Improving Persian Dependency-Based Parser Using Deep Learning*

Research Article

Soghra Lazemi [1] Hossein Ebrahimpour-komleh[2] Nasser Noroozi[3]

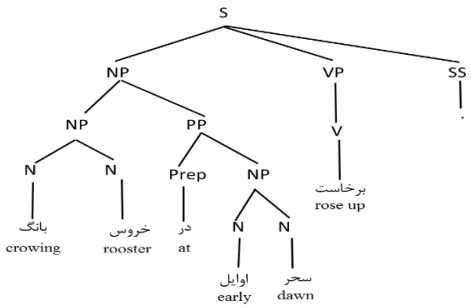
Abstract: One of the most important problems in computational linguistics is the grammar and, consequently, syntactic structures and structural parsing. The structural parser tries to analyze the relationships between words and to extract the syntactic structure of the sentence. The dependency-based structural parser is proper for free-word-order and morphologically-rich languages such as Persian. The data-driven dependency parser performs the categorization process based on a wide range of features, which, in addition to the problems such as sparsity and curse of dimensionality, it requires the correct selection of the features and proper setting of the parameters. The aim of this study is to obtain high performance with minimal feature engineering for dependency parsing of Persian sentences. In order to achieve this goal, the required features of the Maximum Spanning Tree Parser (MSTParser) are extracted with a Bidirectional Long Short-Term Memory (Bi-LSTM) Network and the edges of the dependency graph is scored by that. Experiments are conducted on the Persian Dependency Treebank (PerDT) and the Uppsala Persian Dependency Treebank (UPDT). The obtained results indicate that the definition of new features improves the performance of the dependency parser for Persian. The achieved unlabeled attachment scores for PerDT and UPDT are 90.53% and 87.02%, respectively.

Keywords: Dependency Parser, Data-Driven Parser, MSTParser, Phrase-structure Tree, Deep Learning, Persian

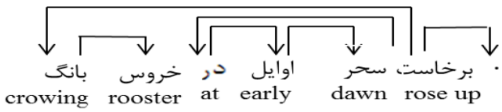
1. Introduction

Processing the natural language is performed at phonology, morphology (including lemmatization and stemming), lexical (including normalization, tokenization and spell checking), syntactic (including part of speech tagging (POS tagging) and structural parsing), semantic (including semantic role labeling and semantic parsing), discourse (including anaphora resolution and discourse/text structure recognition), and pragmatic levels [1]. The structural parsing is performed in the middle layer aiming at determining the grammatical role of words in the sentence. Although structural parsing is not the last step in the process of natural language, it is considered one of the most important steps, because it paves the ground for the next steps [2]. In the theory of knowledge graphs, words are represented by word graphs. Sentences are to be represented by sentence graphs. This is called structural or syntactic parsing [1].

In general, structural parsers are divided into two categories: phrase-structure parsers and dependency parsers [3]. Figure 1 shows a sample of phrase-structure-based parsing and dependency-based parsing and Figure 2 presents a simple categorization of parsers.



a- Phrase-structure-based parsing.



b- Dependency-based parsing.

Figure 1. An example of syntactic parsing

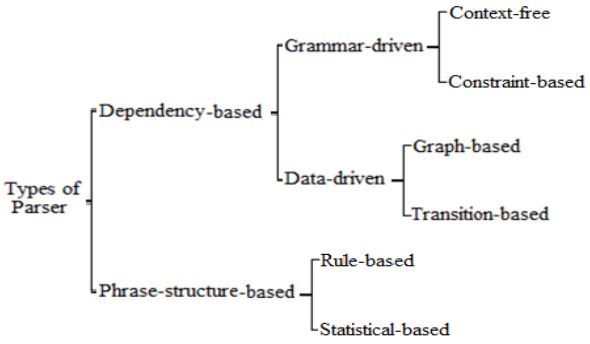


Figure 2. Taxonomy of parser

Phrase-structure parsers: In this type of parser, based on structural grammar, the sentence is parsed into smaller phrases reciprocally, as a first-level phrase [4]. The limitations of this type of parser are as follows:

1. Inflexibility respect to words order: in some languages

* Manuscript received: August, 18, 2019; Revised: December, 22, 2020, Accepted: October,19, 2021.

