OPEN ACCESS
Indo-JC
Indonesian Journal on Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

Ind. Journal on Computing
Vol. 5, Issue. 3, December 2020. pp. 15-22
doi:10.34818/indojc.2021.5.3.504

Implementation of Dependency Parser Using Artificial Neural Network Methods

Nurul Izzah #1, Moch Arif Bijaksana #2, Arief Fatchul Huda *3

School of Computing, Telkom University
Jl. Telekomunikasi Terusan Buah Batu Bandung 40257

[1] nurulizzah@students.telkomuniversity.ac.id

[2] arifbijaksana@telkomuniversity.ac.id

* *School of Science and Technology, UIN Sunan Gunung Djati*
Jl. A.H. Nasution No. 105 Cibiru Bandung 40614

[3] afhuda@uinsgd.ac.id

Abstract

In recent years, parsing has become very popular within the scope of NLP (Natural Language Processing) with the presence of Dependency Parser. However, almost all existing Dependency Parser do classifications based on millions of sparse indicator features. This feature is not only bad in drawing conclusions, but also significantly limits the speed of parsing so that the resulting parsing is not optimal. To overcome these problems, changing the use of sparse features becomes dense features to reduce sparsity between words. The Artificial Neural Network classification method is used to produce fast and concise parsing in the Transition-Based Dependency Parser by using 2 hyperparameters. The dataset used in this study is Arabic, Chinese, English, and Indonesian. Based on the evaluation that has been done, it shows a higher result using the second hyperparameter. In testing with English test data, the accuracy value of LAS (Labeled Attachment Score) is 80.4% and UAS (Unlabelled Attachment Score) is 83%, Then with dev data obtained an accuracy value of LAS 81.1% and UAS 83.7%, and parsing speed of 98 sentences per second (sent/s).

Keywords: Parsing, dependency parser, transition-based dependency parser.

Abstrak

Beberapa tahun terakhir, *parsing* menjadi sangat populer dalam ruang lingkup NLP *(Natural Language Processing)* dengan adanya *Dependency Parser*. Namun, hampir semua *Dependency Parser* yang ada melakukan klasifikasi berdasarkan jutaan fitur indikator jarang *(sparse)*. Fitur ini tidak hanya buruk dalam mengambil kesimpulan, tetapi juga membatasi kecepatan parsing secara signifikan sehingga parsing yang dihasilkan tidak maksimal. Untuk mengatasi permasalahan tersebut, dilakukan pergantian penggunaan fitur jarang *(sparse)* menjadi fitur padat *(dense)*. Dengan penggunaan fitur padat dapat secara efektif mengurangi *sparsity* antar kata. Metode klasifikasi Jaringan Saraf Tiruan digunakan untuk menghasilkan *parsing* yang cepat dan ringkas dalam *Transition-Based Dependency Parser* dengan menggunakan 2 *hyperparameter*. Dataset yang digunakan pada penelitian ini yaitu Bahasa Arab, China, Inggris, dan Indonesia. Berdasarkan evaluasi yang telah dilakukan, menunjukkan hasil yang lebih tinggi dengan menggunakan *hyperparameter* kedua. Pada pengujian dengan data *test* Bahasa Inggris diperoleh nilai akurasi LAS *(Labelled Attachment Score)* 80.4% dan UAS *(Unlabelled Attachment Score)* 83%. Kemudian dengan data *dev* diperoleh nilai akurasi LAS 81.1% dan UAS 83.7%, serta kecepatan *parsing* sebesar 98 kalimat per detik *(sent/s)*.

Kata Kunci: Parsing, dependency parser, transition-based dependency parsing.

Received on xxx, accepted on xxx, published on xxx

I. Introduction

In recent years, parsing has become very popular within the scope of NLP (Natural Language Processing) with the presence of Dependency Parser [9]. Parsing is an important step in several applications that involve document analysis, such as knowledge extraction, question answering, summarization, or filtering [1]. In general, NLP uses the results obtained from parsing to be the basis for further processing, and accuracy in parsing will increase system accuracy [10]. Dependency parsing is an approach used to carry out automatic syntactic analysis of natural languages inspired by dependency grammar. The main purpose of dependency parsing is to perform dependency structure analysis automatically from a given input sentence [9].

