Improving Vietnamese Dependency Parsing Using Distributed Word Representations

Conference Paper · December 2015

DOI: 10.1145/2833258.2833296

Cam Manh Vu
FPT University
1 PUBLICATION **1** CITATION
SEE PROFILE

Tuan A. Luong
FPT University
1 PUBLICATION **1** CITATION
SEE PROFILE

Le-Hong Phuong
Vietnam National University, Hanoi
31 PUBLICATIONS **147** CITATIONS
SEE PROFILE

Improving Vietnamese Dependency Parsing using Distributed Word Representations

Cam Vu-Manh
FPT University
Hanoi, Vietnam
camvm93@gmail.com

Anh Tuan Luong
FPT University
Hanoi, Vietnam
tuanlase02874@fpt.edu.vn

Phuong Le-Hong
Hanoi University of Science
Hanoi, Vietnam
phuonglh@vnu.edu.vn

ABSTRACT

Dependency parsing has become an important line of research in natural language processing in recent years. This is due to its usefulness in a wide variety of real world applications. This paper presents the improvement of Vietnamese dependency parsing using distributed word representations. Our parser achieves an accuracy of 76.29% of unlabelled attachment score or 69.25% of labelled attachment score. This is the most accurate dependency parser for the Vietnamese language in comparison to others which are trained and tested on the same dependency treebank. The distributed word representations are produced by two recent unsupervised learning models, the Skip-gram model and the GloVe model. We also show that distributed representations produced by the GloVe model are better than those produced by the Skip-gram model when being used in dependency parsing. Our dependency parsing system, including software, corpus and distributed word representations, is released as an open source project, freely available for research purpose.

Keywords

dependency parsing, distributed word representations, skip-gram, glove, natural language processing; Vietnamese

Categories and Subject Descriptors

I.2.7 [**Natural Language Processing**]: language parsing and understanding, text analysis

1. INTRODUCTION

Dependency parsing has become an important line of research in natural language processing (NLP) in recent years. This is due to its usefulness in a wide variety of real world applications of NLP, for example, machine translation [4], question and answering [5], sentiment analysis [26]...

The objective of dependency parsing is to find a dependency representation of a sentence. In this representation, the syntactic structure of a sentence is modeled by a dependency graph, which represents each word and its syntactic modifiers through labelled directed arcs, where each arc label comes from a finite set representing possible syntactic roles. Figure 1 shows the dependency analysis of a Vietnamese sentence. In this example, the dependency relation from the finite verb `vào` (enter) to `tôi` (I) labelled `psubj` indicates that `tôi` is the pronominal subject of the finite verb. An artificial word ROOT has been inserted (not shown in the figure), serving as the unique root of the graph. This is a standard device that simplifies both theoretical definitions and computational implementations.

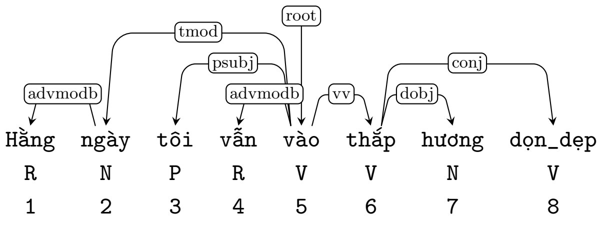

Figure 1: Syntactic dependency structure tree.

Dependency parsing using feature-based discriminative statistical parser has seen enormous success in recent years. Most of the parsers use millions of indicator features [11] and hence suffer from the problem of feature sparseness. From a statistical perspective, their corresponding parameters are poorly estimated because there is insufficient data to correctly estimate such parameters. To alleviate this problem, higher-support features such as word classes have been incorporated to improve parsing performance [7]. However, these parsers are still not perfect in terms of both speed and efficiency.

In this work, we overcome this drawback by using dense features of distributed word representations, also called word embeddings. We present a good dependency parser for the Vietnamese language by integrating distributed representations of words into a transition-based parsing system. Our parsing system achieves an accuracy of 76.29% of unlabelled attachment score or 69.25% of labelled attachment score. The distributed word representations are produced by two recent unsupervised learning models, the Skip-gram [13] model and the GloVe model [23].