[1] Corresponding author. Ph.D. Department of Computer Eng., Faculty of Electrical & Computer Eng., The University of Kashan, Kashan, Iran. **Email:** soghra.lazemi@gmail.com.

[2] Assistants Professor, Department of Computer Eng., Faculty of Electrical & Computer Eng., University of Kashan, Kashan, Iran.

[3] Assistants Professor Faculty of Mathematics, University of Kashan, Kashan, Iran.

(for example Czech, Persian), the words order is free;
2. Language-dependent: the need to new rules set for each language;
3. Syntax-based: does not include conceptual information like conceptual rules.

As depicted in Figure 2, the phrase-structure parser is divided into two categories: rule-based and statistical-based [4]. In the rule-based parsing, a number of rules are defined by linguists based on the grammar for parsing the sentences. Limitations of this method are impossibility of providing complete rules with high coverage power, especially for free word order languages, being time-consuming, and impossibility of parsing for sentences outside the defined rules [4]. Statistical-based parser tries to extract grammar automatically using statistical techniques and linguistic corpora. The problem of this method is the requirement for annotated Treebank [5].

Dependency parsers: In this kind of parsing, it is assumed that syntactical structures include lexical components that are related to each other by asymmetric relations [6]. Based on the dependency grammar, this parser determines the syntactic structure of each sentence by examining the relationship between the head and dependent. Dependency parsing is divided into two categories [6]:

1. Data-driven parsing: In this kind of parsing, a parametric model is created where the model parameters are calculated using the methods of machine learning from the training data. This kind of parsing has been considered as the most important type due to the lack of language dependency and the availability of appropriate data. This method is divided into the general categories of graph-based and transition-based [7]. In the graph-based methodology, the graph theory is used to construct the parsing tree of the input sentence. This method first draws a complete graph for the input sentence and then tries to obtain the graph with the highest score. Dependency graphs can be divided into projective (a word along with its dependents are seen as a substring of the sentence [8] and the dependency graph lacks an intersecting edge (overlapping edge)), non-projective (having at least two overlapping edges) or well-constructed (that has root, unique labels, acyclic, connected and projective) types. In Persian, the projective trees cover more sentences due to the free word-ordering property [9]. The transition-based approach involves a number of configurations and transitions between them (similar to the Turing machine in computation theory). In the algorithms of this method, the parsing operation begins with the initial configuration and proceeds to the next configuration by selecting the transition based on the existing features. This continues until it reaches a final configuration [10]. Graph-based parsing is a global parsing, i.e., all words of the input sentence are analyzed to achieve the dependency tree, while in the transition-based methods a small part of the input sentence is analyzed based on features extracted from the configuration. Transition-based methods are of local parsing type.
2. Grammar-driven parsing: Grammar-driven parsing is similar to the data-driven method, with the difference that in this method some grammar is defined, based on grammatical system [11]. Grammar-driven method is divided into two categories too: context-free and constraint-based. In context-free method, dependency structure is converted to context-free expression structure and uses current algorithms in that field. In constraint-based method, the problem is converted to constraint satisfaction problem and the grammar is determined as a set of dependency graph building constraints. In this case, the aim is to find a dependency graph that satisfies all the constraints of grammar [11].

The most significant difference between data-driven methods with grammar-driven methods is that a data-driven approach produces outputs for the incorrect sentences, while in a grammar-driven approach no parsing tree is made for the sentences that are not part of the language. However, in some languages, the use of a data-driven parser requires proper selection of features and proper adjustment of parameters to achieve maximum performance [12].

Phrase-structure parsers are not flexible in terms of words and by changing the order of words, create another parsing tree; therefore, they cannot be efficient for the Persian language. On the other hand, due to its simplicity, the dependency parser has a higher ability to be learned by machine and humans [13]. The focus of this research is on developing a graph-based dependency parser for Persian.

According to previous research [9] and considering specific features of Persian language, MSTParser (Maximum Spanning Tree) [14] is suitable for this language, for the following reasons:

1. Due to the SOV (Subject-Object-Verb) property, the head and the dependent are usually spaced far from each other, and MSTParser is appropriate to determine long-distance relationships due to the creation of a sentence graph;
2. Due to the free word-order property, most Persian sentences produce non-projective trees, and MSTParser is able to produce non-projective trees.

