

Using BiLSTM in Dependency Parsing for Vietnamese

Luong Nguyen Thi, Linh Ha My, Huyen Nguyen Thi Minh, Phuong Le-Hong

Dalat University, Lamdong, Vietnam, Email:luongnt@dlu.edu.vn
VNU University of Science, Hanoi, Vietnam, Email: halinh.hus@gmail.com,
huyenntm@vnu.edu.vn, phuonglh@vnu.edu.vn

Abstract. Recently, deep learning methods have achieved good results in dependency parsing for many natural languages. In this paper, we investigate the use of bidirectional long short-term memory network models for both transition-based and graph-based dependency parsing for the Vietnamese language. We also report our contribution in building a Vietnamese dependency treebank whose tagset conforms to the Universal Dependency schema. Various experiments demonstrate the efficiency of this method, which achieves the best parsing accuracy in comparison to other existing approaches on the same corpus, with unlabeled attachment score of 84.45% or labeled attachment score of 78.56%.

1 Introduction

Dependency parsing consists of graph-based and transition-based parser (Kubler et al.,2009). Given sentence s , a graph-based algorithm finds the highest scoring parse tree from all possible outputs while a transition-based algorithm builds a parse by a sequence of actions. In recent years, many researchers have developed deep learning approaches with high accuracy in English, Chinese,etc. Chen and Manning proposed a novel way of learning a neural network classifier in a greedy, transition-based dependency parser which achieved USA=92.2% and LSA=89.7% on the English Penn Treebank [1]. Dyer et al. (2015) [2] also presented stack LSTMs, recurrent neural networks for sequences, with push and pop operations, and used them to implement a state-of-the-art of transition-based dependency parser with USA=93.2% and LSA=90.9% in English. Kiperwasser et al. (2016) [3] presented a simple and effective scheme for dependency parsing based on bidirectional-LSTMs (BiLSTMs) which had USA=93.8% and LSA=91.5% for English. Besides, Dozat and Manning (2016) [4] have recently inherited from Kiperwasser et al. using neural attention in a simple graph-based dependency parser. Their parser gained a state-of- the-art or its performance on standard treebanks in six different languages, achieving 95.7% UAS and 94.1% LAS on the most popular English PTB dataset.

Regarding Vietnamese dependency parsing, there have been many contributions to parsing. In 2008, Nguyễn Lê Minh et al. [5] used MST parser on a corpus consisting of 450 sentences. Then, in 2012, Phuong Le et al.[6] applied a

lexicalized tree-adjoining grammar parser trained on a subset of the Vietnamese treebank. In 2013, Thi-Luong et al. [7] used MaltParser on a Vietnamese dependency treebank which is converted automatically from a Vietnamese treebank. One year later, Dat et al. [8] also presented a new conversion method to automatically transform a constituent-based Vietnamese Treebank into dependency trees. In 2015, Phuong Le et al. [9] improved accuracy of Vietnamese dependency parsing, used distributed word representations with Skip-gram and GloVe model for transition-based dependency parsing. In 2016, Thi-Luong et al. [10] also used distributed word representations with Skip-gram in graph-based dependency parsing for Vietnamese and Dat et al. [11] presented an empirical study for Vietnamese dependency parsing. In 2017, Kiem Hieu [12] presented their work on building BKTreebank, a dependency treebank for Vietnamese.

1.1 Transition-based dependency parsing

The transition system has a set of configurations and a set of transitions which are applied to configurating. By parsing a sentence, the system is initialized to an initial configuration based on the input sentence, and transitions are repeatedly applied to this configuration. After a finite number of transitions, the system arrives at a terminal configuration, and a parse tree is read off the terminal configuration. In a greedy parser, a classifier is used to choose the transition and take in each configuration, based on features extracted from the configuration itself. The parsing algorithm is presented in Algorithm 1 below.