Our main contributions in this paper are:

- First, we develop of a state-of-the-art dependency parser

SoICT 2015, December 03-04, 2015, Hue City, Viet Nam
© 2015 ACM. ISBN 978-1-4503-3843-1/15/12...$15.00
DOI: http://dx.doi.org/10.1145/2833258.2833296

on the Vietnamese Dependency Treebank [19];

- Second, we give an empirical study on using distributed word representations in a transition-based dependency parser. We show that distributed representations produced by the GloVe model are better than those produced by the Skip-gram model;

- Third, our dependency parsing system, including the corpus and the parser, is released as an open source project for Vietnamese NLP community, available for free use for research purpose.

The remainder of the paper is structured as follows. Section 2 gives some background knowledge, including a brief introduction to dependency parsing and distributed word representations. Next, Section 3 describes in detail the techniques used in our dependency parser. Then, Section 4 presents the experiments and evaluation results. Finally, Section 5 concludes the paper and suggests some directions for future work.

2. BACKGROUND

In this section, we give a brief overview of dependency parsing methods and distributed word representations.

2.1 Dependency Parsing

Most statistical dependency parsers developed recently use the data-driven approach, which are trained on an annotated dependency corpus of a language. Current dependency parsers can be categorized into four families [6]. The first family is local-and-greedy transition-based parser, for example MaltParser [21]. The second family is globally optimized graph-based parsers, for example MSTParser [12]. The third family is hybrid systems, for example [22] or [24] which combine various parsers. The fourth family is non-directional easy-first parsing [6].

The parser developed in this paper belongs to the first family, that is, it is a transition-based parser. This parser is described in detail in the next subsection 3.1.

2.2 Distributed Representations

In most NLP tasks, the words are usually encoded as symbolic identifiers which are drawn from a vocabulary. Therefore, they are often represented by one-hot vectors (also called indicator vectors) of the same length as the size of the vocabulary. This representation suffers from two major problems. The first problem is data sparseness, that is, the parameters corresponding to rare or unknown words are poorly estimated. The second problem is that it is not able to capture the semantic similarity between closely related words. This limitation of the one-hot word representation has motivated unsupervised methods for inducing word representations over large, unlabelled corpora.

Recently, distributed representations of words have been shown to be advantageous in many natural language processing tasks. A distributed representation is dense, low dimensional and real-valued. Distributed word representations are called word embeddings. Each dimension of the embedding represents a latent feature of the word which hopefully captures useful syntactic and semantic similarities [25].

Word embeddings are typically induced using neural language models, which use neural networks as the underlying predictive model. Historically, training and testing of neural language models has been slow, scaling as the size of the vocabulary for each model computation [1]. However, many approaches have been recently proposed to speed up the training process, allowing scaling to very large corpora [16, 3, 15, 13].

Another method to produce word embeddings has been introduced recently by the NLP group at the Stanford university [23]. They proposed a global log-bilinear regression model that combines the advantages of the two major model families in the literature: global matrix factorization and local context window methods.

We present in the subsections 3.2 and 3.3 how we use a neural language model and a global log-bilinear regression model, respectively, to produce word embeddings for Vietnamese which are used in this study.

3. METHODOLOGY

In this section, we describe the core components of our dependency parsing system, including the arc-eager parsing algorithm, the Skip-gram model and the GloVe model.

3.1 Arc-eager Parsing Algorithm

In 2003, Nivre presented the arc-eager algorithm [20], which is an efficient algorithm for projective dependency parsing. In an arc-eager system, a *configuration* $c = (\sigma, \beta, A)$ consists of *a stack* σ, *a buffer* β and *a set of dependency arcs* A. The initial configuration for a sentence $s = w_1, w_2, \dots, w_n$ is $\sigma = [\text{root}], \beta = [w_1, \dots, w_n]$ and $A = \emptyset$. A configuration c is terminal if the buffer is empty and the stack contains a single element root. We use the notation $v|\beta$ to indicate that the first element of the buffer is the word v; the notation $\sigma|u$ to indicate that the top element of the stack is the word u; and $A_c = \{(x, y)\}$ where x, y are words of a sentence being parsed to indicate the set of dependency arcs of a configuration c.