The rest of the paper is organized as follows. In the second part, the conducted researches on Persian parsers are discussed. In the third section, our method is presented. The fourth section includes experiments and results. Finally, in the fifth section, a summary of the study is presented with some future works.

2. Review of related works

In spite of the fact that dependency parsing has a long history and diverse researches in other languages (especially English) have been done, it doesn't have very long history in Persian and is limited to a number of novice researches and is based on traditional methods (most of them have focused on the compatibility of existing tools for Persian). In our opinion, this occurs because Persian belongs to a low-resource language group. The lack or shortage of data resources has reduced the willingness of researchers to work in the Persian domain.

2.1. Phrase-structure parser

Estiri et al. [15] provided the first rule-based phrase-structure parser. After performing preprocesses (unification and removing the short space), their tried to identify sentences (using punctuation and Persian language grammar) and words (using space and punctuation). In the next step, the initial tag of words is obtained through a search in a provided database and defined rules. In the final step, by combining

the tags and formation the groups, a general tag or identical tag is assigned to several successive words. Finally, by assigning labels to all groups and words, the parsing tree is obtained. Due to the lack of similar parser for Persian, the comparison and evaluation has not been done. One of the disadvantages of the parser can be the inability to deal with unstructured and non-grammatical sentences.

2.2. Dependency parser

Researches in this area are limited to testing the existing state-of-the-art parsers and choosing the proper feature sets for Persian. Seraji et al. [16] compared the state-of-the-art graph-based and transition-based parsers like MaltParser [17], MSTParser [18], MateParser [19, 20] and TurboParser [21] for Persian. They find the best POSTag-set and dependency relationships using Maltparser, and thereafter, perform their experiments with other parsers. According to their experiments, the graph-based parser provides better results.

Falavarjani and Ghassem-Sani [9] tested two state-of-the-art parsers, MaltParser (transition-based) and MSTParser (graph-based) for Persian. They tested two parsers to find the appropriate parser for projective and non-projective sentences. According to their experiments, the graph-based parser provided better results. In another study, Seraji et al. [22] tested two other state-of-the-art parsers, MaltParser and MSTParser, using the UPEDT corpus for Persian with different settings. According to their experiments, arc-eager algorithm is selected with the gold standard POS features for MaltParser and second order feature and projective setting is selected for MSTParser.

Kalash et al. [13] analyzed the effects of lexical and morphological information on MSTParser and MaltParser. They tried to investigate the gold and automatic features and to choose the best feature set. MaltOptimizer was used to optimize MaltParser. Second order feature and non-projective settings have been selected for the MSTParser.

Lazemi and Ebrahimpour-Komleh [23] tried to improve the MSTParser. They defined 21 new semantic and structural features and added them to the MSTParser by the stacking method. Semantic features were obtained using word clustering algorithms based on syntagmatic analysis, and the syntactic features were obtained using the Persian phrase-structure parser, and were used as bit-string.

3. Proposed method

Almost all dependency parsers perform the categorization process based on millions of features, which sparsity is one of their main consequence. Not only these features are not very powerful, but also the extraction of these features and their computations is time consuming. In this study, the required features for the MSTParser were extracted from a deep model, which will be discussed as follow.

The graph-based dependency parsing consists of three main stages (definition of sentence space, learning, and parsing). In the first step, a space of candidate dependency directed graphs is created for the sentence. In the learning phase, a model for scoring is determined for the dependency graph of the sentence. The parsing stage seeks to find the highest-score dependency graph (to resolve the ambiguity). There are different algorithms for graph-based parsing, and the most notable one is the arc-factored model. In the arc-factored model, the dependency graph is divided into several sub-graphs, P_1, P_2, ..., P_n; each sub-graph is individually scored and the score of the graph is considered as the total score of the sub-graphs (Equation 1). In this model, each edge is represented by a feature vector and at the learning stage, each feature is weighted.

$$score(G) = score(P_1) + \cdots + score(P_n) \quad (1)$$

MSTParser is a member of graph-based dependency parsers. For a given sentence $X = (x_1, x_2, \dots, x_n)$, a dependency graph is defined as $G_X = (V_X, E_X)$ such that V_X is a set of word (or token) in X along with a root (x_0), and E_X is a set of labeled dependency relations between elements in V_X. During the parsing stage, MSTParser first builds a complete directed graph; for each pair of (x_i, x_j) where $x_i \neq x_j$, there exists a directed edge from x_i to x_j. The score of an edge is defined as the weighted sum of all its features, (Equation 2).

$$score(x_i, x_j) = f_1 w_1 + \cdots + f_n w_n \quad (2)$$

w is the weight vector, which is calculated using the machine learning algorithms from the training samples (pairs of sentences and the corresponding dependency trees).