Lots of previous research has produced parser, such as PCFG Parser [7], Malt Parser [13], Arabic Parser [4], Chinese Parser [14], and others. However, almost all existing Dependency Parser do classifications based on millions of sparse indicator features. This feature is not only bad in drawing conclusions, but also significantly limits the speed of parsing so that the resulting parsing is not optimal. To overcome these problems, changing the use of sparse features becomes dense features. Using dense features can effectively reduce sparsity between words, and get a good starting point for building word features and their interactions [2]. In this study, the authors used the Artificial Neural Network classification method in the Transition-Based Dependency Parser to make parsing decisions. The Artificial Neural Network classification method only studies and uses a small number of dense features, so this method can work quickly and concisely. This method can also learn POS (Part of Speech) Tagging, and Dependency Relations. In doing parsing, the writer uses Dependency Parser which applies the Artificial Neural Network method based on previous research, namely A Fast and Accurate Dependency Parser using Neural Networks [2]. The aim of this research is to produce a quick and concise classification, both in terms of accuracy of the LAS (Labeled Attachment Score) and UAS (Unlabelled Attachment Score) and the parsing speed by using the Transition-Based Dependency Parser.

Inputs from the system are Arabic, Chinese, English, and Indonesian datasets using CONLL format which has been separated into 3 sections based on PTB3 standard separator rules, namely train, dev, and test. The output from the system is the LAS and UAS values and the parsing speed obtained from the parsing results with Dependency Parser. Parsing is done using the python programming language. The purpose of this study was to parse using 2 hyperparameters in Arabic, Chinese, English, and Indonesian datasets, then calculate the accuracy values of LAS and UAS and the parsing speed generated by the Transition-Based Dependency Parser.

II. Literature Review

A. Dependency Parsing

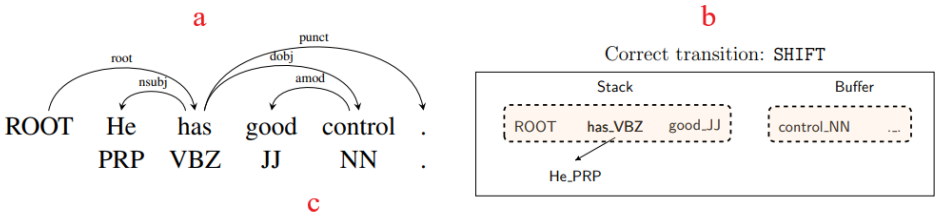
Parsing is an important step in several applications that involve document analysis, such as knowledge extraction, question answering, summarization, or filtering [1]. In general, NLP uses the results obtained from parsing to be the basis for further processing, and accuracy in parsing will increase system accuracy [10]. Dependency parsing is an approach used to carry out automatic syntactic analysis of natural languages inspired by dependency grammar. The main purpose of dependency parsing is to perform dependency structure analysis automatically from a given input sentence [9]. Dependency parsing has 2 methods, namely data-driven dependency parsing and grammar-based parsing. There are 2 classes in data-driven dependency parsing including transition-based dependency parsing and graph-based dependency parsing.

Transition-based dependency parsing or commonly called shift-reduce dependency parsing aims to build an optimal transition sequence for input sentences and be converted to dependency graphs. Graph-based dependency parsing or commonly called maximum spanning tree parsing aims to search for dependency parsing through possible tree spaces for sentences by maximizing the score [6]. In this study, the author used transition-based dependency parsing to do the parsing.

