DOI: 10.15625/vap.2016.00098

USING DISTRIBUTED WORD REPRESENTATIONS IN GRAPH-BASED DEPENDENCY PARSING FOR VIETNAMESE

Thi-Luong Nguyen[1], My-Linh Ha[2], Phuong Le-Hong[2,3], Thi-Minh-Huyen Nguyen[2]

[1]Da Lat University, Lam Dong, Vietnam
luongnt@dlu.edu.vn
[2] VNU University of Science, Hanoi, Vietnam
phuonglh@vnu.edu.vn, huyenntm@vnu.edu.vn, halinh.hus@gmail.com
[3]FPT Research, Hanoi, Vietnam

ABSTRACT— *Dependency parsing has attracted many natural language processing research groups in recent years. Dependency syntax is a form of sentence representation which has many applications in problems such as question answering system, information extraction, text summarization, machine translation ... Currently, there are many approaches for dependency parsing and they have achieved high accuracy for different languages. For Vietnamese, our research group has used Skip-gram and GloVe model to produce distributed word representations. Then we used this feature in transition-based dependency parsing system. In this paper, we propose to use distributed word representations in graph-based dependency parsing system with MSTParser.*

Keywords— *Dependency syntax, Vietnamese syntax, Vietnamese dependency parsing, distributed word representations.*

I. INTRODUCTION

Dependency parsing has become an important and complex problem in natural language processing. In recent years, this problem has attracted many research groups and achieved high accuracy for English, French, Chinese,… For Vietnamese, some groups developed and achieved quite satisfactory accuracy [6, 7, 15]. However, Vietnamese has neither an inconsistent dependency label set nor a perfect tool for parsing. Meanwhile, the parsing efficiency in Vietnamese is not high.

For Vietnamese dependency parsing, there have been many contributions [6, 7, 10, 15, 17]. In 2008, Nguyễn Lê Minh et al. [7] used MST parser on a corpus consisting of 450 sentences. In 2012, Lê Hồng Phương et al. [10] used a lexicalized tree-adjoining grammar parser trained on a subset of the Vietnamese treebank. In 2014, Nguyen et al [17] used MaltParser and MSTParser on a Vietnamese dependency treebank which is converted automatically from a Vietnamese treebank. In 2015, Lê Hồng Phương et al improved accuracy of Vietnamese dependency parsing, used distributed word representations with Skip-gram and GloVe model [6, 15] for transition-based dependency parsing.

In this paper, we propose to use distributed word representations in a graph-based dependency parsing system. The experiments carried out with the MSTParser system confirm that the use of this feature improve the performance of dependency parsing systems, with an accuracy of 73.092% of the unlabeled attachment score (UAS) and 68.321% of the labeled attachment score (LAS), in comparison with the values 70.296% of UAS and 65.772% of LAS when this feature is not used. The distributed word representations are produced by three recent unsupervised learning models, the Skip-gram model, the CBOW model and the GloVe model. Our main contributions in this paper are:

- First, we present experiments using distributed word representations with three unsupervised learning models: Skip-grams, CBOW and GloVe.
- Second, we integrate distributed word representation features for graph-based dependency parsing.

This paper is organized as follows. Section II gives some background knowledge about dependency parsing which focus on graph-based dependency parsing and distributed word representation. Section III describes our method in detail for dependency parsing. Then, Section IV presents the data, experiments and results. Finally, conlusion and future work are provided in Section V.

II. BACKGROUND

In this section, we present an overview of dependency parsing and distributed word representations.

A. Dependency parsing

The dependency parsing of a sentence consists of determining the binary asymmetric relations, called dependencies, relationship between its lexical elements. A dependency relationship between two tokens can be named to clarify the relationship between them. Dependency structure is determined by the relationship between the center token (*head*) and its dependent token (*dependent*), denoted by an arrow. By convention, the root of the arrow is the head, and the top of the arrow is the dependent.

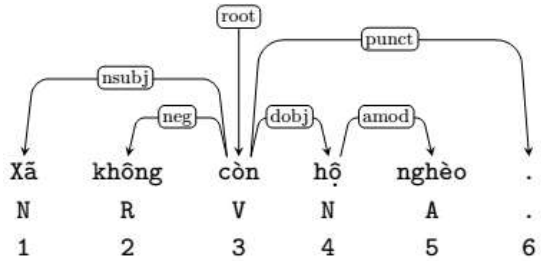

Figure 1. Dependency tree of a Vietnamese sentence

In the above example, the dependency relation from the verb *"còn"* to *"Xã"* labeled *nsubj* indicates that *"Xã"* is subject of this verb. ROOT is the root of the tree and do not correspond to any word in the sentence.

