two mainstream approaches for sequence segmentation. The most common is to regard it as a sequence labeling problem by using IOB tagging scheme (Mesnil et al., 2014; Ma & Hovy, 2016; Liu et al., 2019a; Chen et al., 2019a; Luo et al., 2020). The representative work is Bidirectional LSTM-CRF (Huang et al., 2015), which adopts LSTM (Hochreiter & Schmidhuber, 1997) to read the input sentence and CRF (Lafferty et al., 2001) to decode the label sequence. This type of method is very effective, providing tons of state-of-the-art results. However, it is vulnerable to producing invalid segments, for instance, a segment starting with I-tag. This problem becomes more severe in low resource settings (Peng et al., 2017).

Recently, there is a growing interest in span-based models (Zhai et al., 2017; Li et al., 2019; Yu et al., 2020). They treat the span rather than the token as the basic unit for labeling. Li et al. (2019) cast named entity recognition (NER) to a machine reading comprehension (MRC) task, where entities are extracted as retrieving answer spans. Yu et al. (2020) rank all the spans in terms of the scores predicted by a bi-affine model (Dozat & Manning, 2016). In NER, span-based models have significantly outperformed their sequence labeling based counterparts. While these methods circumvent the use of IOB tagging scheme, they mostly rely on post-processing rules to guarantee the extracted span set to be valid. Moreover, since span-based models are locally normalized at span level, they potentially suffer from the label bias problem (Lafferty et al., 2001).

This paper seeks to provide a new framework which infers the segmentation of a unit sequence by directly selecting from all valid segmentation candidates, instead of manipulating tokens or spans. To this end, we propose **L**exical **U**nit **A**nalysis (**LUA**) in this paper. LUA assigns a score to every valid segmentation candidate and leverages dynamic programming (DP) (Bellman, 1966) to search for the maximum scoring one. The score of a segmentation is computed by using the scores of its all segments. Besides, we adopt neural networks to score each segment of an input sentence. The purpose of using DP is to solve the intractability of extracting the maximum scoring segmentation

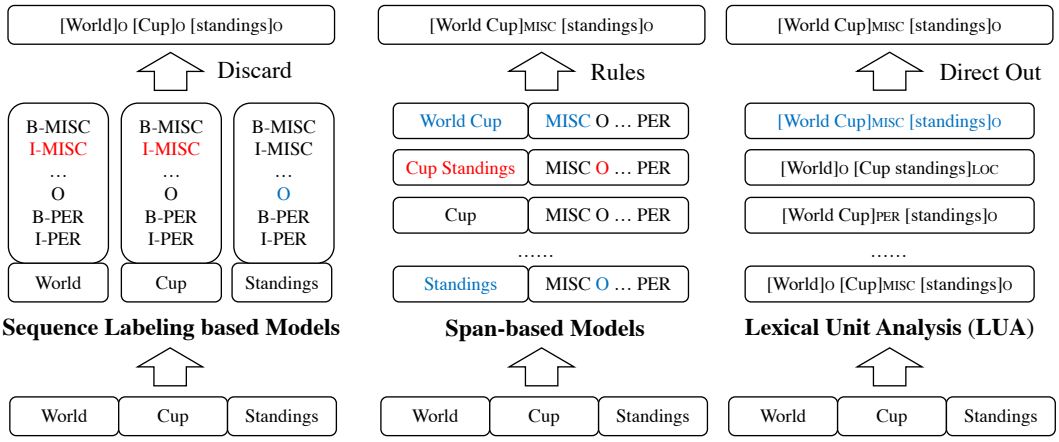

Figure 1: A toy example to show LUA and how it differs from prior methods. The items in blue and red respectively denote valid and invalid predictions.

candidate by brute-force search. The theoretical time complexity of LUA is quadratic time. By performing parallel matrix computations, it can be optimized to linear time, which is very efficient. For training criterion, we induce a hinge loss between the ground truth and the predicted segmentation. We also optimize LUA in terms of capturing label correlation. While sacrificing little running time, it further improves the performances on some tasks.

Figure 1 illustrates the comparison between previous methods and the proposed LUA. Prior models at token level and span level are vulnerable to invalid predictions, and hence rely on heuristic rules to fix them. LUA scores all possible segmentation candidates and uses DP to extract the maximum scoring one. In this way, our models inherently guarantee the predictions to be valid. Moreover, the globality of DP addresses the label bias problem.

Extensive experiments are conducted on syntactic chunking, NER, slot filling, Chinese word segmentation, and Chinese part-of-speech (POS) tagging across 15 tasks. We have obtained new state-of-the-art results on 13 of them and performed competitively on the others. In particular, we observe that LUA is expert at identifying long-length segments.