Algorithm 1 Greedy transision-based parsing

Require: Sentence $s = w_1, w_2, \dots, x_w, t + 1, \dots, t_n$.
parameterized function $SCORE_\theta(\cdot)$ with parameters θ
Ensure: Tree of s
 $c \leftarrow INITIAL(s)$
while not $TERMINAL(c)$ **do**
 $t \leftarrow \arg \max_{t \in LEGAL(c)} SCORE_\theta(\Phi(c), t)$
 $c \leftarrow t$
end while
return $tree(c)$

Many transition-based systems [13] are popular such as arg-eager algorithm, arg-standard algorithm. However in this work, we employ the arc-hybrid system which is similar to these. In the arc-hybrid system, a configuration $c = (\alpha, \beta, T)$ consists of a stack α , a buffer β , and a set T of dependency arcs. Both the stack and the buffer hold integer indices pointing to sentence elements. Given a sentence $s = w_1, w_2, \dots, w_n$, the system is initialized with an empty stack, an empty arc set, and $\beta = 1, \dots, n, ROOT$, where $ROOT$ is the special root index. Any configuration c with an empty stack and a buffer containing only $ROOT$ is terminal, and the parse tree is given by the arc set T_c of c . The arc-hybrid system allows 3 possible transitions, SHIFT, LEFT and RIGHT, defined as:

- $SHIFT[(\alpha, b_0|\beta, T)] = (\alpha|b_0, \beta, T)$
- $LEFT_i[(\alpha|s_1|s_0, b_0|\beta, T)] = (\alpha|s_1, b_0|\beta, T \cup \{(b_0, s_0, l)\})$
- $RIGHT_i[(\alpha|s_1|s_0, \beta, T)] = (\alpha|s_1, \beta, T \cup \{(s_1, s_0, l)\})$

1.2 Graph-based dependency parsing

The second approach is the graph-based dependency parsing algorithm introduced by McDonald et al. [14]. In this algorithm, the weights of the edges are calculated for building dependency graphs of a sentence as follows:

$$s(i, j) = w \cdot f(i, j)$$

where w is the weight of the (i, j) edge, $f(i, j)$ is feature of (i, j) edge. The weight of (i, j) edge represents the ability to create a dependency between the head (w_i) and the dependence (w_j). If the arc score function is known, then the weight of graph is:

$$S(G = (V, E)) = \sum_{(i,j)} s(i, j)$$

Then, based on the weights of all edges in graph, McDonald et al. [15] showed that this problem is equivalent to finding the highest scoring directed spanning tree for the graph G originating out of the root node 0.

1.3 Long short-term memory

Recurrent Neural Network The recurrent neural network (RNN) is a class of artificial neural network designed for sequence labeling task. It takes input as a sequence of vector and returns another sequence. The simple architecture of RNN has an input layer x , hidden layer h and output layer y . At each time step t , the values of each layer are computed as follows:

$$\begin{aligned} h_t &= f_h(W_{ih}x_t + W_{hh}h_{t-1}) \\ y_t &= f_o(W_{ho}h_t) \end{aligned}$$

where W_{ih} , W_{hh} and W_{ho} are the three connection weight matrices and f_h and f_o that are sigmoid and softmax are the hidden and output unit activation functions.

Long Short-Term Memory Long Short-Term Memory (LSTM) was first proposed in 1997 by Sepp Hochreiter et al. [16]. LSTM is an extended model of RNN which is designed to combat with these vanishing and exploding gradient problems when learning with long-range sequences. LSTM networks are the same as RNN, except that the hidden layer updates are replaced by memory cells. Figure 1 shows a LSTM cell, including i, f, o are the input, forget and output gates, respectively. c and \tilde{c} denote the memory cell content. LSTM cell calculates a hidden state s_t as following equations: where σ is the element-wise sigmoid function and \odot is the element-wise product, i, f, o and c are the input gate, forget gate, output gate, and cell vector respectively. U^i, U^f, U^c, U^o are connection weight matrices between input x and gates, and W^i, W^f, W^c, W^o

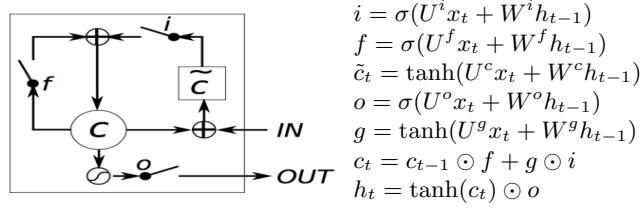


Fig. 1. Long Short-Term Memory cell

are connection weight matrices between gates and hidden state h .

Bidirectional Long Short-Term Memory The original LSTM uses only previous contexts for prediction. For many sequence labeling tasks, it is advisable to take the contexts from two directions. Bidirectional LSTM utilizes both the previous and future context by processing the sequence in two directions, and generate two independent sequences of LSTM output vectors.