There are two dominant approaches of dependency parsing. The first approach is transition-based parsing, for example MaltParser [3]. The second is graph-based parsing, for example MSTParser [11]. In addition, there are two new approaches for dependency parsing such as hybrid systems [2] and non-directional easy-first parsing [5]. However, in this paper, we developed the second approach that is graph-based parsing. This parsing is described in more details in subsection III.A.

B. Distributed word representations

Recently, distributed word representations have been applied in many natural language processing tasks. A distributed representation is dense, low dimensional and real-valued. Distributed word representations are called word embedding. Each dimension of the embedding represents a latent feature of the word which hopefully captures useful syntactic and semantic similarities [1]. In NLP problems, the words are usually encoded by one-hot vectors (also called indicator vectors) of the same length as the size of the vocabulary, if the word appears in any position, the corresponding components of one-hot vector at that position is 1, otherwise it is 0. This present is quite simple and easy to understand but it has two main problems. The first problem is data sparseness and the second problem is that it is not able to catch the semantic between words. This limitation has prompted some machine learning methods to create a better representations, one of these methods is distributed word representations. This represent are proof that it is more effectively for supporting many tasks in natural language processing, for example machine translation, speech recognition.

III. METHODS

A. Graph-based dependency parsing

In this paper, we employ the graph-based dependency parsing algorithm introduced by [16]. In this algorithm, the weights of the edges are calculated for building dependency graphs of s a sentence as follows:

$$s(i,j)=\mathbf{w}.f(i,j)$$

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:

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

Then, based on the weights of all edges in graph, McDonald et al (2005) 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. In Figure 2, an example graph G and the dependency tree maximizing the scoring function are given for the sentence "*John saw Mary"*.

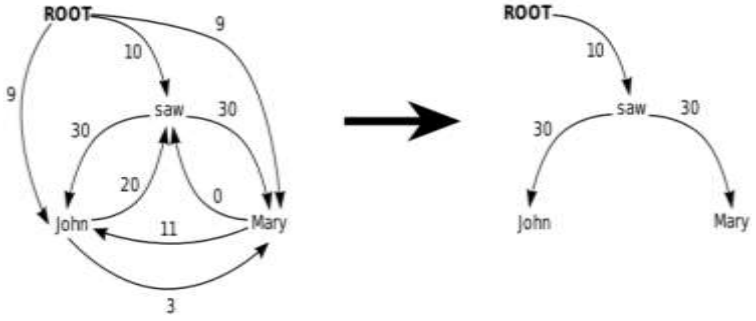

Figure 2. Example of Graph-based dependency parsing

B. Skip-gram model

In this model [1] we are given a corpus of words w and their contexts c. Now, let's define the skip-gram neural network model as in Figure 3.

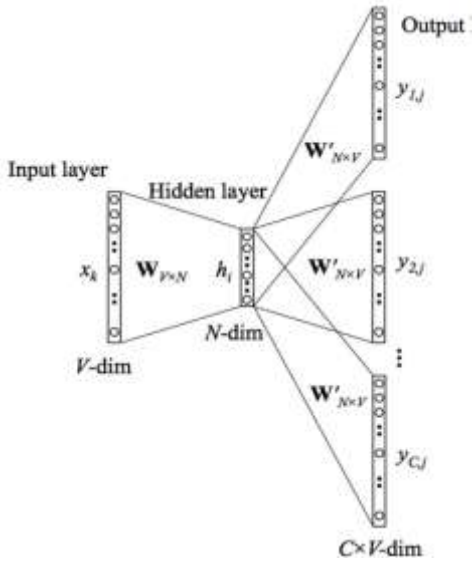

Figure 3. Skip-gram model

Input layer: w_i with vector one-hot X.
Hidden layer: vector h with N-dim.
Output layer: probabilities for words within the context of w_i.
Learning process:

- $W_{V\times N}$ is the weight matrix between the input layer and the hidden layer whose the i[th] row represents the weights corresponding to the i[th] word of the vocabulary. Each output word vector has an associated N×V output matrix $W'_{V\times N}$. There is a hidden layer consisting of N nodes (the exact size of N is a training parameter).
- The input to unit in the hidden layer h_i is denoted by the weighted sum of its inputs. Since the input vector x is *one-hot* encoded, the weights coming from the nonzero element will be the only ones contributing to the hidden layer. Therefore, for the input x with $x_k = 1$ and $x_{k'} = 0\ \forall\ k \neq k'$, the outputs of the hidden layer will be equivalent to the k[th] row of W:

$$h = x^T W = W_k := v_{w_i}$$

- Therefore, the input to the j[th] node of the c[th] output word is:

$$u_{c,j} = v'_{w_j} h$$

- However it is obvious that all of the output layers of the same output word share the same weights; therefore: $u_{c,j} = u_j$. It finally computes the output of the j[th] node of the c[th] output word via the *softmax* function which produces a multinomial distribution:

$$p(w_{c,j} = w_{O,c}|w_i) = y_{c,j} = \frac{exp(u_{c,j})}{\sum_{j=1}^{V} exp(u_j)}$$

- Update the weighted matrix in the hidden-output layer:
  - Compute: $y_{c,j} - t_{c,j}$
  - Update:

$$w'^{(new)}_{ij} = w'^{(old)}_{ij} - \eta . \sum_{c=1}^{C} (y_{c,j} - t_{c,j}) . h_i$$

- Update the weighted matrix in the input-hidden layer:

$$w_{ij}^{(new)} = w_{ij}^{(old)} - \eta . \sum_{j=1}^{V} \sum_{c=1}^{C} (y_{c,j} - t_{c,j}) . w'_{ij} . x_j$$

(With η is a parameter to the learning process**)**

C. CBOW model

CBOW model [1] is proposed based on the neural network. This model predicts words based on the context of the neighbor words, the input is a set of words $\{w_1, w_2,...,w_t\}$ in the context C, this model aims to optimize the parameter in order to maximize the occurrence conditional probability of word *w* within the context C.
The CBOW model is illustrated in Figure 4.

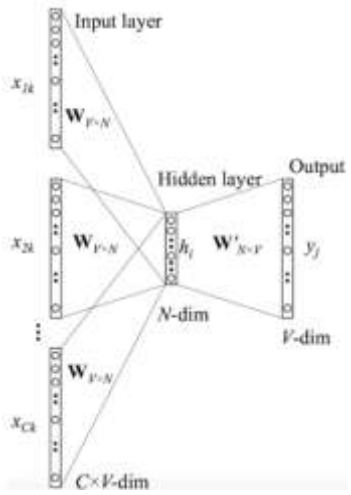

Figure 4. CBOW model

Input layer: the context C with k-dim on the set of vocabulary V.
Hidden layer: vector h with N-dim.
Output layer: occurrence probabilities for a word y_j in the context C.
Learning process:

- Similarly to proposed Skip-gram model, $W_{V\times N}, W'_{V\times N}$ are two weighted matrixs.
- Transfer function: *tang* function, *softmax* function to compute the probability:

$$P(w_O|w_I) = y_i = \frac{exp(w_I.w'^T_O)}{\sum_{j=1}^{V} exp(w_I.w'^T_j)}$$

- Update the weight matrix in the hidden-output layer:
  - Compute: $t_j - y_j = e_j$
  - Update: $w'^{T(t)}_j = w'^{T(t-1)}_j - v.e_j.h_j$
    (h_j is a vector corresponding to the input w_j)
- Update the weight matrix in the input-hidden layer:
  - Compute: $\sum_{j=1}^{V} e_j . W'_{i,j} = EH_i$
  - Update: $W_{w_I}^{(t)} = W_{w_I}^{(t-1)} - v.EH$

(With v is a patameter to the learning process)

D. GloVe: Global Vectors for Word Representation

GloVe is proposed by Jeffrey Pennington et al [4]. GloVe is an unsupervised learning algorithm to obtain vector representations of words. The training process is performed on the aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. It uses the matrix of word-word co-occurrence counts denoted by X, whose entries X_{ij} tabulate the number of times word j occurs in the context of word i. Let $X_i = \sum_k X_{ik}$ be the number of times a word appears in the context of the word i. Finally, let $P_{ij} = P(j|i) = X_{ij}/X_i$ be the probability of the word j appears in the context of the word i.
The above argument suggests that the appropriate starting point for the word vector learning should be with the ratios of the co-occurrence probabilities rather than the probabilities themselves. Noted that the ratio P_{ik}/P_{jk}depends on three words i, j, k.The most general model takes the form:

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

Where all $w \in \mathbb{R}^d$ are word vectors and all $\tilde{w} \in \mathbb{R}^d$ are separate context word vectors. Finally, adding an additional bias $\tilde{b}_k$ for $\tilde{w}_k$ to restore the symmetry,

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = log(X_{ik})$$

This model is minimizing an objective function J, which evaluates the sum of all squared errors based on the above equation, weighted with a function f:

$$J = \sum_{i,j=1}^{V} f(X_{ij}) \left(w_i^T \tilde{w}_k + b_i + \tilde{b}_k - log(X_{ik})\right)^2$$

where V is the size of the vocabulary.
One class of the weighting functions found to work well can be parameterized as:

$$f(x) = \begin{cases} (x/x_{max})^{\alpha} if x < x_{max} \\ 1 \qquad otherwise \end{cases}$$

When it encounters extremely common word pairs ($x > x_{max}$) this function will trim off the normal output and simply returns to one. For all other word pairs, it returns some weight between 0 and 1 where the distribution of the weights within the (0, 1) range is decided by α. It uses the training code from tool GloVe[1].

IV. EXPERIMENTS

A. Data

We use the similar database in our research [6, 15].

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 that 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 [9]. 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 in Vietnamese: We conduct our experiments on the Vietnamese dependency treebank dataset. This treebank is derived automatically from the constituency-based annotation of the VTB [8], 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 with a training set of 2,305 sentences, and a test set of 576 sentences.

B. Vietnamese Dependency Parsing based-on MSTParser

MSTParser is developed by Ryan McDonald *et al* [12]. MSTParser has two processes: training and analysis. In training, MSTParser uses on-line algorithms [11]. In analysis, MSTParser uses a graph-based algorithm. The accuracy of MSTParser on a variety of languages is quite high: ASU= 92.8%, ASL= 90.7% for Japanese, ASU= 91.1%, ASL= 85.9% for Chinese, ASU= 90.4%, ASL = 87.3% for German. . .

Table 1 is the features set that are used by MSTParser[2]. We denote (x_i, x_j) to indicate that there is a directed edge from word x_i to word x_j. The basic features of this model are outlined in Table 1a and b. All features are conjoined with the direction of attachment as well as the distance between two words being attached. In Table 1c, the features take the form of a POS trigram: the POS of the parent, of the child, and of a word in between, for all words linearly between the parent and the child. We denote x_i-word for word of the parent node in the dependency tree, x_j-word for word of child nodes, x_i-pos to POS of the parent node, x_j-pos to POS of child nodes, x_i-pos+1 to POS to the right of the parent in sentence, x_i-pos-1to POS to the left of the parent, x_j-pos+1 to POS to the right of the child, x_j-pos-1 to POS to the left of child, b-pos to POS of a word in between parent and child nodes [14].

a **Basic Uni-gram features**
x_i-word, x_i-pos
x_i-word
x_i-pos
x_i-word, x_i-pos
x_i-word
x_i-pos

b **Basic Bi-gram features**
x_i-word, x_i-pos, x_i-word, x_i-pos
x_i-pos, x_i-word, x_i-pos
x_i-word, x_i-pos, x_i-pos
x_i-pos, x_i-word, x_i-pos
x_i-word, x_i-pos, x_i-word
x_i-word, x_i-word
x_i-pos, x_i-pos

c
In between POS features
x_i-pos, b-pos, x_j-pos
Surrounding word POS features
x_i-pos, x_i-pos + 1, x_j-pos – 1, x_j-pos
x_i-pos - 1, x_i-pos, x_j-pos – 1, x_j-pos
x_i-pos, x_i-pos + 1, x_j-pos, x_j-pos + 1
x_i-pos - 1, x_i-pos, x_j-pos, x_j-pos + 1

Table 1. Features used by MSTParser system

Let Φ_0 is the base feature set, Φ_0 consists of the following feature templates:

- Part-of-speech tags and word in basic Uni-gram features (Table 1 a)
- Part-of-speech tags and word in basic Bi-gram features (Table 1 b)
- Part-of-speech tags in between POS features, surrounding word and POS features (Table 1 c)

[1]http://nlp.stanford.edu/projects/glove/
[2]http://sourceforge.net/projects/mstparser/

We test with Φ_0 using MSTParser in our ***dependency treebank in Vietnamese***, then we have result UAS=70.296%, LAS=65.772%.