2 Methodology

We denote an input sequence (i.e., fine-grained lexical units) as $\mathbf{x} = [x_1, x_2, \cdots, x_n]$, where n is the sequence length. An output sequence (i.e., coarse-grained lexical units) is represented as the segmentation $\mathbf{y} = [y_1, y_2, \cdots, y_m]$ with each segment y_k being a triple (i_k, j_k, t_k). m denotes its length. (i_k, j_k) specifies a span that corresponds to the phrase $\mathbf{x}_{i_k,j_k} = [x_{i_k}, x_{i_k+1}, \cdots, x_{j_k}]$. t_k is a label from the label space $\mathcal{L}$. We define a valid segmentation candidate as its segments are non-overlapping and fully cover the input sequence.

A case extracted from CoNLL-2003 dataset (Sang & De Meulder, 2003):

$$\begin{array}{l} \mathbf{x} = [[\text{SOS}], \text{Sangthai}, \text{Glory}, 22/11/96, 3000, \text{Singapore}] \\ \mathbf{y} = [(1, 1, \text{O}), (2, 3, \text{MISC}), (4, 4, \text{O}), (5, 5, \text{O}), (6, 6, \text{LOC})] \end{array} .$$

Start-of-sentence symbol [SOS] is added in the pre-processing stage.

2.1 Model: Scoring Segmentation Candidates

We denote $\mathcal{Y}$ as the universal set that contains all valid segmentation candidates. Given one of its members $\mathbf{y} \in \mathcal{Y}$, we compute the score $f(\mathbf{y})$ as

$$f(\mathbf{y}) = \sum_{(i,j,t) \in \mathbf{y}} \left(s^c_{i,j} + s^l_{i,j,t} \right), \tag{1}$$

Algorithm 1: Inference via Dynamic Programming (DP)

Input: Composition score $s^c_{i,j}$ and label score $s^l_{i,j,t}$ for all possible segments (i, j, t).
Output: The maximum segmentation scoring candidate $\hat{\mathbf{y}}$ and its score $f(\hat{\mathbf{y}})$.

Set two $n \times n$ shaped matrices, $\mathbf{c}^L$ and $\mathbf{b}^c$, for computing maximum scoring labels.
Set two n-length vectors, $\mathbf{g}$ and $\mathbf{b}^g$, for computing maximum scoring segmentation.
for $1 \leq i \leq j \leq n$ **do**
    Compute the maximum label score for each span (i, j): $c^L_{i,j} = \max_{t \in \mathcal{L}} c^l_{i,j,t}$.
    Record the backtracking index: $b^c_{i,j} = \arg\max_{t \in \mathcal{L}} c^l_{i,j,t}$.
Initialize the value of the base case $\mathbf{x}_{1,1}$: $g_1 = s^c_{1,1} + s^L_{1,1}$.
for $i \in [2, 3, \cdots, n]$ **do**
    Compute the value of the prefix $\mathbf{x}_{1,i}$: $g_i = \max_{1 \leq j \leq i-1} \left(g_{i-j} + (s^c_{i-j+1,i} + s^L_{i-j+1,i})\right)$.
    Record the backtracking index: $b^g_i = \arg\max_{1 \leq j \leq i-1} \left(g_{i-j} + (s^c_{i-j+1,i} + s^L_{i-j+1,i})\right)$.
Get the maximum scoring candidate $\hat{\mathbf{y}}$ by back tracing the tables $\mathbf{b}^g$ and $\mathbf{b}^c$.
Get the maximum segmentation score: $f(\hat{\mathbf{y}}) = g_n$.

where $s^c_{i,j}$ is the composition score to estimate the feasibility of merging the fine-grained units $[x_i, x_{i+1}, \cdots, x_j]$ into a coarse-grained unit and $s^l_{i,j,t}$ is the label score to measure how likely the label of this segment is t. Both scores are obtained by a scoring model.

Scoring Model a scoring model scores all the possible segments (i, j, t) of an input sentence $\mathbf{x}$. Firstly, we get the representation of each fine-grained unit. Following prior works (Li et al., 2019; Luo et al., 2020; Yu et al., 2020), we adopt BERT (Devlin et al., 2018), a powerful pre-trained language model, as the sentence encoder. Specifically, we have

$$[\mathbf{h}^w_1, \mathbf{h}^w_2 \cdots, \mathbf{h}^w_n] = \mathrm{BERT}(\mathbf{x}), \tag{2}$$

Then, we compute the representation of a coarse-grained unit $\mathbf{x}_{i,j}, 1 \leq i \leq j \leq n$ as

$$\mathbf{h}^p_{i,j} = \mathbf{h}^w_i \oplus \mathbf{h}^w_j \oplus (\mathbf{h}^w_i - \mathbf{h}^w_j) \oplus (\mathbf{h}^w_i \odot \mathbf{h}^w_j), \tag{3}$$

where $\oplus$ is vector concatenation and $\odot$ is element-wise product.

