Converting an Indonesian Constituency Treebank to the Penn Treebank Format

Jessica Naraiswari Arwidarasti, Ika Alfina, and Adila Alfa Krisnadhi
Faculty of Computer Science, Universitas Indonesia
Depok, Indonesia
Email: jessica.naraiswari91@ui.ac.id, ika.alfina@cs.ui.ac.id, adila@cs.ui.ac.id

***Abstract*—A constituency treebank is a key component for deep syntactic parsing of natural language sentences. For Indonesian, this task is unfortunately hindered by the fact that the only one constituency treebank publicly available is rather small with just over 1000 sentences, and not only that, it employs a format incompatible with readily available constituency treebank processing tools. In this work, we present a conversion of the existing Indonesian constituency treebank to the widely accepted Penn Treebank format. Specifically, the conversion adjusts the bracketing format for compound words as well as the POS tagset according to the Penn Treebank format. In addition, we revised the word segmentation and POS tagging of a number of tokens. Finally, we performed an evaluation on the treebank quality by employing the Shift-Reduce parser from Stanford CoreNLP to create a parser model. A 10-fold cross-validated experiment on the parser model yields an F1-score of 70.90%.**

***Keywords*-constituency parsing; Indonesian; Penn Treebank; Stanford parser; treebank format**

I. Introduction

In Natural Language Processing (NLP) research, syntactic parsing plays a major role in determining the semantics of the sentences. Syntactic parsing is a key prerequisite subtask for many general NLP tasks such as machine translation, grammar checking, natural language generation, information extraction, and question answering system.

In general, syntactic parsing can be divided into two classes, phrase-structure or constituency parsing and dependency parsing [1]. Constituency parsing aims to parse a sentence into its constituent phrases. Meanwhile, dependency parsing is used to uncover the dependency relation between words in the sentence.

Figure 1 shows an example of a constituency tree for the sentence *"Bus menabrak sebuah motor." ("A bus hit a motorcycle.")*. The root *S* represents that this sentence is a simple declarative clause. This sentence is parsed into two parts: an NP-SBJ (Noun Phrase as subject) for the segment *"Bus" (bus)* and a VP (Verb Phrase) for the segment *"menabrak sebuah motor" (hit a motorcycle)*. The VP is then parsed into two further parts: a terminal with Part-of-Speech (POS) tag of VB for *"menabrak" (hit)* and an NP (Noun Phrase) for the segment *"sebuah motor" (a motorcycle)*. The latter phrase is the parsed into two terminals: *"sebuah" (a)* with POS tag DT (determiner) and *"motor" (motorcycle)* as NN (noun).

The availability of syntactic corpus (treebank) is very important to build a parser model using data-driven parsing method. For English, various treebanks are available,

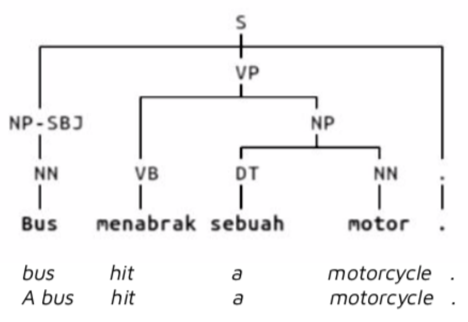

Figure 1. A constituency tree

among which is the Penn Treebank [2]. The annotation format of this treebank is considered the standard format in building constituency treebanks.

Various tools to process the treebanks with the Penn Treebank format are readily available, such as the Stanford Parser [3], the Trance Parser [4], and Discodop [5]. With those tools, we can train treebanks of any language to build a constituency parser model.

On the other hand, resources for syntactic parsing for Indonesian are very limited, despite being a language spoken by more than 260 million people. The only constituency treebank available was developed by Universitas Indonesia[1] in 2015 as the continuation of the development of their POS-tagger corpus [6]. This treebank, hereinafter called Universitas Indonesia Constituency Treebank (UI-CTB), consists of 1,030 sentences containing 27,115 words and was annotated manually. It adopted the Penn Treebank format with some differences in the bracketing format and the POS tagset.