C. Integrating distributed word representations to MSTParser

We integrate distributed word representations with our features set in MSTParser. We add distributed word representations for x_i, x_j, b (word between x_i and x_j) in Table 2.

Feature set	Feature template
Φ_1	Φ_0, $s(x_i)$, $s(x_j)$, s(b), $s(x_i+1)$, $s(x_i-1)$, $s(x_j+1)$, $s(x_j-1)$
Φ_2	Φ_0, $c(x_i)$, $c(x_j)$, c(b), $c(x_i+1)$, $c(x_i-1)$, $c(x_j+1)$, $c(x_j-1)$
Φ_3	Φ_0, $g(x_i)$, $g(x_j)$, g(b), $g(x_i+1)$, $g(x_i-1)$, $g(x_j+1)$, $g(x_j-1)$

Table 2. Our feature sets in MSTParser

We use two attachment scores, labeled attachment score (LAS) and unlabelled attachment score (UAS) to evaluate the accuracy of MSTParser. 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).

The accuracy of MSTParser system with different feature sets is shown in Table 3. The feature set Φ_1 consists of the features defined in Φ_0, plus the distributed word representations produced by the Skip-gram model: x_i, x_j, b (word between x_i to x_j), word on the left and on the right of x_i, word on the left and on the right of x_j. The feature set Φ_2 and Φ_3 consists of the features defined in Φ_0 like Φ_1, but replaces distributed word representations produced by the CBOW model and GloVe model. In this result, we show that distributed word representations are produced by GloVe model has the best accuracy.

Feature	Train		Test	
	UAS	LAS	UAS	LAS
Φ_0	95.658%	93.361%	70.296%	65.772%
Φ_1	98.093%	96.079%	72.665%	67.961%
Φ_2	97.916%	95.938%	72.452%	67.771%
Φ_3	**98.048%**	**96.055%**	**73.092%**	**68.321%**

Table 3. Accuracy of MSTParser with new feature sets

V. CONCLUSION

In this paper, we presented the integration of distributing word representations in a graph-based dependency parsing system which is the MSTParser system. We evaluated the accuracy of the system for Vietnamese parsing in two cases: with or without using the distributed word representations feature in MSTParser. We used three different models (Skip-gram, CBOW and GloVe) and it is shown that the GloVe model is the best for producing distributed word representations feature in graph-based dependency parsing.

The accuracy of our system are UAS=73.092%, and LAS=68.321% when we use GloVe model for producing distributed word representations. Although this result is less efficient than our previous research [6, 15], we showed that the use of distributed word representations feature allows to obtain a better result than without this feature (UAS=70.296% and LAS=65.772%) in MSTParser system.

In the future, we will integrate the distributed word representations for hybrid systems. Another approach that we can adopt is unsupervised dependency parsing, a recent trend that comes with several benefits.

VI. ACKNOWLEDGEMENTS

We are grateful to our anonymous reviewers for their helpful comments which helped us improve the quality of the article in terms of both the presentation and the content.

REFERENCES

[1] Turian, J., Ratinov, L., Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. Proceedings of ACL, (pp. 384–394). Uppsala, Sweden.

[2] J. Nivre and R. McDonald. (2008). Integrating graph-based and transition-based dependency parsers. ACL-08, (pp. 950–958). Columbus Ohio, USA.

[3] J. Nivre, J. Hall, J. Nilsson, C. Atanas, G. Erigit,. (2007). Maltparser: A language-independent system for data-driven dependency parsing. Natural Language Engineering, (pp. 13:95–135).

[4] Jeffrey Pennington and Richard Socher and Christopher D. Manning. (2014). GloVe: Global Vectors for Word Representation. In Empirical Methods in Natural Language Processing (EMNLP) (pp. 1532--1543).

[5] K. Sagae and A. Lavie. (2006). A best-first probabilistic shift-reduce parser. COLING/ACL 2006, (pp. 691–698). Sydney, Australia.

[6] Le Hong, P., T. M. H. Nguyen, T. L. Nguyen, and M. L. Ha. (2015). Fast Dependency Parsing using Distributed Word Representations. Springer Proceedings of PAKDD 2015 Workshops, Trends and Applications in Knowledge Discovery and Data Mining, vol. LNAI 9441 (pp. 261-272). HCM City, Vietnam: Springer.