Eventually, we employ two non-linear feedforward networks to score a segment (i, j, t):

$$\begin{cases} s^c_{i,j} = \left(\mathbf{v}^c\right)^T \tanh(\mathbf{W}^c \mathbf{h}^p_{i,j}) \\ s^l_{i,j,t} = \left(\mathbf{v}^l_t\right)^T \tanh(\mathbf{W}^l \mathbf{h}^p_{i,j}) \end{cases}, \tag{4}$$

where $\mathbf{v}^c$, $\mathbf{W}^c$, $\mathbf{v}^l_t, t \in \mathcal{L}$, and $\mathbf{W}^l$ are all learnable parameters. Besides, the scoring model used here can be flexibly replaced by any regression model.

2.2 Inference via Dynamic Programming

The prediction of the maximum scoring segmentation candidate can be formulated as

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{y}). \tag{5}$$

Because the size of search space $|\mathcal{Y}|$ increases exponentially with respect to the sequence length n, brute-force search to solve Equation 5 is computationally infeasible. LUA uses DP to address this issue, which is facilitated by the decomposable nature of Equation 1.

DP is a well-known optimization method which solves a complicated problem by breaking it down into simpler sub-problems in a recursive manner. The relation between the value of the larger problem and the values of its sub-problems is called the Bellman equation.

Sub-problem In the context of LUA, the sub-problem of segmenting an input unit sequence $\mathbf{x}$ is segmenting its prefixes $\mathbf{x}_{1,i}, 1 \leq i \leq n$. We define g_i as the maximum segmentation score of the prefix $\mathbf{x}_{1,i}$. Under this scheme, we have $\max_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{y}) = g_n$.

The Bellman Equation The relatinship between segmenting a sequence $\mathbf{x}_{1,i}, i > 1$ and segmenting its prefixes $x_{1,i-j}, 1 \leq j \leq i-1$ is built by the last segments $(i-j+1, i, t)$:

$$g_i = \max_{1 \leq j \leq i-1} \big(g_{i-j} + (s^c_{i-j+1,i} + \max_{t \in \mathcal{L}} s^l_{i-j+1,i,t})\big). \tag{6}$$

In practice, to reduce the time complexity of above equation, the last term is computed beforehand as $s^L_{i,j} = \max_{t \in \mathcal{L}} s^l_{i,j,t}, 1 \leq i \leq j \leq n$. Hence Equation 6 is reformulated as

$$g_i = \max_{1 \leq j \leq i-1} \big(g_{i-j} + (s^c_{i-j+1,i} + s^L_{i-j+1,i})\big). \tag{7}$$

The base case is the first token $\mathbf{x}_{1,1} = [[\text{SOS}]]$. We get its score g_1 as $s^c_{1,1} + s^L_{1,1}$.

Algorithm 1 shows how DP is applied in inference. Firstly, we set two matrices and two vectors to store the solutions to the sub-problems (1-st to 2-nd lines). Secondly, we get the maximum label scores for all the spans (3-rd to 5-th lines). Then, we initialize the trivial case g_1 and recursively calculate the values for prefixes $\mathbf{x}_{1,i}, i > 1$ (6-th to 9-th lines). Finally, we get the predicted segmentation $\hat{\mathbf{y}}$ and its score $f(\hat{\mathbf{y}})$ (10-th to 11-th lines).

The time complexity of Algorithm 1 is $\mathcal{O}(n^2)$. Since the $\max$ operation of Equation 7 is performed in parallel on GPU, it can be optimized to only $\mathcal{O}(n)$, which is highly efficient. Besides, DP, as the backbone of the proposed model, is non-parametric. The trainable parameters only exist in the scoring model part. These show LUA is a very light-weight algorithm.