B. Transition-Based Dependency Parsing

Transition-based dependency parsing is one of the methods of data-driven dependency parsing which aims to predict the order of the transition from initial configuration to several terminal configurations to obtain the results of parsing in the form of dependency trees [2], as shown in Fig 1. In this study, the authors used the arc-standard system [12]. The arc-standard system is one of the most popular transition systems used today, this system has 3 types of transitions [6]:

- LEFT-ARC: States the head-dependent relationship between the word at the top of the stack and the word directly below it, delete the word that is lower than the stack.
- RIGHT-ARC: States the head-dependent relationship between the second word in the stack and the word above, delete the word at the top of the stack.
- SHIFT: Remove the word from the front of the input buffer and push it to the stack.



c

Transition	Stack	Buffer	A
	[ROOT]	[He has good control .]	$\emptyset$
SHIFT	[ROOT He]	[has good control .]	
SHIFT	[ROOT He has]	[good control .]	
LEFT-ARC(nsubj)	[ROOT has]	[good control .]	$A \cup$ nsubj(has,He)
SHIFT	[ROOT has good]	[control .]	
SHIFT	[ROOT has good control]	[.]	
LEFT-ARC(amod)	[ROOT has control]	[.]	$A \cup$amod(control,good)
RIGHT-ARC(dobj)	[ROOT has]	[.]	$A \cup$ dobj(has,control)
...	...	...	...
RIGHT-ARC(root)	[ROOT]	[]	$A \cup$ root(ROOT,has)

Figure 1. An example of transition-based dependency parsing. a: the desired dependency tree, b: an intermediate configuration, c: a transition sequence of the arc-standard system.

Figure 2 is an example of a feature template taken from previous research [2]. In that study, it was explained that the features in Figure 2 had several problems, namely sparsity, incompleteness [8], expensive feature computation [5]. In this study the authors only focus on issues related to sparsity. By using the Artificial Neural Network classification method, it can learn and use a small number of dense features instead of sparse features.

Single-word features (9)
$s_1.w$; $s_1.t$; $s_1.wt$; $s_2.w$; $s_2.t$;
$s_2.wt$; $b_1.w$; $b_1.t$; $b_1.wt$
Word-pair features (8)
$s_1.wt \circ s_2.wt$; $s_1.wt \circ s_2.w$; $s_1.wts_2.t$;
$s_1.w \circ s_2.wt$; $s_1.t \circ s_2.wt$; $s_1.w \circ s_2.w$
$s_1.t \circ s_2.t$; $s_1.t \circ b_1.t$
Three-word feaures (8)
$s_2.t \circ s_1.t \circ b_1.t$; $s_2.t \circ s_1.t \circ lc_1(s_1).t$;
$s_2.t \circ s_1.t \circ rc_1(s_1).t$; $s_2.t \circ s_1.t \circ lc_1(s_2).t$;
$s_2.t \circ s_1.t \circ rc_1(s_2).t$; $s_2.t \circ s_1.w \circ rc_1(s_2).t$;
$s_2.t \circ s_1.w \circ lc_1(s_1).t$; $s_2.t \circ s_1.w \circ b_1.t$

Figure 2. The feature templates used for analysis. $lc_1(s_i)$ and $rc_1(s_i)$ denote the leftmost and rightmost children of s_i, w denotes word, t denotes POS tagging.

C. Artificial Neural Network

Artificial Neural Networks (ANN) is a classification method that copy the working principle of the human neural network. This method maps input data at the input layer to the target at the output layer through neurons in the hidden layer [11]. In this Final Project, the author uses the Artificial Neural Networks classification method to do parsing using Dependency Parser. This method works to study the dense vector representation of words, POS tagging, and dependency relations. Figure 3 is an example of the Artificial Neural Network method architecture taken from previous research [2].