The arc-eager parsing algorithm defines four types of transitions as shown in Table 2. In the labelled version of parsing, there are in total $|\mathcal{T}| = 2N_l + 2$ transitions where N_l is the number of different arc labels. The preconditions of the four transition types are explained as follows:

- The precondition of LEFT-ARC $u \leftarrow v$ is that there does not exist any arc coming to u; in other words, u has not been a dependent of another word. After this transition, the parsing of u is done and popped from the stack.

- The precondition of RIGHT-ARC $u \rightarrow v$ is that there does not exist any arc coming to v. After this transition, the word v is pushed onto the stack to consider next word. Note that there can be multiple arcs coming out of u.

- The REDUCE transition pops the stack and presupposes that the top element has already been attached to its head in a previous RIGHT-ARC transition.

- The SHIFT transition extracts the first element of the buffer and pushes it onto the stack. This transition does not require any precondition.

Figure 3 illustrates an example of one transition sequence from the initial configuration to a terminal one.

The arc-eager parsing algorithm is deterministic which defines an incremental left-to-right parsing order. A crucial

Name	**Operation**	**Precondition**
LEFT-ARC	$(\sigma\|u, v\|\beta, A) \Rightarrow (\sigma, v\|\beta, A \cup \{(v, u)\})$	$\nexists k : (k, u) \in A$
RIGHT-ARC	$(\sigma\|u, v\|\beta, A) \Rightarrow (\sigma\|u\|v, \beta, A \cup \{(u, v)\})$	$\nexists k : (k, v) \in A$
REDUCE	$(\sigma\|u, \beta, A) \Rightarrow (\sigma, \beta, A)$	$\exists v : (v, u) \in A$
SHIFT	$(\sigma, v\|\beta, A) \Rightarrow (\sigma\|v, \beta, A)$	

Figure 2: Four transition types of the arc-eager parsing algorithm.

Transition	**Stack**	**Input**	A
	[root]	[Hằng,ngày,tôi,vẫn,vào,thấp,hương,dọn_dẹp]	$\emptyset$
SHIFT	[root,Hằng]	[ngày,tôi,vẫn,vào,thấp,hương,dọn_dẹp]	
LEFT-ARC$_{advmodb}$	[root]	[ngày,tôi,vẫn,vào,thấp,hương,dọn_dẹp]	+advmodb(ngày,Hằng)
SHIFT	[root,ngày]	[tôi,vẫn,vào,thấp,hương,dọn_dẹp]	
SHIFT	[root,ngày,tôi]	[vẫn,vào,thấp,hương,dọn_dẹp]	
SHIFT	[root,ngày,tôi,vẫn]	[vào,thấp,hương,dọn_dẹp]	
LEFT-ARC$_{advmodb}$	[root,ngày,tôi]	[vào,thấp,hương,dọn_dẹp]	+advmodb(vào,vẫn)
LEFT-ARC$_{psubj}$	[root,ngày]	[vào,thấp,hương,dọn_dẹp]	+psubj(vào,tôi)
LEFT-ARC$_{tmod}$	[root]	[vào,thấp,hương,dọn_dẹp]	+tmod(vào,ngày)
RIGHT-ARC$_{root}$	[root,vào]	[thấp,hương,dọn_dẹp]	+root(root,vào)
RIGHT-ARC$_{vv}$	[root,vào,thấp]	[hương,dọn_dẹp]	+vv(vào,thấp)
RIGHT-ARC$_{dobj}$	[root,vào,thấp,hương]	[dọn_dẹp]	+dobj(thấp,hương)
REDUCE	[root,vào,thấp]	[dọn_dẹp]	
RIGHT-ARC$_{conj}$	[root,vào,thấp,dọn_dẹp]	[]	+conj(thấp,dọn_dẹp)

Figure 3: Nivre arc-eager dependency parsing example

component of such an algorithm is a statistical classifier which maps each parser configuration to a transition. This classifier is usually induced from a treebank by a machine learning method.

3.2 Skip-gram Model

We use word embeddings produced by Mikolov's continuous Skip-gram model using the neural network and source code introduced in [14]. The continuous skip-gram model itself is described in details in [13].