2 Approach

2.1 Universal dependency parsing in Vietnamese

Universal dependency The dependency label represents the dependence between the two words in the sentence. Each pair of words, in different positions, will have a different dependency label. There is a general conversion rule to do the dependency label which is uniform throughout the language. There are many sets of relational labels for a language which are different from each others.

The *Universal dependencies - UD*¹ was developed by the Stanford University team, *Marneffe et al.* [17]. This is a project developed based on the treebank annotation for multi-language, with the goal of facilitating the development of multilingual parsing, cross-language learning, research and analysis from the perspective of the type of language. This project was developed based on the *Stanford Dependency - SD* dependency labels, also by the Stanford University team (*Marneffe et al.*, 2015) based on multi-lingual labels (*Petrov et al.*, 2012) and the magnetic word form (*Zeman*, 2008).

The general objective of developing Universal dependencies is to provide a labels set and guidelines to facilitate the construction of similar works for other languages and allow expansion to a new language. The labels in SD are organized in groups of subject, object, clauses, word definitions, or nouns. Stanford offers nearly 50 types of English dependencies based-on PennTreebank corpus. All of these dependencies are twofold: between a head word and its dependent word. Each relation is given by three components: dependency label, head word and dependent word.

Universal dependencies can be applied to many different languages, which can be used to suggest improvements in dependency parsing, even for English.

¹ <http://universaldependencies.org/guidelines.html>

This research team has developed a core label set that has been extensively tested in a variety of languages, meaning that this core label set can be applied in many different languages. It is also possible to add new labels as needed by categorizing special linguistic relationships, or for individual cases of one or more groups of languages. This label set may correspond to many different languages such as English, French, German, Chinese This label is useful because it can indicate a dependency for the same sentence, in different languages.

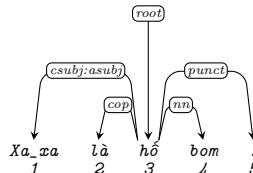
Universal dependencies contain 40 labels that were organized to allow principles of the UD taxonomy such that rows correspond to functional categories in relation to the head (core arguments of clausal predicates, non-core dependents of clausal predicates, and dependents of nominals) while the columns correspond to structural categories of the dependent (nominals, clauses, modifier words, function words) as in table 1. All of Universal dependencies are defined and there are specific examples that can use to develop and build a complete label for the others language.

Table 1. Dependencies in universal Stanford Dependencies

	<i>Nominals</i>	<i>Clauses</i>	<i>Modifier words</i>	<i>Function Words</i>
<i>Core arguments</i>	<i>nsubj</i> <i>obj</i> <i>iobj</i>	<i>csubj</i> <i>ccomp</i> <i>xcomp</i>		
<i>Non-core arguments</i>	<i>nsubj</i> <i>obl</i> <i>vocative</i> <i>expl</i> <i>dislocated</i>	<i>csubj</i> <i>advcl</i>	<i>advmod</i> <i>discourse</i>	<i>aux</i>
<i>Coordination</i>	<i>MWE</i> <i>conj</i> <i>cc</i>	<i>Loose</i> <i>list</i> <i>parataxis</i>	<i>Special</i> <i>orphan</i> <i>goeswith</i> <i>reparandum</i>	<i>Other</i> <i>punct</i> <i>root</i> <i>dep</i>
	<i>compound</i>			

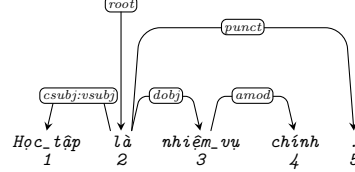
Vietnamese dependencies Based-on universal dependencies and Viettreebank, we has built a Vietnamese dependencies. This set has labels that coincide with the labels in the UD and several new labels. The Vietnamese dependencies set has 46 labels. Some of the dependent labels that we have designed specifically for Vietnamese:

- *csubj:asubj* (adjective subject: A adjective subject is an adjective phrase which is the syntactic subject of a clause. In Vietnamese, the subject is usually a noun (or a noun phrase), but there are some cases adjectives be the subject.
 - *Xa_xa là hổ bom.*



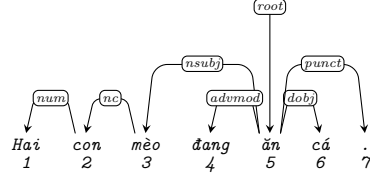
- *csubj:vsubj* (*verb subject*): This is used to describe the phenomenon as a verb is a subject of a sentence. In Vietnamese, the subject is usually a noun, but there are some cases adjective, verb, clause can do the subject of a sentence.