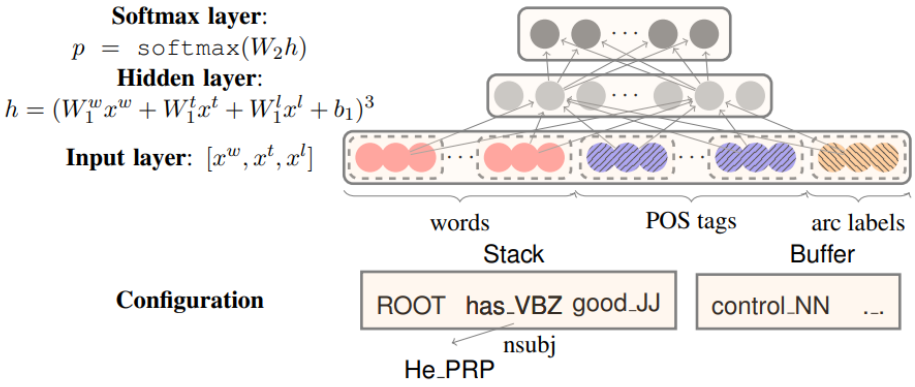


Figure 3. Artificial Neural Network architecture

In the Artificial Neural Network classification method there are 3 layers, namely the Input Layer, Hidden Layer, and the Output Layer or Softmax Layer. At the input layer, element selection is based on the Stack and Buffer positions for each type of information in the form of words, POS tagging, and arc labels. Each information is given a notation as S_w, S_t, and S_l. Appropriate embeddings of selected information will be added to the input layer. In the figure, ROOT shows POS or word and NULL indicates there is no correct feature value that can be calculated [15].

From the input layer then will enter the hidden layer, in this final project the hidden layer is used for testing which amounts to 1 and 5 consisting of each Relu unit. Each unit in the hidden layer is connected to the previous layer. After that will enter the output layer or softmax layer, this layer is the outermost layer or output layer on the Artificial Neural Network. Softmax functions to convert numbers into probabilities that add up to one, wherein each information will be added to the value of one [2].

D. Evaluation Metrics

Dependency Parser evaluation is done by measuring how good the parser works on the test set. Therefore, the most common evaluation metrics suitable for evaluating Dependency Parsers are the accuracy of the LAS (Labelled Attachment Score) and the UAS (Unlabelled Attachment Score). LAS (Labelled Attachment Score) is an evaluation metric that labels words appropriately and has correct dependency relationships between words. UAS (Unlabelled Attachment Score) is an evaluation metric that labels words appropriately but ignores existing dependency relations [6].

In addition to the LAS and UAS accuracy values, the authors also use parsing speed as one of the evaluation metrics. Parsing speed is used to measure the time needed by the system to parse using the Artificial Neural Network classification method. While the LAS and UAS accuracy values were obtained after parsing by finding the best accuracy values from the data train and data dev on each Arabic, Chinese, English, and Indonesian dataset.

III. RESEARCH METHOD

A. System Description

The system built in this paper aims to produce a fast and concise classification, both in terms of accuracy and parsing speed, by calculating the value of LAS and UAS on each dataset obtained from parsing results using Dependency Parser. A general description of the system to be built in this paper can be seen in Fig 4.

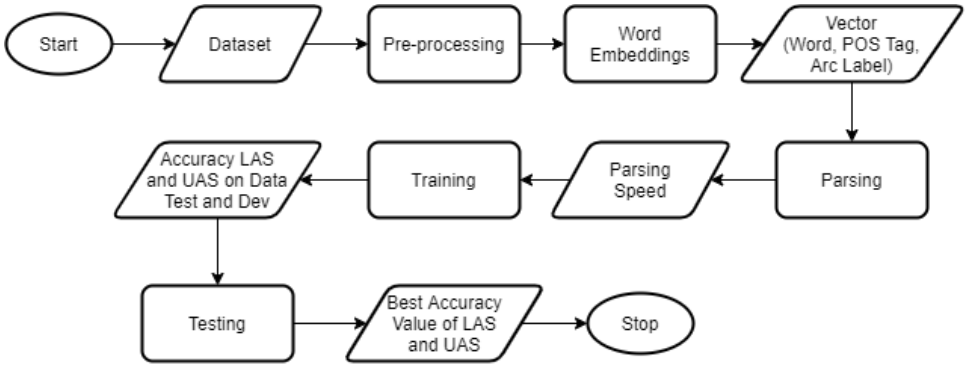


Figure 4. General Description of the System

B. Dataset

The dataset used in this paper is Arabic, Chinese, English, and Indonesian from the Universal Dependency website [3] using the CONLL format. The dataset is divided into 3 parts, namely train, dev, and test based on PTB3 standard separator rules, for training using sections 2-21, development uses section 22, and section 23 as a test set. This dataset will be used as input from the system. The use of 4 different datasets aims to see the performance of the parser if tested in various types of languages that have different characteristics and grammar. In previous studies it was said that grammar in every language can significantly affect the performance of parsing [4].