Assume that $T(G_X)$ is the spanning tree set of the G_X. In MSTParser, finding a dependency graph with the highest score converts to the determining of the spanning tree $\acute{G}$ with the highest weight (Equation 3):

$$ParseTree(X) = \arg \max_{\acute{G} \in T(G_X)} score_{global}(X, \acute{G}) \quad (3)$$

that the dependency tree score equals to the sum of edge scores (Equation 4).

$$ParseTree(X) = \arg \max_{\acute{G} \in T(G_X)} \sum_{(x_i, x_j) \in \acute{G}} score(x_i, x_j) \quad (4)$$

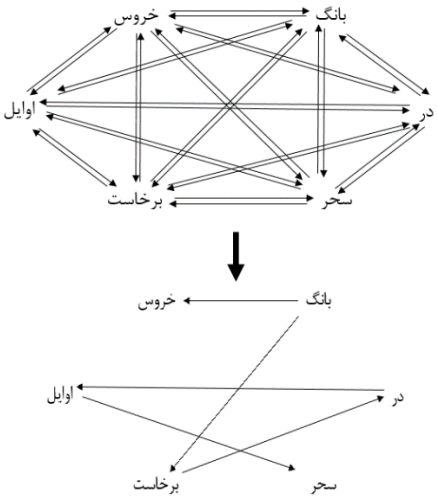


Figure 3. Maximum spanning tree derived from a complete directed graph

Figure 3 shows the maximum spanning tree derived from a complete directed graph for a sentence:

«بانگ خروس در اوایل سحر برخاست».

The default feature set that has been used in MSTParser is summarized as follows, used as uni-gram and bi-gram:

1. POS tags of the words H_i and D_j and the label L_K
2. POS tags of words surrounding and between H_i and D_j
3. Number of words between H_i and D_j and their orientation
4. Label features

Despite the traditional MSTParser method, in which each edge score is obtained through weighted sum of features, our proposed method extracts the score of each edge in three steps:

1. A Bi-LSTM network that receives the distributive representation of words and their lexical-morphological information in vector space as inputs.

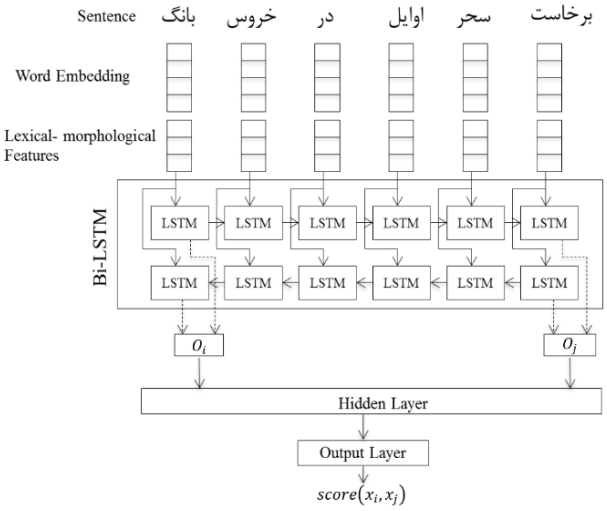


Figure 4. An example of score computing for each pair of (x_i, x_j) using our proposed method from sentence shown in Figure 2

2. A hidden layer with a rectified activation function combining the output features of each pair of words (x_i, x_j).
3. An output layer for generating scores for each kind of dependency type between each pair of words (x_i, x_j).