- *Học tập là nhiệm vụ chính* → *csubj:vsubj*(là, học tập)



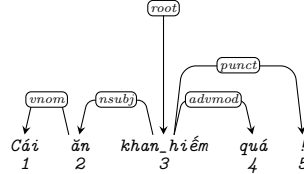
- *nc* (*classifier noun*): This relation represents the relationship between a classifier noun with common nouns. The classifier noun always stands before the common noun, for example, “cái”, “con”, ...

- *Hai con mèo đen đang ăn cá.* → *nc*(mèo, con)



- *vnom* (*verb nominal*): This is used for the relationship between a verb nominal and a classifier noun. The classifier noun is always before the verb. Example: “cái”, “sự”, “việc”, ...

- *Cái ăn khan hiếm quá!* → *vnom*(ăn, cái)



Then, we have a comparison between the two sets of labels under Table 2.

2.2 BiLSTM in dependency parsing

Using BiLSTM feature representation Instead of using direct feature vectors in dependency parsing, we use the same method in [3]. Each of feature vectors by its BiLSTM encoding, and uses a concatenation of a minimal set of such BiLSTM encodings as a feature function, which is then passed to a non-linear scoring function (multi-layer perceptron).

Give input sentence s with n words: w_1, \dots, w_n and the corresponding POS tags p_1, \dots, p_n . Each word w_i and POS p_i with embedding vectors $e(w_i)$ and $e(p_i)$ and denote $x_{1:n}$ is a sequence of input vectors with:

$$x_i = e(w_i) \circ e(p_i)$$

The embedding are trained together with the model. We also denoted v_i is the output of this model. v_i is computed as follows:

Table 2. Comparison between Vietnamese dependencies (VD) and Universal dependencies (UD)

VD (2016)	UD (2015)	Meaning
csubj	csubj	Clausal subject
csubj:asubj		
csubj:vsubj		
acomp	xcomp	Adjectival complement
amod	amod	Adjectival modier
apredmod	advmod	Adjectival modier of a predicate
advmod	advmod	Adverbial modier
advcl	advcl	Adverbial clause modier
aux	aux	Auxiliary
auxpass	auxpass	Passive auxiliary
appos	appos	Appositional modier
cc	cc	Coordination
ccomp	ccomp	Clausal complement
conj	conj	Conjunct
cop	cop	Copula
dep	dep	Dependent
det	det	Determiner
discourse	discourse	Discourse element
dislocated	dislocated	Dislocated elements
doobj	doobj	Direct object
foreign	foreign	Foreign words
iobj	iobj	Indirect object
list	list	List
mark	mark	Marker
neg	neg	Negation modier
nn	compound	Noun compound modier
nsubj	nsubj	Nominal subject
num	nummod	Numeric modier
number	compound	Element of compound number
parataxis	parataxis	Parataxis
pcomp	mark	Prepositional complement
pobj	case	Object of a preposition
prep	nmod	Prepositional modier
punct	punct	Punctuation
remnant	remnant	Remnant in ellipsis
reparandum	reparandum	Overridden disfluency
rcmod	acl:relcl	Relative clause modier
ref	ref	Referent
root	root	root
tmod	nmod:tmod	Temporal modier
vcomp	ccomp	Verb complement of a verb
vmod	amod:vmod	Verb modier of an NP
vocative	vocative	Vocative
xcomp	xcomp	Open clausal complement
nsubjpass	nsubjpass	Passive nominal subject
csubjpass	csubjpass	Clausal passive subject
-	expl	Expletive
-	goeswith	Goes with
nc	-	Classifier noun
vnom	-	Verb nominal

$$v_i = BiLSTM(x_{1:n}, i)$$

A Bidirectional LSTM composed of two LSTMs: $LSTM_f$ and $LSTM_b$. The $LSTM_f$ reads the sequence in its regular order and the $LSTM_b$ reads it in reverse. Concretely, given a sequence of vectors $x_{1:n}$ and index i , the function $BiLSTM_\theta(x_{1:n}, i)$ is defined as:

$$\begin{aligned} BiLSTM_\theta(x_{1:n}, i) &= LSTM_f(x_{1:i}) \circ LSTM_b(x_{n:i}) \\ v_i &= BiLSTM_\theta(x_{1:n}, i) \end{aligned}$$

The feature function ϕ is then the concatenation of a small number of BiLSTM vectors. The resulting feature vectors are then scored using a non-linear function, namely a multi-layer perceptron with one hidden layer (MLP):

$$MLP_\theta(x) = W^2 \cdot \tanh(W^1 \cdot x + b^1) + b^2$$

where $\theta = W^2, W^1, b^2, b^1$ are the model parameters.