For our experiments we used a continuous skip-gram window of size 2, *i.e.* the actual context size for each training sample is a random number up to 2. The neural network uses the central word in the context to predict the other words, by maximizing the average conditional log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j=-c}^{c}\log p(w_{t+j}|w_t),$$

where $\{w_i : i \in T\}$ is the whole training set, w_t is the central word and the w_{t+j} are on either side of the context. The conditional probabilities are defined by the softmax function

$$p(a|b) = \frac{\exp(o_a^\top i_b)}{\sum\limits_{w\in\mathcal{V}}\exp(o_w^\top i_b)},$$

where i_w and o_w are the input and output vector of w respectively, and $\mathcal{V}$ is the vocabulary. For computational efficiency, Mikolov's training code approximates the softmax function by the hierarchical softmax, as defined in [16]. Here the hierarchical softmax is built on a binary Huffman tree with one word at each leaf node. The conditional probabilities are calculated according to the decomposition:

$$p(a|b) = \prod_{i=1}^{l} p(d_i(a)|d_1(a)...d_{i-1}(a), b),$$

where l is the path length from the root to the node a, and $d_i(a)$ is the decision at step i on the path (for example 0 if the next node the left child of the current node, and 1 if it is the right child). If the tree is balanced, the hierarchical softmax only needs to compute around $\log_2|\mathcal{V}|$ nodes in the tree, while the true softmax requires computing over all $|\mathcal{V}|$ words.

The training code was obtained from the tool `word2vec`[1] and we used frequent word subsampling as well as a word appearance threshold of 5. The output dimension is set to 50, *i.e.* each word is mapped to a unit vector in $\mathbb{R}^{50}$. This is deemed adequate for our purpose without overfitting the training data. Figure 4 shows the scatter plot of some Vietnamese words which are projected onto the first two principal components after performing the principal component analysis of all the word distributed representations. We can see that semantically related words are grouped closely together.

3.3 GloVe Model

In 2014, Pennington et al. [23] introduced the global vector model for learning word representations (GloVe). Similar to the Skip-gram model, GloVe is a local context window method but it has the advantages of the global matrix factorization method.

The main idea of GloVe is to use word-word occurrence counts to estimate the co-occurrence probabilities rather than the probabilities by themselves. Let P_{ij} denote the probability that word j appear in the context of word i; $w_i \in \mathbb{R}^d$ and $w_j \in \mathbb{R}^d$ denote the word vectors of word i and word j respectively. It is shown that

$$w_i^\top w_j = \log(P_{ij}) = \log(C_{ij}) - \log(C_i),$$

where C_{ij} is the number of times word j occurs in the context of word i.

It turns out that GloVe is a global log-bilinear regression model. Finding word vectors is equivalent to solving a weighted least-squares regression model with the cost func-

[1]http://code.google.com/p/word2vec/

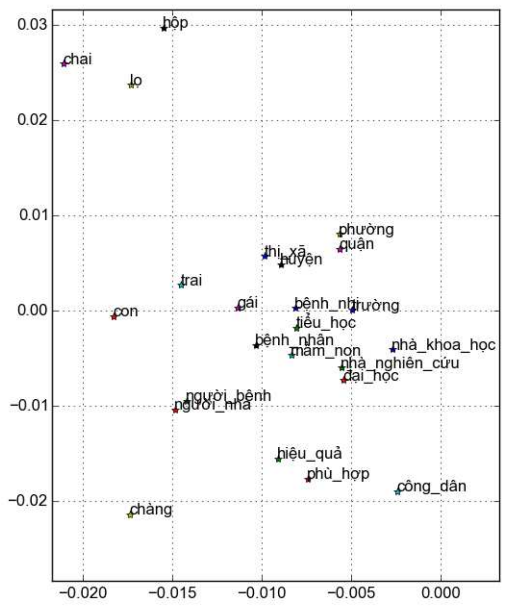

Figure 4: Some Vietnamese words produced by the Skip-gram model, projected onto two dimensions