[7] N. L. Minh, H. T. Điệp, and T. M. Kế. (2008). Nghiên cứu luật hiệu chỉnh kết quả dùng phương pháp MST phân tích cú pháp phụ thuộc tiếng Việt. ICT-rda 8, (pp. 258–267). Hà Nội, Việt Nam.

[8] Nguyen T. L., M. L. Ha, V. H. Nguyen, T. M. H. Nguyen, and P. Le Hong. (2013). Building a Treebank for Vietnamese Dependency Parsing. The 10th IEEE RIVF International Conference on Computing and Communication Technologies (pp. 147--151). Hanoi, Vietnam: IEEE.

[9] P. Le Hong, T. M. H. Nguyen, A. Roussanaly, and T. V. Ho. (2008). A hybrid approach to word segmentation of Vietnamese texts. Proceedings of LATA, LNCS 5196 (pp. 240–249). Springer.

[10] P. Le Hong, T. M. H. Nguyen, and R. Azim. (2012). Vietnamese parsing with an automatically extracted tree-adjoining grammar. RIVF. HCMC, Vietnam.

[11] R. McDonald and F. Pereira. (2006). Online learning of approximate dependency parsing algorithms. EACL, (pp. 81–88). Trento, Italy.

[12] R. McDonald, F. Pereira, K. Ribarov, and J. Haji . (2005). Non-projective Dependency Parsing using Spanning Tree Algorithms. Proceedings of HLT/EMNLP .

[13] R. McDonald, K. Lerman, and F. Pereira. (2006). Multilingual dependency parsing with a two-stage discriminative parser. Proceedings of the Tenth Conference on Computational Natural Language Learning.

[14] Ryan McDonald, Koby Crammer and Fernando Pereira. (2005). Online Large-Margin Training of Dependency Parsers. 43rd Annual Meeting of the Association for Computational Linguistics. ACL.

[15] Vu, M. C., A. T. Luong, and P. Le Hong. (2015). Improving Vietnamese Dependency Parsing using Distributed Word Representations. The Sixth International Symposium on Information and Communication Technology (pp. 54-60). Hue, Vietnam: ACM.

[16] R. McDonald and J. Nivre (2011). Analyzing and integrating dependency parsers. Computational Linguistics, (pp. 197–230).

[17] D. Q. Nguyen, D. Q. Nguyen, S. B. Pham, P. T. Nguyen, and M. L. Nguyen. (2014). From treebank conversion to automatic dependency parsing for Vietnamese. In Proceedings of the 19th International Conference on Application of Natural Language to Information Systems, NLDB'14, pages 196–207.

SỬ DỤNG BIỂU DIỄN PHÂN TÁN CỦA TỪ CHO PHÂN TÍCH CÚ PHÁP PHỤ THUỘC DỰA VÀO ĐỒ THỊ TRÊN TIẾNG VIỆT

Nguyễn Thị Lương, Hà Mỹ Linh, Lê Hồng Phương, Nguyễn Thị Minh Huyền

TÓM TẮT— *Phân tích cú pháp phụ thuộc là một vấn đề đã thu hút sự quan tâm của nhiều nhóm nghiên cứu xử lý ngôn ngữ tự nhiên trong những năm gần đây. Cú pháp phụ thuộc là một dạng biểu diễn phụ thuộc ngữ nghĩa của các từ trong câu có nhiều ứng dụng cho các bài toán phức tạp như hệ thống hỏi đáp, trích chọn thông tin, tóm tắt văn bản, dịch máy... Hiện nay, đã có rất nhiều hướng nghiên cứu cho bài toán phân tích cú pháp phụ thuộc và đạt được kết quả khả quan cho các ngôn ngữ khác nhau. Đối với tiếng Việt, nhóm nghiên cứu của chúng tôi đã sử dụng mô hình Skip-gram, Glove để sinh biểu diễn phân tán cho từ vựng. Sau đó sử dụng đặc trưng này cho bài toán phân tích cú pháp phụ thuộc dựa trên các bước chuyển. Trong bài báo này, chúng tôi đề xuất sử dụng đặc trưng biểu diễn phân tán của từ cho phân tích cú pháp phụ thuộc dựa trên đồ thị với công cụ MSTParser.*