Transition-Based dependency parsing uses BiLSTM feature representation Given a sentence s , the transition-based parser is initialized with configuration c . Then, a feature function $\phi(c)$ represents the configuration c as a vector. The feature function is the concatenated BiLSTM vectors of the some items on the stack and the buffer. For example, for a configuration $c = (...|s_2|s_1|s_0, b_0|..., T)$ the feature extractor is the top 3 items on the stack and the first item on the buffer. It is defined as:

$$\begin{aligned} \phi(c) &= v_{s_2} \circ v_{s_1} \circ v_{s_0} \circ v_{b_0} \\ v_i &= BiLSTM(x_{1:n}, i) \end{aligned}$$

Each transition is scoring using an MLP that is fed the BiLSTM encodings of vectors that are gotten from the feature extractor. Each x_i is concatenation of a word and a POS vector. SCORE assigning scores to (configuration, transition) pairs. SCORE scores the possible transition $t = Shift, Left_Arc, Right_Arc$, and the highest scoring transition \hat{t} is chosen. The transition \hat{t} is applied to the configuration that will output a new configuration.

Graph-based dependency parsing uses BiLSTM feature representation

In graph-based parsing, the weights of the edges are calculated for building dependency graphs of $s = x_{1:n}$ a sentence as follows:

$$\begin{aligned} predict(s) &= [\arg \max_{y \in Y(s)} score_{global}(s, y)] \\ score_{global}(s, y) &= \sum_{part \in y} score_{local}(s, part) \end{aligned}$$

where space $Y(s)$ of valid dependency trees over s .

Arc-factored parsing decomposes the score of a tree to the sum of the score of its head-modifier arcs (h, m) :

$$parse(s) = [\arg \max_{y \in Y(s)} \sum_{(h, m) \in y} score(\phi(s, h, m))]$$

where $\phi(s, h, m)$ is the feature extractor which uses the BiLSTM encoding of the head word and the modifier word: $\phi(s, h, m) = BiLSTM(x_{1:n}, h) \circ BiLSTM(x_{1:n}, m)$. The final model is:

$$\begin{aligned} parse(s) &= \arg \max_{y \in Y(s)} \sum_{(h,m) \in y} score(\phi(s, h, m)) \\ &= \arg \max_{y \in Y(s)} \sum_{(h,m) \in y} MLP(v_h \circ v_m) \\ v_i &= BiLSTM(x_{1:n}, i) \end{aligned}$$

3 Experiments

3.1 Datasets

We use the similar database in our research [7], [9], [10].

Text corpus for distributed word representations: To create distributed word representations, we use the dataset consisting of 7.3GB of text from 2 million articles collected via the Vietnamese news portal. The text is first normalized to lower case. All special characters are removed except these common symbols: the comma, the semi-colon, the colon, the full stop and the percentage sign. All numeral sequences are replaced with the special token <number>, so those correlations between a certain word and a number are correctly recognized by the neural network or the log-bilinear regression model.

Each word in the Vietnamese language may consist of more than one syllable with spaces in between, which could be regarded as multiple words by the unsupervised models. Hence it is necessary to replace the spaces within each word with underscores to create full word tokens. The tokenization process follows the method described in [18]. After removal of special characters and tokenization, the articles add up to 969 million word tokens, spanning a vocabulary of 1.5 million unique tokens. We train the unsupervised models with the full vocabulary to obtain the representation vectors, and then prune the collection of word vectors to the 5,000 most frequent words, excluding special symbols and the token <number> representing numeral sequences.

Dependency treebank. We conduct our experiments on the Vietnamese dependency treebank dataset. This treebank is derived automatically from the constituency-based annotation of the VTB [7], containing 10,471 sentences (225,085 tokens). We manually check the correctness of the conversion on a subset of the converted corpus to come up 3,000 of universal dependency with a training set of 2,200 sentences, a test set of 400 sentences and a dev set of 400 sentences.