Each step will be described in details. Neighborhood information is very useful in graph-based structure parsing [14, 18], so we use the LSTM to extract foundation information. The LSTM network allows the use of information far from the current word. The bidirectional network allows us to access the information on the right and left side of the words. In other words, by using the bi-directional network, we can obtain the information of surrounding neighbors and neighbors between head and dependent. To achieve this, at first, each word is represented by word embedding with a d-dimensional vector, $e_i^w \in \Re^d$. The generated vector for each word is used as an input for the Bi-LSTM network.

In morphologically-rich languages, the data on lexical features are also introduced as useful features. Therefore, we enter these features into the network as the binary features.

In Persian, the verb corresponds to the subject in terms of person and number [24]. The use of these two features can be useful in determining some roles such as subject in sentences containing more than one word with the noun tag.

In some special cases, some words are created by the combination of several components that each of them is attached to the previous or next component [24]. The relationship between these components should be defined in the dependency parsing in order to extract the relation between the components of the sentence.

Table 1 shows different morphological categories and the possible values of them. Given that word order in Persian sentences is relatively free, morphological information is important to determine dependency relation.

The concatenation of generated word embedding vector, POS tag and extracted morphological features of each word are fed into a forward LSTM network and a backward LSTM network. The concatenation of the output of two words (O_i, O_j) goes into the hidden layer, including d_H number of nodes and the rectified activation function (Equation 5).

$$H = g(W_1(O_i \oplus O_j) + b_1) \quad (5)$$

where $\oplus$ denotes the concatenation process.

The output score (Equation 6), $score(x_i, x_j) \in R^{|L|}$, is a score vector where |L| is the number of dependency types and each dimension of $score(x_i, x_j)$ is the score for each kind of dependency type of a pair of words (x_i, x_j). The maximum value is used in Equation 2 as $score(x_i, x_j)$.

$$score(x_i, x_j) = Softmax(W_2 H + b_2) \quad (6)$$

Figure 4 illustrates summarily the architecture of our approach. Given a sentence, we first get representation for each word. Word representation captures the features locally embedded around the word. The features used in our work are the word embedding and its lexical-morphological information. All these features are concatenated. After concatenation, we get the word representation feature vector as input of network. Then, bidirectional LSTM RNN layer is designed to combine the local information of a word and its contextual information from both directions. For each word, x_i, there is an output vector, O_i, the concatenation of the two output vectors (O_i, O_j) is fed into a hidden layer to combine them, and finally an output layer is deigned to get the output. The output of network is a vector, which each dimension is a score corresponding to a kind of dependency type of a pair of input words (x_i, x_j). The maximum value is selected as $score(x_i, x_j)$ used in MSTParser.

4. Experiments and results

4.1. Corpora and evaluation metrics

Having about 110 million speakers around the world, Persian language is one of the poorest languages in terms of availability of annotated corpora. To conduct experiments on dependency parsing of sentences, there are only two corpora (Persian Dependency Treebank (PerDT) [25] and Uppsala Persian Dependency Treebank (UPDT) [26]) that we have used both. PerDT is the first Persian dependency Treebank, and includes about 30,000 sentences annotated with syntactic roles and morpho-syntactic features and the corresponding dependency tree. There are 44 dependency

relations, 17 types of coarse-grained, and 32 types of fine-grained POS tags. In UPDT, the syntactic relation of words is determined by the dependency grammar. This corpus contains 6000 sentences from the Uppsala Persian Corpus with a corresponding dependency tree. In this corpus, there are 48 types of dependency relations, 13 types of coarse-grained, and 18 types of fine-grained POS tags. Both corpora have been prepared based on the CoNll template and the Stanford Typed. More information about the used corpora is given in Table 2.

Tense, mood, and aspect are not separately annotated in the PerDT treebank, so we used them as one feature. Types are not available in the first corpus. In the second corpus, morphological features are limited and represented as a single atomic feature. The available morphological features in the second corpus are tense, mood, type, and number.