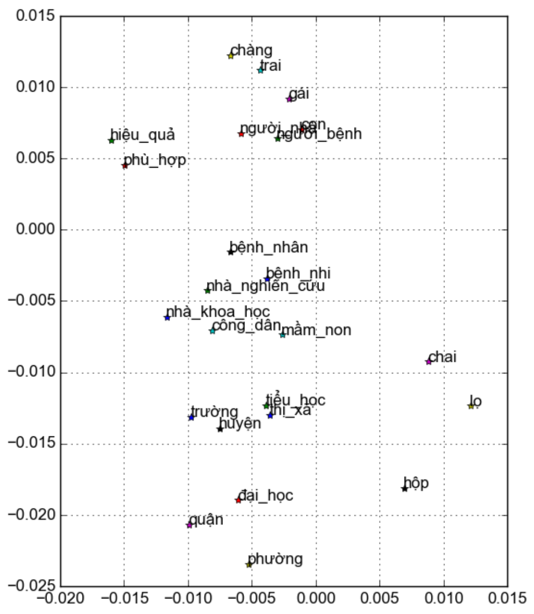

Figure 5: Some Vietnamese words produced by the GloVe model, projected onto two dimensions

tion:

$$J = \sum_{i,j=1}^{n} f(C_{ij})(w_i^\top w_j + b_i + b_j - \log(C_{ij}))^2,$$

where n is the size of the vocabulary, b_i and b_j are additional bias terms and $f(C_{ij})$ is a weighting function. A class of weighting functions which are found to work well can be parameterized as

$$f(x) = \begin{cases} \left(\frac{x}{x_{\max}}\right)^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

The training code was obtained from the tool `GloVe`[2] and we used a word appearance threshold of 2,000. Figure 5 shows the scatter plot of the same words in Figure 4, but this time their word vectors are produced by the GloVe model.

4. EXPERIMENTS

In this section, we first introduce the datasets used in this study and the evaluation methods. Next, we present the baseline performance obtained by MaltParser. Finally, we present and discuss our improved result.

4.1 Datasets

We use two datasets in our system. The first dataset is a text corpus which is used to produce the distributed word representations. The second dataset is a Vietnamese dependency treebank.

4.1.1 Text Corpus

To create distributed word representations, we use a dataset consisting of 7.3GB of text from 2 million articles collected through a Vietnamese news portal.[3] The text is first normalized to lower case and all special characters are removed except these common symbols: the comma, the semicolon, the colon, the full stop and the percentage sign. All numeral sequences are replaced with the special token <number>, so that correlations between certain words and numbers 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 syllables 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 [10].

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.

4.1.2 Dependency Treebank

We conduct our experiments on the Vietnamese dependency treebank dataset [19]. This treebank is derived automatically from the constituency-based annotation of the VTB treebank [18], 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 treebank of 2,700 sentences, containing 34,887 word tokens.

4.2 Evaluation Methods

There are three methods for evaluating dependency parsers, namely attachment scores, exact match and precision/recall [8].

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

[3]http://www.baomoi.com

Algorithm	Data Test	Word		Sentence	
		UAS	LAS	UAS	LAS
Nivre arc-eager	Test	**75.82**	**69.61**	**77.58**	**70.98**
	Train	**89.22**	**86.19**	**90.13**	**86.81**
Nivre arc-standard	Test	74.07	68.43	76.06	70.41
	Train	86.47	82.42	87.66	83.60
Stack projective	Test	74.15	68.53	76.13	70.49
	Train	86.47	82.40	87.66	83.59
Covington projective	Test	75.23	69.22	77.04	70.65
	Train	87.67	84.71	88.76	85.49

Table 1: Accuracy of projective parsing algorithms in MaltParser for Vietnamese

In this paper, we use two attachment scores, labelled attachment score (LAS) and unlabelled attachment score (UAS) to evaluate the accuracy of our parser. 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).

These attachment scores can be computed at two levels, word level or sentence level. For example, if the corpus has only two sentences of 12 relations and 10 relations. The parser is able to identify correctly 9 relations of the first sentence and 7 relations of the second sentence.[4] Then, the LAS at the word level is defined as

$$\mathrm{LAS}_w = \frac{9+7}{12+10}$$

and the LAS at the sentence level is defined as

$$\mathrm{LAS}_s = \frac{\frac{9}{12}+\frac{7}{10}}{2}$$

We use cross validation technique to evaluate the accuracy of the parser. The dependency treebank is divided into 10 subsets, each contains 270 sentences. In each run, 9 subsets are used for training and 1 subset is used for testing. The final accuracy of the parser is the average accuracy over 10 runs.

4.3 Using MaltParser for Vietnamese Parsing

MaltParser[5] is a system for data-driven dependency parsing which is categorized as a local-and-greedy transition-based parser. This parser is robust, efficient and accurate parsing for numerous occidental languages with rather limited amount of training data.