2.3 Training Criterion

We adopt max-margin penalty as the loss function for training. Given the predicted segmentation $\hat{\mathbf{y}}$ and the ground truth segmentation $\mathbf{y}^*$, we have

$$\mathcal{J} = \max\big(0, 1 - f(\mathbf{y}^*) + f(\hat{\mathbf{y}})\big). \tag{8}$$

3 Extension to Unlabeled Sequence Segmentation

In some tasks (e.g., Chinese word segmentation), the segments are unlabeled. We denote this type of a segment as $y_k = (i_k, j_k)$. The Equation 1 and Equation 7 are also reformulated as

$$\begin{cases} f(\mathbf{y}) = \sum\limits_{(i,j) \in \mathbf{y}} s^c_{i,j} \\ g_i = \max\limits_{1 \leq j \leq i-1} (g_{i-j} + s^c_{i-j+1,i}) \end{cases}. \tag{9}$$

4 Extension to Capturing Label Correlation

In some tasks, such as Chinese POS tagging, the labels of successive segments are strongly correlated. To incorporate this type of information, we redefine $f(\mathbf{y})$ as

$$f(\mathbf{y}) = \sum_{1 \leq k \leq m} \Big(s^c_{i_k,j_k} + s^l_{i_k,j_k,t_k}\Big) + \sum_{2 \leq k \leq m} s^d_{t_{k-1},t_k}. \tag{10}$$

Score $s^d_{t_{k-1},t_k}$ models the label dependency between two successive segments, y_{k-1} and y_k. In practice, we parameterize a learnable matrix $\mathbf{W}^d \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ to implement it.

The corresponding Bellman equation to above scoring function is

$$g_{i,t} = \max_{1 \leq j \leq i-1} \big(\max_{t' \in \mathcal{L}}(g_{i-j,t'} + s^d_{t',t}) + (s^c_{i-j+1,i} + s^l_{i-j+1,i,t})\big), \tag{11}$$

where $g_{i,t}$ is the maximum score of labeling the last segment of the prefix $\mathbf{x}_{1,i}$ with t. For initialization, we set the value of $g^d_{1,\text{O}}$ as 0 and the others as $-\infty$. By performing the inner loops of two $\max$ operations in parallel, the practical time complexity for computing $g_{i,t}, 1 \leq i \leq n, t \in \mathcal{L}$ is also $\mathcal{O}(n)$. Ultimately, the segmentation score $f(\hat{\mathbf{y}})$ is obtained by $\max_{t \in \mathcal{L}} g_{n,t}$.

This extension further improves the results on syntactic chunking and Chinese POS tagging, as both tasks have rich sequential features among the labels of the segments.

Model		Chunking	NER	
		CoNLL-2000	CoNLL-2003	OntoNotes 5.0
Bi-LSTM + CRF (Huang et al., 2015)		94.46	90.10	-
Flair Embeddings (Akbik et al., 2018)		96.72	93.09	89.3
GCDT w/ BERT (Liu et al., 2019a)		96.81	93.23	-
BERT-MRC (Li et al., 2019)		-	93.04	91.11
HCR w/ BERT (Luo et al., 2020)		-	93.37	90.30
BERT-Biaffine Model (Yu et al., 2020)		-	**93.5**	91.3
This Work	LUA	96.95	93.46	**92.09**
	LUA w/ Label Correlation	**97.23**	-	-

Table 1: Experiment results on syntactic chunking and NER.

5 EXPERIMENTS

We have conducted extensive experiments on 5 tasks, including syntactic chunking, NER, slot filling, Chinese word segmentation, Chinese POS tagging, across 15 datasets. Firstly, our models have achieved new state-of-the-art performances on 13 of them. Secondly, the results demonstrate that the F1 score of identifying long-length segments has been notably improved. Finally, we show that LUA is a very efficient algorithm concerning the running time.

5.1 SETTINGS

We use the same configurations for all 15 datasets. The hidden dimension of scoring model is 300. L2 regularization and dropout ratio are respectively set as 1×10^{-6} and 0.2 for reducing overfit. We use Adam (Kingma & Ba, 2014) to optimize our model. Following prior works, $\mathrm{BERT}_{\mathrm{BASE}}$ is adopted as the sentence encoder. We adopt uncased $\mathrm{BERT}_{\mathrm{BASE}}$ for slot filling, Chinese $\mathrm{BERT}_{\mathrm{BASE}}$ for Chinese tasks (e.g., Chinese POS tagging), and cased $\mathrm{BERT}_{\mathrm{BASE}}$ for others (e.g., syntactic chunking). We tokenize all the complete tokens into the sub-word pieces (Devlin et al., 2018) and expand the corresponding spans. In addition, the improvements of our models over all baselines are statistically significant with $p < 0.05$ under t-test.