Table I is a data statistic from the dataset that will be used in this study, where there is the amount of data that has been separated into a train, dev, and test, the total number of words available, the number of POS tagging, the number of dependency relations, and the total of each dataset.

Table I
DATA TEST INPUT

Dataset	**#Train**	**#Dev**	**#Test**	**#Word(N_w)**	**#POS(N_t)**	**#Label(N_l)**
Arabic	590,819	73,945	74,125	738,889	16	24
Chinese	110,058	10,094	10,562	130,714	13	39
English	204,585	25,148	25,096	254,829	17	49
Indonesian	97,531	12,612	11,780	121,923	16	31
Total	**1,002,993**	**121,799**	**109,783**	**1,246,355**	**62**	**143**

C. Pre-processing

Pre-processing is the stage carried out to process documents or data, this is done to ensure the dataset is ready for processing. This study used a dataset that has been separated into 3 parts, namely train, dev, and test, based on PTB3 standard separator rules.

1) Word Embeddings: At this stage, every word in the entire dataset will be converted into a vector consisting of a collection of numbers with word embeddings. In this study the word embeddings used are word2vec. This aims to make it easier to do parsing later. In the Artificial Neural Network classification

method, word embeddings occur in the configuration to the input layer, where elements are selected based on the position of the Stack and Buffer for each type of information in the form of words, POS tagging, and arc labels.

2) Parsing: After that parsing is done using 2 different hyperparameters. In the Artificial Neural Network classification method, parsing occurs in the hidden layer where testing is performed using hidden layers totaling 1 and 5 consisting of each Relu unit connected to the previous layer. Then it will enter the softmax layer where every information in the form of words, POS taggging, and arc labels will be added to the value of one which will be the output of the Artificial Neural Network classification. When parsing is done, the system will automatically calculate the speed of the parser which will then be displayed if the parsing has finished.

D. Training

At the training stage, each test data and dev data will be calculated LAS and UAS accuracy values for every 10 epochs. Every 1 epoch of LAS and UAS values will automatically compared to the previous epoch, if the current LAS and UAS value is greater than before, then the current LAS and UAS value will be automatically set to the best value until training completion.

E. Testing

After completing the training, it will proceed with the testing phase. At this stage, the best LAS and UAS values for each test data and dev data obtained from the previous training results will be displayed as the output of the system.

F. Evaluation

The presentation of results should be simple and straightforward. This section reports the most important findings, including the results of statistical analysis as appropriate and comparisons to other research results. Results given in figures should not be repeated in tables. This is where the author should explain in words what he/she/they discovered in the research. It should be laid out and in a logical sequence. This section should be supported by suitable references.

Testing is done using the python programming language that aims to obtain the value of LAS and UAS from each dataset. The dataset used is Arabic, Chinese, English, and Indonesian obtained from the Universal Dependency website [3], the total number of words in the dataset is 1,246,355. The test results obtained from 2 different hyperparameters can be seen in Table II, the first hyperparameter based on research that has been done before, [2] and the second hyperparameter is a new parameter that the authors made to be used as a comparison in testing.

Table II
The hyperparameter used for Testing

Hyperparameter	**First**	**Second**
Learning Rate	0.001	0.001
Hidden Layer	1	5
Hidden Layer Size	200	500
Epoch	10	10
l2 Regularization	$10e^{-8}$	$10e^{-8}$
Activation Function	Relu	Relu
Optimizer	Adam	Adam

IV. Results and Discussion

From the tests that have been carried out using the first hyperparameter the results can be seen in Table III, and the results of tests conducted using the second hyperparameter can be seen in Table IV. In both

tables, it can be seen that the tests conducted with the second hyperparameter obtain higher results than the first hyperparameter. As in the English dataset, the first hyperparameter in the test data obtained LAS 79.9% and UAS 82.4%, and with dev data obtained LAS values of 81.2% and UAS 83.8%, and parsing speed of 86 sentences per second. The second hyperparameter in the test data obtained LAS 80.4% and UAS 83%, and with dev data obtained LAS values 81.1% and UAS 83.7%, and parsing speed of 98 sentences per second.