For evaluation, the Unlabeled Attachment Score and the Labeled Attachment Score are used and defined as Equations 7 and 8.

$$UAS = \frac{Number\ of\ identical\ edges\ in\ two\ trees\ regardless\ of\ the\ label}{Total\ number\ of\ two\ tree\ edges} \quad (7)$$

$$LAS = \frac{Number\ of\ identical\ edges\ in\ two\ trees\ with\ label}{Total\ number\ of\ two\ tree\ edges} \quad (8)$$

Table 1. Description of morphological features, possible values and compatible POS tags

Category	Values	Compatible PoS Tags
Number	Singular, Plural, Dual	Noun, Pronoun, Verb
Person	1st, 2nd, 3rd	Verb
Tense	Present, Past, Future	Verb
Mood	Indicative, Subjunctive,Imperative	Verb
Attachment	Isolated word, Attached to the next word, Attached to the previous word	All
Type	Cardinal, Ordinal, Multiple	Numeral
Polarity	Positive, Negative	Verb

Table 2. Statistical properties of PerDT corpus and UPDT corpus

	Persian Dependency Treebank	Uppsala Persian Dependency Treebank
Number of sentences	29982	6000
Number of words	498081	151671
Number of distinct words	37618	15692
Number of verbs	9200	-
Number of distinct verbs	62889	-
Average sentence length	16.61	25.28
Coarse-grained POS tags	17	13
Fine-grained POS tags	32	18
	available in CoNLL format	available in CoNLL format

4.2. Hyper-parameter tuning

We implemented the neural network using the UKPLab[1]. Training was performed with mini-batch stochastic gradient descent (SGD) with a fixed learning rate. Moreover, we explored AdaGrad, AdaDelta, RMSProp, Adam and Nadam optimization algorithms, but they did not improve upon SGD.

In order to reduce overfitting, we applied the dropout method [27]. We applied dropout on output vector of each LSTM layer. In order to create word embeddings, we used the word2vec [28] algorithm. We tuned the hyper-parameters on the development sets by random search and evaluated 300 hyper-parameters setting. Table 3 summarizes the chosen hyper-parameters for all experiments and Table 4 shows the development set's performance of the best setting by projective-first order setting.

Table 3. Hyper-parameters values for experiments

Hyper-parameter	
Word embedding dimension	50
Morphological feature dimension	20
POS tag dimension for PerDT	49
POS tag dimension for UPDT	31
LSTM layers	2
LSTM state size	100
Learning rate	0.0005
Dropout rate	0.5
Mini-batch size	10
Number of neurons	100

Table 4. Results on development set

	PerDT		UPDT	
	UAS	LAS	UAS	LAS
Our Method with Gold POS tag	86.03	83.14	80.11	76.89

4.3. Results and discussion

The data is split into standard train, development and test sets by the ratio of 80%-10%-10% in the CoNLL dependency format.

MSTParser is used with four settings: projective-first order, projective-second order, non-projective-first order, and non-projective-second order. MIRA is used to estimate the weight vector. Experiments were done with Gold POS Tag and Predicated POS Tag. To generate POS tags of the words in the first corpus, Parsipardaz toolkit [29] was used, which has been provided by Sarabi et al. [29]. That's because, the Parsipardaz-POSTagger's tag set is the same as the tag set of the first corpus. The accuracy of Parsipardaz-POSTagger has been reported to be about 98.5%. For the second corpus, the TagPer tool [30] provided by Seraji et al.

[1]. https://github.com/UKPLab/emnlp2017-bilstm-cnn-crf

[29] was used. The accuracy of TagPer has been reported to be about 97.8% for the UPEC (Uppsala Persian Corpus) corpus. POSTaggers were applied to the test data sets.

Tables 5, 6, 7, and 8 illustrate the results for both corpora. For comparisons, the tests were implemented in four modes: Baseline-features with Gold Pos tags, Baseline-features with Predicated Pos tags, our proposed method with Gold Pos tags and our proposed method with Predicated Pos tags. According to the tables, the use of Gold Pos tags outperform the Predicated Pos tags in both the baseline mode and our proposed approach.

The results indicate that our proposed method has better performance. Our suggested method, although does not indicate any significant improvement in comparison with our previous research, was able to achieve some good results in some cases.

By comparing Baseline-features and our proposed method, it can be seen that neural models are able to obtain precise features which are difficult to gain using manual feature selection methods.