5.2 SYNTACTIC CHUNKING AND NER

Syntactic chunking identifies and labels the constituents of an input word sequence. We use CoNLL-2000 dataset (Sang & Buchholz, 2000), which defines 11 syntactic chunk types (NP, VP, PP, etc.). Standard data includes train set and test set. NER locates and classifies the named entities mentioned in unstructured text into predefined categories. We use CoNLL-2003 dataset (Sang & De Meulder, 2003) and OntoNotes 5.0 dataset (Pradhan et al., 2013). CoNLL-2003 dataset consists of 22137 sentences totally and is split into 14987, 3466, and 3684 sentences for the training set, development set, and test set, respectively. It's tagged with four linguistic entity types (PER, LOC, ORG, MISC). OntoNotes 5.0 dataset contains 76714 sentences from a wide variety of sources (e.g., magazine and newswire). It includes 18 types of named entity, which consists of 11 types (Person, Organization, etc.) and 7 values (Date, Percent, etc.). We follow the same format and partition of above datasets as in (Li et al., 2019; Luo et al., 2020; Yu et al., 2020). At test time, we convert the predicted segments into IOB format and utilize conlleval script[1] to compute the F1 score.

Table 1 shows the experiment results. We adopt the results of all baselines from (Akbik et al., 2018; Li et al., 2019; Luo et al., 2020; Yu et al., 2020). Besides, following (Luo et al., 2020), we rerun the source code[2] of GCDT and report its performance on CoNLL-2000 using standard evaluation method. On CoNLL-2000 and Ontonotes 5.0, our models have significantly outperformed previous methods and obtained state-of-the-art performances. We improve the F1 scores by 0.43% on CoNLL-2000 and 0.87% on Ontonotes 5.0. Compared with the strong baseline, Flair Embedding, the increasements are 0.53% and 3.12%. All these results confirm the great effectiveness of LUA. Besides, incorporating the label correlation contributes to 0.29% improvement on CoNLL-2000,

[1] https://www.clips.uantwerpen.be/conll2000/chunking/conlleval.txt.

[2] https://github.com/Adaxry/GCDT.

Model		ATIS	SNIPS	MTOD
Slot-Gated SLU (Goo et al., 2018)		95.20	88.30	95.12
Bi-LSTM + EMLo (Siddhant et al., 2019)		95.42	93.90	-
Joint BERT (Chen et al., 2019b)		96.10	97.00	96.48
CM-net (Liu et al., 2019b)		96.20	97.15	-
This Work	LUA	96.15	97.10	97.53
	LUA w/ Intent Detection	**96.27**	**97.20**	**97.55**

Table 2: Experiment results on the three datasets of slot filling.

Model	AS	MSR	CITYU	PKU	CTB6
Rich Pretraining (Yang et al., 2017)	95.7	97.5	96.9	96.3	96.2
Bi-LSTM (Ma et al., 2018)	96.2	98.1	97.2	96.1	96.7
Multi-Criteria Learning + BERT (Huang et al., 2019)	96.6	97.9	97.6	96.6	97.6
BERT (Meng et al., 2019)	96.5	98.1	97.6	96.5	-
Glyce + BERT (Meng et al., 2019)	96.7	**98.3**	97.9	96.7	-
Unlabeled LUA	**96.94**	98.27	**98.21**	**96.88**	**98.13**

Table 3: Experiment results on Chinese word segmentation.

which verifies the idea proposed in Section 4. On CoNLL-2003, we achieve competitive performances. The F1 score of LUA is lower than Biaffine Model by only 0.04%.

5.3 Slot Filling

Slot filling, as a crucial task in spoken language understanding (SLU), extracts semantic constituents from an utterance. We use ATIS dataset (Hemphill et al., 1990), SNIPS dataset (Coucke et al., 2018), and MTOD dataset (Schuster et al., 2018). ATIS dataset consists of audio recordings of people making flight reservations. The training set contains 4478 utterances and the test set contains 893 utterances. SNIPS dataset is collected by Snips personal voice assistant. The training set contains 13084 utterances and the test set contains 700 utterances. MTOD dataset has three domains, including Alarm, Reminder, and Weather. We use the English part of MTOD dataset, where training set, dev set, and test set respectively contain 30521, 4181, and 8621 utterances. We follow the same partition of above datasets as in (Goo et al., 2018; Schuster et al., 2018).