Table III
RESULT USING THE FIRST HYPERPARAMETER

Dataset	Test		Dev		Parsing Speed
	LAS	UAS	LAS	UAS	
Arabic	73.0	78.3	73.2	78.8	71
Chinese	64.6	72.8	68.5	75.3	74
English	79.9	82.4	81.2	83.8	86
Indonesian	73.0	78.6	71.3	77.1	87

Based on the LAS and UAS accuracy values obtained from test and dev data, higher results are obtained using the second hyperparameter than the first hyperparameter. There are differences in the value of the first and second hyperparameters, namely the number of hidden layers and hidden layer sizes. In the first hyperparameter, there are 1 hidden layer and 200 hidden layer sizes, while in the second hyperparameter there are 5 hidden layers and 500 hidden layer sizes. This proves that the more hidden layers, the better the results of accuracy and parsing speed.

Table IV
RESULT USING THE SECOND HYPERPARAMETER

Dataset	Test		Dev		Parsing Speed
	LAS	UAS	LAS	UAS	
Arabic	73.5	78.8	73.6	79.0	79
Chinese	66.5	74.9	68.7	75.6	71
English	80.4	83.0	81.1	83.7	98
Indonesian	74.2	79.9	72.7	78.5	83

V. CONCLUSION

In this study, testing using the second hyperparameter resulted in an accuracy of LAS and UAS as well as a higher parsing speed than the first hyperparameter. This proves that the more hidden layers, the better the results of accuracy and parsing speed. Further research is expected to use more datasets. The use of different hyperparameters can also be done for parsing. Testing on the Transition-Based Dependency Parser can use other Neural Network methods such as RNN.

ACKNOWLEDGMENT

The author would like to thank the people involved in this research, such as lecturers, friends, and parents who have provided many input and suggestions to the author. And to all authors whose papers are used as reference material in this paper. That means a lot to the writer in conducting this research.

References

[1] Giuseppe Attardi. Experiments with a multilanguage non-projective dependency parser. pages 166–170, 2006.
[2] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. pages 740–750, 2014.
[3] Filip Ginter, Yoav Goldberg, and Jan Hajić. Universal dependency. https://universaldependencies.org/, 2014. Online; Accessed 9 June 2020.
[4] Spence Green and Christopher D Manning. Better arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 394–402, 2010.
[5] He He, Hal Daumé III, and Jason Eisner. Dynamic feature selection for dependency parsing. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1455–1464, 2013.
[6] Daniel Jurafsky and James H Martin. *Dependency parsing*. Speech and Language Processing, 2009.
[7] Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. pages 423–430, 2003.
[8] Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, 2008.
[9] Sandra Kübler, Ryan McDonald, and Joakim Nivre. Dependency parsing. *Synthesis lectures on human language technologies*, 1(1):1–127, 2009.
[10] Tetsuya Nasukawa. Parsing method and system for natural language processing, June 2 1998. US Patent 5,761,631.
[11] D. Nasution, T. H. F. Harumy, E. Haryanto, F. Fachrizal, Julham, and A. Turnip. A classification method for prediction of qualitative properties of multivariate eeg-p300 signals. In *2015 International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT)*, pages 82–86, 2015.
[12] Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 149–160, 2003.
[13] Joakim Nivre, Johan Hall, and Jens Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. 6:2216–2219, 2006.
[14] Chuan Wang, Bin Li, and Nianwen Xue. Transition-based chinese amr parsing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 247–252, 2018.
[15] David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. Structured training for neural network transition-based parsing. *arXiv preprint arXiv:1506.06158*, 2015.