Table 5. Projective-first order

	PerDT		UPDT	
	UAS	LAS	UAS	LAS
Baseline-Features with Gold POS tag	80.21	77.38	77.69	71.13
Baseline-Features with Predicated POS tag	78.85	75.89	76.46	70.01
Our Method with Gold POS tag	84.29	81.42	77.52	75.10
Our Method with Predicated POS tag	83.01	80.17	75.39	73.94

Table 6. Non projective-first order

	PerDT		UPDT	
	UAS	LAS	UAS	LAS
Baseline-Features with Gold POS tag	82.12	79.81	83.81	76.45
Baseline-Features with Predicated POS tag	80.63	77.02	81.03	74.11
Our Method with Gold POS tag	85.96	83.25	83.94	82.05
Our Method with Predicated POS tag	84.33	82.22	81.00	79.88

Table 7. Projective-Second Order

	PerDT		UPDT	
	UAS	LAS	UAS	LAS
Baseline-Features with Gold POS tag	80.93	78.13	79.86	74.60
Baseline-Features with Predicated POS tag	78.88	76.33	76.92	72.22
Our Method with Gold POS tag	88.03	85.11	82.21	79.39
Our Method with Predicated POS tag	86.40	84.02	79.97	77.64

As it is evident, non-projective settings for both corpora have brought good results.

Based on Tables 5, 6, 7, and 8, our improved MSTParser outperforms its baseline for all settings.

Table 8. Non Projective-Second Order

	PerDT		UPDT	
	UAS	LAS	UAS	LAS
Baseline-Features with Gold POS tag	84.79	82.06	85.02	83.26
Baseline-Features with Predicated POS tag	84.06	80.55	83.41	82.25
Our Method with Gold POS tag	90.53	88.41	87.02	85.78
Our Method with Predicated POS tag	87.74	85.36	85.10	83.99

Table 9. Comparison of results

	PerDT		UPDT	
	UAS	LAS	UAS	LAS
Lazemi et al. [23]	89.17	85.83	88.96	86.25
Our Proposed Method	92.08	89.11	91.28	87.69

In our previous work [23], we faced with the sparsity problem, so to reduce the problem, the sentences that contain full-frequent words of corpora (PerDT and UPDT) were selected and experiments were done by them. In order to make a comparison, we repeated experiments with those data. Table 8 demonstrates the obtained results for non-projective-second order setting. It can be seen that our proposed method has better results. On the other hand, the use of the distributive representation of words solved the sparsity problem.

5. Conclusion

This study introduced a deep learning based approach for improving MSTParser for Persian sentences. In the proposed method, the words were first transmitted to the vector space, and then by using the Bi-LSTM network, the information of the neighbors of the words were extracted and the contextual representation of the words was obtained. The extracted features were fed as inputs to the neural network and, for each pair of words, the score of each syntactic class was calculated. The results of the experiments reveal the appropriateness and efficiency of the proposed method.

Unfortunately, in the area of dependency parsing, there is a huge gap between performed researches for Persian and other languages; the main reason for this is the lack of labeled corpora and tools. Since Persian belongs to the morphologically-rich group of languages, it is recommended that in future studies, large corpora of lexical and morphological information and created, and some tools be created that are able to extract more morphological data from words. On the other hand, since Persian is a generative language, it is likely that we encounter many words not included in the dictionary. Dealing with this issue can also increase the speed and accuracy of the parser.

6. References

[1] Liddy, E. D., "Natural language processing", 2001.

[2] Nadkarni, P. M., Ohno-Machado, L., and Chapman, W. W., "Natural language processing: an introduction", *Journal of the American Medical Informatics Association*, vol. 18 No, 5, pp. 544-551, 2011.

[3] Jurafsky, D., and James, H., "Speech and language processing an introduction to natural language processing", computational linguistics, and speech, 2000.

[4] Sakaguchi, K., and Nagata, R., "Phrase structure annotation and parsing for learner English. Information and Media Technologies", vol. 12, pp. 316-339, 2017.

[5] Khatun, A., and Hoque, M. M., "Statistical parsing of Bangla sentences by CYK algorithm", *International Conference on Electrical, Computer and Communication Engineering,*, pp. 655-661, 2017.

[6] Nivre, J., "Dependency grammar and dependency parsing", MSI report, pp. 1-32, 2005.

[7] Zhang, X., Cheng, J., and Lapata, M., "Dependency parsing as head selection", arXiv preprint arXiv:1606.01280, 2016.

[8] Grella, M., "Notes About a More Aware Dependency Parser", arXiv preprint arXiv:1507.05630, 2015.