MaltParser implements nine parsing algorithms. Since the Vietnamese dependency treebank contains only projective dependency trees, we focus ourselves on experimenting with four algorithms for projective parsing, including Nivre arc-eager algorithm, Nivre arc-standard algorithm, stack projective algorithm, and Covington projective algorithm. The accuracy of these algorithms on the training set and test set is shown in Table 1. We use the default values for parameters of MaltParser. The best accuracy is marked in bold font.

We see that the Nivre arc-eager algorithm has the highest accuracy, of 75.82% (UAS) or 69.61% (LAS). Therefore, we will use this accuracy as the baseline for our system. However, the current implementation of MaltParser does not provide an easy way to integrate real-valued features into the algorithms. Therefore, instead of reusing MaltParser, we reimplement the Nivre arc-eager parsing algorithm and develop our own parser. Similar to MaltParser, we use LibLinear library[6] in our classifier. The integration of distributed word representations into the parser is presented in the next subsection.

[4]In projective dependency parsing, the number of dependency relations of a sentence is equal to its number of words, excluding the dummy root word.

[5]http://maltparser.org

Feature Set	Feature Templates
Φ_1	$\Phi_0,\ w(\sigma_0), w(\beta_0), w(\beta_1), w(h(\sigma_0))$
Φ_2	$\Phi_0,\ s(\sigma_0), s(\beta_0), s(\beta_1), s(h(\sigma_0))$
Φ_3	$\Phi_1 \cup \Phi_2$
Φ_4	$\Phi_0, g(\sigma_0), g(\beta_0), g(\beta_1), g(h(\sigma_0))$
Φ_5	$\Phi_1 \cup \Phi_4$
Φ_6	$\Phi_1 \cup \Phi_2 \cup \Phi_4$

Table 2: Feature sets

4.4 Integrating Distributed Representations

In Nivre arc-eager parsing algorithm (and others), training data for the transition classifier were generated by parsing each sentence in the training set using the gold-standard dependency graph. For each parser configuration c and transition $f(c)$ in the gold parse, a training instance $(\phi(c), f(c))$ was created, where $\phi(c)$ is the feature vector representation of the parser configuration c.

Table 2 shows the feature templates used for the classifier. We use the notation ℓ_i to denote the ith element of the stack or buffer ℓ, with ℓ_0 for the first element (top element of the stack σ or the first element of the buffer β). The function $w(\cdot)$ extracts the word. The functions $s(\cdot)$ and $g(\cdot)$ extract the distributed representation of the word produced by the Skip-gram and the GloVe model, respectively. The base feature set Φ_0 consists of the following feature templates:

1. Part-of-speech tags of tokens on the buffer and stack: $t(\sigma_0), t(\sigma_1), t(\beta_0), t(\beta_1), t(\beta_2)$
2. Relation of the top token on the stack: $dep(\sigma_0)$
3. Left relation of the top token on the buffer: $ldep(\beta_0)$
4. Left and right relation of the top token on the stack: $ldep(\sigma_0), rdep(\sigma_0)$
5. Joint features: $t(\sigma_0)+t(\beta_0)$, $t(\sigma_0)+w(\beta_0), t(\beta_0)+ldep(\beta_0)$
$t(\sigma_0)+t(\sigma_1)+t(\beta_0), t(\sigma_0)+dep(\beta_0), t(\sigma_0)+t(\beta_0)+t(\beta_1)$
$t(\beta_0)+t(\beta_1)+t(\beta_2), t(\sigma_0)+ldep(\sigma_0)+rdep(\sigma_0)$

The accuracy of our parsing 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 four indicators word features: the top word on the stack, the first and the second word on the buffer, and the head word of the top word on the stack. The feature set Φ_2 consists of the features defined in Φ_0 like Φ_1, but replaces indicator word features with their corresponding distributed representations produced by the Skip-gram model. The feature set Φ_3 consists of all features defined in Φ_1 and Φ_2. The feature set Φ_4 is similar to Φ_2 but replacing the distributed representations of the