3.2 Feature sets

Feature sets in transition-based: For each parser configuration $c = (...|s_2|s_1|s_0, b_0|..., T)$ and transition $f(c)$ in the gold parse. $\phi(c)$ is the feature vector representation if the parser configuration c . We denoted part-of-speech tags of token w is $p(w)$.

We use the notation $tk(w)$ and $e(w)$ to denote the extracting the word and the distributed representation of the word of token w . $rm(w)$ and $lm(w)$ corresponding to the right-most and left-most modifier of token w . We used the feature templates for the classifier in table 3. Each feature $v_{tk}(w) = p(w) \circ tk(w)$ or $v_e = p(w) \circ e(w)$ is a feature template of token w .

Table 3. Feature sets for use in the transition classifier

Feature set	Feature templates
ϕ_0	$v_{tk}(s_0), v_{tk}(s_1), v_{tk}(s_2), v_{tk}(b_0)$
ϕ_1	$v_e(s_0), v_e(s_1), v_e(s_2), v_e(b_0)$
ϕ_2	$\phi_0, v_{tk}(rm(s_0)), v_{tk}(lm(s_0)), v_{tk}(rm(s_1)),$ $v_{tk}(lm(s_1)), v_{tk}(rm(s_2)), v_{tk}(lm(s_2)), v_{tk}(lm(b_0))$
ϕ_3	$\phi_1, v_e(rm(s_0)), v_e(lm(s_0)), v_e(rm(s_1)), v_e(lm(s_1)),$ $v_e(rm(s_2)), v_e(lm(s_2)), v_e(lm(b_0))$

Feature sets in graph-based: The feature-set proposed by McDonald et al. (2005) with 18 templates for a first-order parser, while the first order feature extractor in the actual implementation’s code (MSTParser²) includes roughly a hundred feature templates. In this case, feature extractor uses merely encoding of the headword and the modifier word with pos and word.

3.3 Vietnamese dependency parsing based-on bist-parser

The Bist-parser is a tool, using BiLSTM feature extractors with graph-based and transition-based dependency parsers. This tool was developed by Kiperwasser et al., using BiLSTM feature extractors in Section 2.2.

We use two attachment scores, labeled attachment score (LAS) and unlabelled attachment score (UAS) to evaluate the accuracy of the dependency parsing system. Attachment scores are defined as the percentage of correct dependency relations recovered by the parser. A dependency relation is considered correct if both the source word and the target word are correct (UAS), plus the dependency type is correct (LAS).

We also estimate on the Vietnamese dependency treebank [7]. The result is the highest accuracy in Vietnamese dependency parsing as presenting in table 5.

4 Conclusion

In this paper, we presented in detail to contribute Vietnamese universal dependency. We also use this data in the Bist-parser system which is based on bidirectional LSTMs for dependency parser. We evaluated the accuracy of the system

² <http://www.seas.upenn.edu/~strctlm/MSTParser/MSTParser.html>

Table 4. Accuracy of Bist-parser with feature sets on the Vietnamese universal dependency treebank

Feature set	System	Test	
		USA	LSA
ϕ_2	Transition-based	76.86%	72.38%
	Graph-based	77.79%	74.08%
ϕ_3	Transition-based	75.75%	71.13%
	Graph-based	78.17%	74.84%
Phuong et al. [9]	Transition-based	73.21%	63.06%
Luong et al. [10]	Graph-based	73.09%	68.32%

Table 5. Accuracy of Bist-parser with feature sets on Vietnamese dependency treebank [7]

Feature set	System	Test	
		USA	LSA
ϕ_2	Transition-based	82.77%	76.02%
	Graph-based	84.05%	78.35%
ϕ_3	Transition-based	83.17%	76.70%
	Graph-based	84.45%	78.56%
Luong et al. [7]	Transition-based	73.03%	66.35%
Some results on the other dependency banks in Vietnamese			
Kiem-Hieu [12]	Graph-based	84.4%	81.4%
Dat Quoc et al. [8]	Graph-based (MSTParser)	79.08%	71.66%
Dat Quoc et al. [11]	Graph-based (Neural network)	80.66%	73.53%

for Vietnamese parsing in two cases: with or without using the distributed word representations feature in the Bist-parser system. The accuracy of our system is UAS=78.17% and LAS= 74.84% when we use gloVe model for producing distributed word representations on Vietnamese universal dependency. This result is the highest accuracy in comparison with the previous researches. It increases about 5.0%, with details increasing from 73.21% to 78.17% and from 68.32% to 74.84% for USA and LSA respectively. This system gets state of the art performance on Viettreebank [7] with UAS=84.45% and LAS= 78.56%.