The main difference between UI-CTB and the Penn Treebank format is in the handling of compound words. A compound is *"a combination of two simple words which come together to form a complex word"* [7]. In Indonesian there are three ways to write the compound words: 1) as a single word, such as *kacamata* (*eyeglasses*); 2) hyphenated, such as *pemuda-pemudi* (*youngsters*); and 3) as multiple tokens, such as *rumah sakit* (*hospital*). The bracketing format of UI-CTB was designed to maintain the last type of compound word as a single unit in the treebank. Table I shows some examples of how UI-CTB represents the compound words.

While the bracketing format used by UI-CTB retains the meaning of the compound words, this format unfortunately

[1]https://github.com/famrashel/idn-treebank

Table I
THE REPRESENTATION OF COMPOUND WORDS IN UI-CTB.

Compound	Representation in UI-CTB
tanggung jawab (responsibility)	(NN (tanggung jawab))
bertanggung jawab (to be responsible)	(VB (bertanggung jawab))
luar biasa (excellent)	(JJ (luar biasa))
sama sekali (at all)	(RB (sama sekali))
salah satu (one of)	(CD (salah satu))
salah seorang (one of)	(CD (salah seorang))
lagi pula (moreover)	(CC (lagi pula))
di mana (where)	(IN (di mana))

does not comply with the expected input format of various constituency treebank processing tools. These tools, in general, expected input in the Penn Treebank format for which compound words are treated rather differently. This format will be explained in Section 2.

To overcome this problem, earlier works that utilized UI-CTB to build Indonesian constituency parser converted the treebank so that compound words become one token and adjusted the bracketing format to the Penn Treebank format. For example, Filino & Purwarianti [8] and [9] opted to change the representation of *"luar biasa"* from *(JJ (luar biasa))* to *(JJ luar_biasa)* by combining the individual tokens into a single token by using an underline character. We consider this solution oversimplified since there are no such words in real sentences.

The objective of our work is thus to convert the UI-CTB so that it conforms to the Penn Treebank format, both the bracketing format and the POS tagset. We have made the new treebank public[2] so that it can be used by other studies on Indonesian syntactic parsing. To evaluate the quality of the treebank, we used it to build an Indonesian constituency parser model using Stanford Parser and got F1-score of 70.90%.

The rest of this paper is organized as follows: Section 2 discusses the related work; Section 3 presents the proposed conversion method; Section 4 is for experiments using Stanford Parser, and finally, Section 5 presents the conclusion and future work.

II. Related Work

In this section, we discuss the Penn Treebank format, UI-CTB, Stanford parser, and Shift-Reduce parser.

A. The Penn Treebank format

The Penn Treebank [2] is a constituency treebank for English built by University of Pennsylvania in 1993. It contains around 7 million words with Part-of-Speech (POS) tags and around 3 million words with syntactic tags [10]. The Penn Treebank annotation (hereinafter referred to as PTB) used the POS tagset of 48 labels and the syntactic tagset of 14 labels. The annotation scheme of PTB is considered as the reference standard in creating the constituency treebanks for English and other languages.

[2]https://github.com/ialfina/kethu

Unfortunately, we could not find any published work discussing the procedure employed by PTB to annotate compound words. We thus resort to direct observation to examples of compound nouns and verbs in PTB to gain some understanding on how PTB generally represents compound words. Table II lists some of these examples. From the observation, we found that:

- For the compound nouns, each token is labeled as NN and all tokens are wrapped in the NP (Noun Phrase) syntactic tag. For example, for *paper work* that is a compound noun, *paper* and *work* are labeled as NN and both tokens are wrapped in an NP syntactic tag.
- For the compound verbs, the second token is labeled with POS tag of RP (particle) and this token also wrapped out with a special syntactic tag of *PRT*. Then, all tokens are wrapped in a VP (Verb Phrase) syntactic tag.

Table II
THE REPRESENTATION OF COMPOUND WORDS IN PTB.