[6]http://www.csie.ntu.edu.tw/~cjlin/liblinear/

Feature	Data test	Word		Sentence	
		UAS	LAS	UAS	LAS
Φ_0	Test	72.33	62.59	74.06	64.22
	Train	79.90	72.55	81.46	74.08
Φ_1	Test	75.34	68.33	76.87	69.83
	Train	91.23	88.91	91.94	89.61
Φ_2	Test	74.75	67.68	76.49	69.37
	Train	84.04	79.66	85.44	81.08
Φ_3	Test	75.94	68.92	77.44	70.39
	Train	92.09	90.08	92.75	90.74
Φ_4	Test	74.77	67.78	76.59	69.47
	Train	85.04	79.86	86.44	82.08
Φ_5	Test	**76.29**	**69.25**	77.71	70.67
	Train	92.11	90.08	92.76	90.73
Φ_6	Test	76.10	69.01	**77.72**	**70.59**
	Train	**92.64**	**90.80**	**93.39**	**91.58**

Table 3: Accuracy of our dependency parser

Skip-gram model by those produced by the GloVe model. Φ_5 consists of all features defined in Φ_1 and Φ_4. Finally, Φ_6 is the most complete feature set, containing all the indicator and distributed features.

4.5 Discussion

We see that distributed word representations, either produced by the Skip-gram model or by the GloVe model help improve the parsing accuracy. In particular, the feature sets Φ_3 and Φ_5 both give a better result than the feature set Φ_1; for example, 75.94% (Φ_3 UAS$_w$) or 76.29% (Φ_5 UAS$_w$), compared to 75.34% (Φ_1 UAS$_w$).

We also see that the distributed representations produced by the GloVe model is superior than those produced by the Skip-gram model. The difference of accuracy between feature sets Φ_3 and Φ_5 is about 0.35%. In the original work of the GloVe model, it has been shown to outperform the Skip-gram model on word similarity tasks and named entity recognition. In this work, we have shown that it also outperforms the related model on dependency parsing.

The accuracy of the feature set Φ_5 is roughly 1.00% better than the accuracy of Φ_1. This indicates that the indicator features and the distributed features can work well together.

The accuracy of our parser is slightly inferior than that of MaltParser on the same feature set, for example on Φ_1, we achieve 75.34%, compared to 75.82% of UAS$_w$. This may be due to some advanced optimization technique used in MaltParser. However, when distributed word features are integrated, our parser outperforms MaltParser by a margin of 0.47% of UAS$_w$.

Compared to existing works on the dependency parsing of Vietnamese, our result is about 3% better than the parser presented in [19], in both UAS or LAS. However, this is only a relative comparison since their parsing system is trained and tested on different parts of the Vietnamese dependency treebank of about 10,000 sentences, which are automatically converted from VTB.

In 2014, Nguyen et al. [17] presented an empirical study of dependency parsing for Vietnamese. As in the work above, they use MaltParser and MSTParser on a Vietnamese dependency treebank which is converted automatically from VTB. They report the best accuracy of 77.37% of UAS$_w$ with the MaltParser. However, their results are not directly comparable with ours, since the datasets and the annotation schema (i.e., dependency types) are not the same.

The most closely related work to this work is [9]. In that work, the authors reported a preliminary investigation of the usefulness of distributed word representations in Vietnamese dependency parsing. Nevertheless, they only employed the distributed word representations produced by the Skip-gram model and experimented with some restricted feature sets. Therefore, their accuracy is modest compared to ours, 57.65% of F_1 score.

Our parsing system, including the parser, the dependency corpus and the distributed word representations, is released as an open-source toolkit and is freely available for research purpose.[7] We believe that this resource will be useful for the Vietnamese speech and language processing community.

5. CONCLUSION

In this paper, we have presented a dependency parser for Vietnamese which is the most accurate, compared to other dependency parsing systems on the same dependency treebank published so far, of 76.29% of UAS$_w$ and 69.25% of (LAS$_w$). This result is achieved by integrating distributed word representations into a statistical classifier, which is then used in the Nivre arc-eager parsing algorithm. We use two recent models to produce distributed representations, the Skip-gram model and the GloVe model.

We have also shown that the GloVe model is better than the Skip-gram model on producing word distributed representations in dependency parsing. Although in this work, we have investigated the usefulness of the distributed representations with Vietnamese only, we believe that they are also useful for dependency parsing of other languages. This is a direction of our future research.

We suppose that distributed word representations are not only helpful for transition-based dependency parsing, but they are also useful for other families of dependency parsers. In a future work, we plan to investigate the integration of distributed representations in graph-based dependency parsing and easy-first non-directional dependency parsing.