[9] Falavarjani, S. A. M., and Ghassem-Sani, G., "Advantages of dependency parsing for free word order natural languages", *International Conference on Current Trends in Theory and Practice of Informatics*, January 24, 2015, pp. 511-518, 2015.

[10] Dyer, C., Ballesteros, M., Ling, W., Matthews, A., Smith, NA., "Transition-based dependency parsing with stack long short-term memory", arXiv preprint arXiv:1505.08075, 2015.

[11] Kübler, S., McDonald, R., and Nivre, J., "Dependency parsing", *Synthesis Lectures on Human Language Technologies*, vol. 1, pp. 1-127, 2009.

[12] Plank, B., and Van Noord, G., "Grammar-driven versus data-driven: which parsing system is more affected by domain shifts?", *Proceedings of the 2010 Workshop on NLP and Linguistics: Finding the common ground*, pp. 25-33, 2010.

[13] Khallash, M., Hadian, A., and Minaei-Bidgoli, B., "An empirical study on the effect of morphological and lexical features in Persian dependency parsing", Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages. pp 97-107, 2013.

[14] McDonald, R., Crammer, K., and Pereira, F. C., "Spanning tree methods for discriminative training of dependency parsers", *Technical Reports (CIS)*, pp. 1-55, 2006.

[15] Estiri, A., Kahani, M., Hoseini, M., and Asgarian, E., "Designing Persian language parser tool", *International Conference on Asian Language Processing*, 2012.

[16] Seraji, M., Bernd, B., and Nivre, J., "ParsPer: A dependency parser for Persian", *International Conference on Dependency Linguistics (DepLing 2015)*, August 24-26, 2015, Uppsala, Sweden, pp. 300-309, 2015.

[17] Nivre, J., Hall, J., and Nilsson, J., "Maltparser: A data-driven parser-generator for dependency parsing", *Proceedings of Language Resources and Evaluation Conference*, pp. 2216-2219, 2006.

[18] McDonald, R., Pereira, F., Ribarov, K., and Hajič, J., "Non-projective dependency parsing using spanning tree algorithms", *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 523-530, 2005.

[19] Bohnet, B., and Kuhn, J., "The best of both worlds: a graph-based completion model for transition-based parsers", *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 77-87, 2012.

[20] Bohnet, B., Nivre, J., "A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing", *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1455-1465, 2012.

[21] Martins, A. F., Smith, N. A., Xing, E. P., Aguiar, P. M., Figueiredo, M. A., "Turbo parsers: dependency parsing by approximate variational inference", *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 34-44, 2012.

[22] Seraji, M., Megyesi, B., and Nivre, J., "Dependency parsers for Persian", *24th International Conference on Computational Linguistics*, 8-15 pp. 35-44, 2012.

[23] Lazemi, S., Ebrahimpour-Komleh, H., "Feature engineering in Persian dependency parser", *Journal of AI and Data Mining*, vol. 7 No.30, pp. 467-474, 2018.

[24] Shamsfard, M., "Challenges and open problems in Persian text processing", *Proceedings of LTC*, vol. 11, pp.65-69, 2011.

[25] Rasooli, M. S., Kouhestani, M., and Moloodi, A., "Development of a Persian syntactic dependency treebank", *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 306-314, 2013.

[26] Seraji, M., Jahani, C., Megyesi, B., and Nivre, J., "A Persian treebank with Stanford typed dependencies", *Proceedings of Language Resources and Evaluation Conference*, Reykjavik, Iceland, pp. 796-801, 2014.

[27] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R., "Dropout: a simple way to prevent neural networks from overfitting", *The journal of machine learning research*, vol. 15, No. 1, pp. 1929-1958, 2014.

[28] Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T., "Learning word vectors for 157 languages. ", arXiv preprint arXiv:1802.06893, 2018.

[29] Sarabi, Z., Mahyar, H., and Farhoodi, M., "ParsiPardaz: Persian language processing toolkit", *Computer and Knowledge Engineering*, pp. 73-79, IEEE, 2013.

[30] Seraji, M., Megyesi, B., and Nivre, J., "A basic language resource kit for Persian", *Eight International Conference on Language Resources and Evaluation*, pp. 2245-2252, 2012.