Table 2 shows the experiment results. For ATIS and SNIPS, we follow the results of all baselines as reported in (Liu et al., 2019b). For MTOD, we rerun the open source toolkit Slot-gated SLU[3] and Joint BERT[4]. Previous methods all joint model slot filling and intent detection (a classification task of SLU). For a fair comparison, we also report the results (last row) of using the hidden representation of [CLS] to predict the intent of an input utterance $\mathbf{x}$. On the one hand, our models have surpassed prior all approaches and obtained the state-of-the-art results on the three datasets. We increase the F1 scores by 0.07% on ATIS, 0.05% on SNIPS, and 1.11% on MTOD. On the other hand, compared with the strong baseline (i.e., Joint BERT), LUA achieves the improvements of 0.18% and 0.21% on ATIS and SNIPS without even modeling intent detection. All above results strongly verify the great effectiveness of LUA.

5.4 Chinese Word Segmentation

Chinese word segmentation divides a Chinese character sequence into successive words. We use SIGHAN 2005 bake-off (Emerson, 2005) and Chinese Treebank 6.0 (CTB6) (Xue et al., 2005). SIGHAN 2005 back-off consists of 5 datasets: AS, MSR, CITYU, and PKU. Following (Ma et al., 2018), we randomly select 10% training data as development set. We convert all digits, punctuation, and Latin letters to half-width for handling full/half-width mismatch between training and test set. We also convert AS and CITYU to simplified Chinese. For CTB6, we follow the same format and partition as in (Yang et al., 2017; Ma et al., 2018).

[3]https://github.com/MiuLab/SlotGated-SLU.

[4]https://github.com/monologg/JointBERT.

Model		CTB5	CTB6	CTB9	UD1
Bi-RNN + CRF (Single) (Shao et al., 2017)		94.07	90.81	91.89	89.41
Bi-RNN + CRF (Ensemble) (Shao et al., 2017)		94.38	-	92.34	89.75
Lattice-LSTM (Meng et al., 2019)		95.14	91.43	92.13	90.09
Glyce + Lattice-LSTM (Meng et al., 2019)		95.61	91.92	92.38	90.87
BERT (Meng et al., 2019)		96.06	94.77	92.29	94.79
Glyce + BERT (Meng et al., 2019)		96.61	95.41	93.15	96.14
This Work	LUA	96.79	95.39	93.22	96.01
	LUA w/ Label Correlation	**97.96**	**96.63**	**93.95**	**97.08**

Table 4: Experiment results on the four datasets of Chinese POS tagging.

Model	$1-3$ (8695)	$4-7$ (2380)	$8-11$ (151)	$12-24$ (31)	Overall
HCR w/ BERT	91.15	85.22	50.43	20.67	90.27
BERT-Biaffine Model	91.67	87.23	70.24	40.55	91.26
LUA	**92.31**	**88.52**	**77.34**	**57.27**	**92.09**

Table 5: The study is conducted on OntoNotes 5.0 dataset.

Table 3 demonstrates the experiment results. We adopt the performances of all baselines from (Yang et al., 2017; Ma et al., 2018; Huang et al., 2019; Meng et al., 2019). We have achieved new state-of-the-art performances on all the datasets, except for MSR. Our model improves the F1 score by 0.25% on AS, 0.32% on CITYU, 0.19% on PKU, and 0.54% on CTB6. Note that our model doesn't use any external resource, such as glyph information (Meng et al., 2019) or POS tags (Yang et al., 2017). On MSR, we are slightly lower than Glyce + BERT by 0.03%.

5.5 Chinese POS Tagging

Chinese POS tagging assigns a POS tag to every segmented word of a character sequence. We use Chinese Treebank 5.0 (CTB5), CTB6, Chinese Treebank 9.0 (CTB9) (Xue et al., 2005), and the Chinese section of Universal Dependencies 1.4 (UD1) (Nivre et al., 2016). CTB5 is comprised of newswire data. CTB9 consists of source texts in various genres, which cover CTB5. we convert the texts in UD1 from traditional Chinese into simplified Chinese. We follow the same train/dev/test split of above datasets as in (Yang et al., 2017).

Table 4 shows the experiment results. We follow the performances of all baselines reported in (Meng et al., 2019). Our models have obtained new state-of-the-art results on all the datasets. We increase the F1 scores by 1.40% on CTB5, 1.28% on CTB6, 0.86% on CTB9, and 0.98% on UD1. Besides, integrating the label dependency contributes to the improvements of 1.21%, 1.30%, 0.78%, and 1.11%. Single LUA also outperforms Glyce + BERT by 0.19% on CTB5 and 0.08% on CTB9. All these results further verify the effectiveness of LUA and its variant.