Finally, a recent research has shown that feature embeddings could be helpful to address the feature sparseness problem for dependency parsing [2]. We would like to apply the unsupervised learning models in feature space in a future work.

6. ACKNOWLEDGEMENTS

This research is partly funded by the FPT Technology Research Institute of FPT University. The last author is partly funded by the Vietnam National University, Hanoi (VNU) under project number QG.15.04. We are grateful to our three anonymous reviewers for their helpful comments.

7. REFERENCES

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[2] W. Chen, Y. Zhang, and M. Zhang. Feature embedding for dependency parsing. In *Proceedings of COLING*, pages 816–826, Dublin, Ireland, 2014.

[7]http://mim.hus.vnu.edu.vn/phuonglh/tools/VSDP.zip

[3] R. Collobert and J. Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167, New York, NY, USA, 2008.

[4] M. R. Costa-Jussà and M. Farrús. Statistical machine translation enhancements through linguistic levels: A survey. *ACM Computing Surveys (CSUR)*, 46(3):42, 2014.

[5] H. Cui, R. Sun, K. Li, M.-Y. Kan, and T.-S. Chua. Question answering passage retrieval using dependency relations. In *Proceedings of ACM SIGIR*, pages 400–407, Salvador, Brazil, 2005.

[6] Y. Goldberg and M. Elhadad. An efficient algorithm for easy-first non-directional dependency parsing. In *Proceedings of NAACL*, pages 742–750, Los Angeles, California, 2010.

[7] T. Koo, X. Carreras, and M. Collins. Simple semi-supervised dependency parsing. In *Proceedings of ACL-HLT*, pages 595–603, Columbus, Ohio, USA, 2008.

[8] S. Kübler, R. McDonald, and J. Nivre. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127, 2009.

[9] P. Le-Hong, T.-M.-H. Nguyen, T.-L. Nguyen, and M.-L. Ha. Fast dependency parsing using distributed word representations. In *Proceedings of PAKDD 2015 Workshops (LNAI)*. Springer, to appear, 2015.

[10] P. Le-Hong, T. M. H. Nguyen, A. Roussanaly, and T. V. Ho. A hybrid approach to word segmentation of Vietnamese texts. In *Proceedings of LATA, LNCS 5196*, pages 240–249. Springer, 2008.

[11] R. McDonald and J. Nivre. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230, 2011.

[12] R. McDonald and F. Pereira. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88, Trento, Italy, 2006.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*, Scottsdale, Arizona, USA, 2013.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[15] A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1081–1088. Curran Associates, Inc., 2009.

[16] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of AISTATS*, pages 246–252, Barbados, 2005.

[17] D. Q. Nguyen, D. Q. Nguyen, S. B. Pham, P. T. Nguyen, and M. L. Nguyen. 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, 2014.

[18] P. T. Nguyen, L. V. Xuan, T. M. H. Nguyen, V. H. Nguyen, and P. Le-Hong. Building a large syntactically-annotated corpus of Vietnamese. In *Proceedings of the 3rd Linguistic Annotation Workshop, ACL-IJCNLP*, pages 182–185, Suntec City, Singapore, 2009.

[19] T.-L. Nguyen, V.-H. Nguyen, T.-M.-H. Nguyen, and P. Le-Hong. Building a treebank for Vietnamese dependency parsing. In *Proceedings of RIVF*, pages 147–151. IEEE, 2013.

[20] J. Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 149–160, Nancy, France, 2003.

[21] J. Nivre, J. Hall, J. Nilsson, C. Atanas, G. Erigit, S. Kübler, S. Marinov, and E. Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13:95–135, 2007.

[22] J. Nivre and R. McDonald. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08*, pages 950–958, Columbus, Ohio, USA, 2008. ACL.

[23] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543, Doha, Qatar, 2014.

[24] K. Sagae and A. Lavie. A best-first probabilistic shift-reduce parser. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 691–698, Sydney, Australia, 2006.

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

[26] U. Yasavur, J. Travieso, C. Lisetti, and N. Rishe. Sentiment analysis using dependency trees and named-entities. In *Proceedings of the 27th International Flairs Conference*, pages 134–139, Pensacola Beach, Florida, 2014.