In the future, we will integrate the CRF into this system. We also conduct another approach to apply this model to a constituency-based structure in Vietnamese.

References

1. Chen, D., Manning, C.D.: A fast and accurate dependency parser using neural networks. In Moschitti, A., Pang, B., Daelemans, W., eds.: EMNLP, ACL (2014) 740–750
2. Dyer, C., Ballesteros, M., Ling, W., Matthews, A., Smith, N.A.: Transition-based dependency parsing with stack long short-term memory. CoRR **abs/1505.08075** (2015)

3. Kiperwasser, E., Goldberg, Y.: Simple and accurate dependency parsing using bidirectional lstm feature representations. CoRR **abs/1603.04351** (2016)
4. Dozat, T., Manning, C.D.: Deep biaffine attention for neural dependency parsing. CoRR **abs/1611.01734** (2016)
5. Minh, N.L., Điệp, H.T., Kế, T.M.: Nghiên cứu luật hiệu chỉnh kết quả dùng phương pháp MST phân tích cú pháp phụ thuộc tiếng việt. In: ICT-rda 8, Hanoi, Vietnam (2008) 258–267
6. Le-Hong, P., Nguyen, T.M.H., Azim, R.: Vietnamese parsing with an automatically extracted tree-adjoining grammar. In: Proceedings of the IEEE International Conference in Computer Science: Research, Innovation and Vision of the Future, RIVF, HCMC, Vietnam (2012)
7. T.L., N., M.L., H., V.H., N., T.M.H., N., P, L.H.: Building a treebank for vietnamese dependency parsing. In: International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future, RIVF 2013, Hanoi, Vietnam, November 10-13, 2013, IEEE (2013) 147–151
8. Nguyen, D.Q., Nguyen, D.Q., Pham, S.B., Nguyen, P.T., Nguyen, M.L.: From Treebank Conversion to Automatic Dependency Parsing for Vietnamese. In: Proceedings of 19th International Conference on Application of Natural Language to Information Systems. (2014) 196–207
9. Le-Hong, P., Nguyen, T.M.H., Nguyen, T.L., Ha, M.L. In: Fast Dependency Parsing Using Distributed Word Representations. Springer International Publishing, Cham (2015) 261–272
10. Nguyen, T.L., Ha, M.L., Le-Hong, P., Nguyen, T.M.H. In: Using distributed word representations in graph-based dependency parsing for Vietnamese. (2016) 804–810
11. Nguyen, D.Q., Dras, M., Johnson, M.: An empirical study for vietnamese dependency parsing. In: Proceedings of the Australasian Language Technology Association Workshop 2016, Melbourne, Australia (2016) 143–149
12. Nguyen, K.H.: Bktreebank: Building a vietnamese dependency treebank. CoRR **abs/1710.05519** (2017)
13. Le-Hong, P., Nguyen, T.M.H., Nguyen, P.T., Roussanaly, A.: Automated extraction of tree adjoining grammars from a treebank for Vietnamese. In: Proceedings of The Tenth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+10), Yale University, New Haven, CT, USA (2010)
14. McDonald, R.T., Nivre, J.: Analyzing and integrating dependency parsers. Computational Linguistics **37** (2011) 197–230
15. McDonald, R., Crammer, K., Pereira, F.: Online large-margin training of dependency parsers. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL’05). (2005) 91–98
16. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9** (1997) 1735–1780
17. Marneffe, M.C.D., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., Manning, C.D.: Universal stanford dependencies: a cross-linguistic typology. In Chair), N.C.C., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., Piperidis, S., eds.: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14), Reykjavik, Iceland, European Language Resources Association (ELRA) (2014)
18. Phuong, L.e., Thi Minh Huyen, N., Roussanaly, A., Vinh, H.T. In: A Hybrid Approach to Word Segmentation of Vietnamese Texts. Springer Berlin Heidelberg, Berlin, Heidelberg (2008) 240–249