5.6 Long-length Segment Identification

Intuitively, the proposed LUA should be more accurate in recognizing long-length segments than sequence labeling based methods. To verify it, we measure the F1 scores of the segments of different lengths on OntoNotes 5.0. LUA is compared to a best sequence labeling based model (i.e., HCR) and a best span-based model (i.e., Biaffine Model). We reproduce the results of baselines by using open source codes, biaffine-ner[5] and Hire-NER[6].

Table 5 demonstrates the experiment results. The column names denote the segment length range and the total number in test corpus. On the one hand, both LUA and Biaffine Model obtain much higher scores of identifying long-length entities than HCR. For example, LUA increases F1 score of $12-24$ range by almost twofold. These results verify our intuition. On the other hand, LUA achieves much better results than Biaffine Model. It improves the performances by 10.11% on $8-11$ range and 41.23% on $12-24$ range. We attribute this to the optimality of DP.

[5]https://github.com/juntaoy/biaffine-ner.

[6]https://github.com/cslydia/Hire-NER.

Model	Theoretical Complexity	Practical Complexity	Running Time
BERT	$\mathcal{O}(n)$	$\mathcal{O}(1)$	5m11s
BERT + CRF	$\mathcal{O}(n\|\mathcal{L}\|^2)$	$\mathcal{O}(n)$	7m33s
LUA	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	7m2s
LUA w/ Label Correlation	$\mathcal{O}(n^2\|\mathcal{L}\|^2)$	$\mathcal{O}(n)$	8m15s

Table 6: Running time comparison on CoNLL-2000 dataset.

5.7 Running Time Analysis

Table 6 shows the running time of different models. The middle columns are the time complexity of decoding a label sequence. The last column records the time cost of one epoch in training. We set batch size as 16 and run models on 1 GPU. The results indicate that LUA is very fast. For example, LUA is only 1m51s slower than BERT (Devlin et al., 2018).

6 Related Work

Tasks in sequence segmentation aim to partition a fine-grained unit sequence into multiple labeled coarse-grained units. Traditionally, there are two types of approaches. The most common is to cast it into a token-level sequence labeling task (Mesnil et al., 2014; Ma & Hovy, 2016; Chen et al., 2019a) by using IOB tagging scheme. Every word in the sentence is labeled with B-tag if it's the beginning of a segment, I-tag if it's inside but not the first one within the segment, or O otherwise. Although the segments are indirectly extracted, this method is very effective, providing a number of state-of-the-art results. For example, Akbik et al. (2018) present Flair Embeddings that pretrain character embedding in a large corpus and directly use it, instead of word representation, to encode a sentence. Liu et al. (2019a) introduce GCDT that deepens the state transition path at each position in a sentence, and further assigns each word with global representation. Luo et al. (2020) use hierarchical contextualized representations to incorporate both sentence-level and document-level information. Nevertheless, these models are vulnerable to producing invalid segments, for instance, a segment starting with I-tag. This problem becomes more severe in low resource settings (Peng et al., 2017). Moreover, in Section 5.6, we observe that it performs much worse on identifying long-length segments than the proposed LUA.

An alternative approach that is less studied uses a transition-based system to incrementally segment and label input sequence (Zhang et al., 2016; Lample et al., 2016). For instance, Qian et al. (2015) present a transition-based model for joint word segmentation, POS tagging, and text normalization. Wang et al. (2017) apply a transition-based model to disfluency detection task, which helps capture non-local chunk-level features. These models have many advantages like theoretically lower time complexity and directly recognizing the segments. However, to our best knowledge, no recent transition-based model surpasses its sequence labeling based counterparts.

More recently, there is a surge of interest in span-based models. They treat the segment, instead of the word, as the basic unit for labeling. For example, Li et al. (2019) regard NER as an MRC task, where entities are recognized as retrieving answer spans. Since these methods are locally normalized at span level, they rely on rules to ensure the extracted span set to be valid and severely suffer from the label bias problem. Span-based methods also emerge in other fields of NLP. In constituent parsing, Stern et al. (2017) integrate the LSTM-minus feature (Wang et al., 2017) into parsing models. In coreference resolution, (Lee et al., 2018) consider all spans in the text as the potential mentions and learn distributions over possible antecedents.

7 Conclusion

This work presents a novel LUA for general sequence segmentation tasks. LUA directly scores all the valid segmentation candidates and uses dynamic programming to extract the maximum scoring one. Compared with previous models, LUA naturally guarantees the predicted segmentation to be valid and circumvents the label bias problem. Extensive experiments are conducted on 5 tasks across 15 datasets. We have achieved the state-of-the-art performances on 13 of them. Importantly, the F1 score of identifying long-length segments is significantly improved.

REFERENCES

Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1638–1649, 2018.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

Hui Chen, Zijia Lin, Guiguang Ding, Jianguang Lou, Yusen Zhang, and Borje Karlsson. Grn: Gated relation network to enhance convolutional neural network for named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 6236–6243, 2019a.

Qian Chen, Zhu Zhuo, and Wen Wang. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*, 2019b.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*, 2018.

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

Timothy Dozat and Christopher D Manning. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*, 2016.

Thomas Emerson. The second international chinese word segmentation bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, 2005.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 753–757, 2018.

Charles T Hemphill, John J Godfrey, and George R Doddington. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990.

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

Weipeng Huang, Xingyi Cheng, Kunlong Chen, Taifeng Wang, and Wei Chu. Toward fast and accurate neural chinese word segmentation with multi-criteria learning. *arXiv preprint arXiv:1903.04190*, 2019.

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

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

Aviral Kumar and Sunita Sarawagi. Calibration of encoder decoder models for neural machine translation. *arXiv preprint arXiv:1903.00802*, 2019.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

Phong Le and Willem Zuidema. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 729–739, 2014.

Kenton Lee, Luheng He, and Luke Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference. *arXiv preprint arXiv:1804.05392*, 2018.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. A unified mrc framework for named entity recognition. *arXiv preprint arXiv:1910.11476*, 2019.

Yijin Liu, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. Gcdt: A global context enhanced deep transition architecture for sequence labeling. *arXiv preprint arXiv:1906.02437*, 2019a.

Yijin Liu, Fandong Meng, Jinchao Zhang, Jie Zhou, Yufeng Chen, and Jinan Xu. Cm-net: A novel collaborative memory network for spoken language understanding. *arXiv preprint arXiv:1909.06937*, 2019b.

Ying Luo, Fengshun Xiao, and Hai Zhao. Hierarchical contextualized representation for named entity recognition. In *AAAI*, pp. 8441–8448, 2020.

Ji Ma, Kuzman Ganchev, and David Weiss. State-of-the-art chinese word segmentation with bi-lstms. *arXiv preprint arXiv:1808.06511*, 2018.

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

Yuxian Meng, Wei Wu, Fei Wang, Xiaoya Li, Ping Nie, Fan Yin, Muyu Li, Qinghong Han, Xiaofei Sun, and Jiwei Li. Glyce: Glyph-vectors for chinese character representations. In *Advances in Neural Information Processing Systems*, pp. 2746–2757, 2019.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539, 2014.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 1659–1666, 2016.

Nanyun Peng et al. *Jointly Learning Representations for Low-Resource Information Extraction*. PhD thesis, Ph. D. thesis, Johns Hopkins University, 2017.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pp. 143–152, 2013.

Tao Qian, Yue Zhang, Meishan Zhang, Yafeng Ren, and Donghong Ji. A transition-based model for joint segmentation, pos-tagging and normalization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1837–1846, 2015.

Erik F Sang and Sabine Buchholz. Introduction to the conll-2000 shared task: Chunking. *arXiv preprint cs/0009008*, 2000.

Erik F Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*, 2003.

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. Cross-lingual transfer learning for multilingual task oriented dialog. *arXiv preprint arXiv:1810.13327*, 2018.

Yan Shao, Christian Hardmeier, Jörg Tiedemann, and Joakim Nivre. Character-based joint segmentation and pos tagging for chinese using bidirectional rnn-crf. *arXiv preprint arXiv:1704.01314*, 2017.

Aditya Siddhant, Anuj Goyal, and Angeliki Metallinou. Unsupervised transfer learning for spoken language understanding in intelligent agents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 4959–4966, 2019.

Mitchell Stern, Jacob Andreas, and Dan Klein. A minimal span-based neural constituency parser. *arXiv preprint arXiv:1705.03919*, 2017.

Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang, and Ting Liu. Transition-based disfluency detection using lstms. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2785–2794, 2017.

Wenhui Wang and Baobao Chang. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2306–2315, 2016.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207, 2005.

Nianwen Xue. Chinese word segmentation as character tagging. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 8, Number 1, February 2003: Special Issue on Word Formation and Chinese Language Processing*, pp. 29–48, 2003.

Jie Yang, Yue Zhang, and Fei Dong. Neural word segmentation with rich pretraining. *arXiv preprint arXiv:1704.08960*, 2017.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. Named entity recognition as dependency parsing. *arXiv preprint arXiv:2005.07150*, 2020.

Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. Neural models for sequence chunking. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Meishan Zhang, Yue Zhang, and Guohong Fu. Transition-based neural word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 421–431, 2016.