Compound	Representation in PTB
paper work	(NP (NN paper) (NN work))
point out	(VP (VBP point) (PRT (RP out)))
picking up	(VP (VBG picking) (PRT (RP up)))
took away	(VP (VBD took) (PRT (RP away)))

B. UI Constituency Treebank

UI Constituency Treebank (UI-CTB) was built in 2015, as the continuation of a project to build a POS tagger corpus by Dinakaramani et al. [6]. This POS tagger corpus consists of 10,000 sentences containing 262,330 lexical tokens that had been annotated manually. The sentences of this corpus are the first 10,000 sentences taken from IDENTIC corpus that built by Larasati in 2012 [11]. IDENTIC consists of 45,000 Indonesian-English parallel sentences. In [11], it was stated that some parts of the IDENTIC corpus are taken from the corpus from PAN Localization Project[3] [12].

Although UI-CTB annotation format is similar to the PTB format, there are some notable differences. First, there are differences in bracketing format as explained in the earlier section, and second, it uses a different Part-of-Speech (POS) tagset. UI-CTB POS tagset consists of 23 tags [6], while PTB has a total of 48 POS tags (36 POS tags plus 12 other tags for punctuation and currency symbols).

As far as we know, UI-CTB has been utilized in two earlier works. In 2016, Filino & Purwarianti [8] used UI-CTB as the basis for building an Indonesian constituency parser model using Trance parser, which is a shift-reduce neural constituent parser employing recurrent neural networks [4]. In that work, 978 trees were used out of the total of 1,030 trees in UI-CTB as trees that do not represent complete sentences were excluded. Some preprocessing steps were done to the UI-CTB, such as deletion of *U* symbol, modification of compound words, deletion of null

[3]http://www.panl10n.net/indonesia/#Linguistic_Resources

element, deletion of null subordinate conjunction, and trees normalization. Also, the compound words are joined into a single token by using the underline character. They achieved an F1-score of 74.91% when using parameter of hidden state vector size of 128 and word representation vector size of 2048.

The second work that utilized UI-CTB is from Kurniawan [9]. He built an Indonesian constituency parser from UI-CTB using Recurrent Neural Network Grammars (RNNG) [13]. All trees in the treebank were used in this work and compound words are treated similarly as in Filino & Purwarianti's work [8]: they are joined into a single token using the underline character. From several scenarios, it was reported that the highest F1-score of 78.63% was achieved when using Generative RNNG with a pre-trained word embedding.

C. Stanford Parser

Stanford Parser is a part of Stanford CoreNLP [3]. For constituency parsing, Stanford Parser provides three classifiers to create the parser model: 1) Probabilistic Context-Free Grammar (PCFG) parser; 2) Recursive Neural Network (RNN) parser; and 3) Shift-Reduce Parser.

PCFG parser was built by Klein and Manning [14] in 2003. The RNN parser was made by Socher et al. [15] in 2011, and the Shift-Reduce parser was written by John Bauer based on the works of Zhu et al. [16] and others. Based on experiments run in 2014, when tested on PTB, it was reported that the F1-score of those three parses are: 85.54% for the PCFG parser, 88.55% for the Shift-Reduce parser, and 89.96% for the RNN parser.

In this work, we only utilized the Shift-Reduce parser of the Stanford CoreNLP to evaluate the new treebank.

D. Shift-Reduce Parser

The Shift-Reduce Parser performs the parsing by maintaining a state of the current parsed tree, with the words of the sentence on a queue and partially completed trees on a stack. The parser applies the transitions to the state until the queue is empty and the current stack only contains a finished tree. According to Zhu et al. [16], the Shift-Reduce (SR) parser was developed based on the shift-reduce process proposed by Sagae & Lavie [17] and employs global perceptron training and beam search.

The shift-reduce parsing is based on a scan of the input sentence. A transition action is applied to consume an input word or construct a new phrase-structure at each step. A stack is used to maintain partially constructed phrase structures, while the input words are stored in a buffer. The set of transitions are [16]:

- SHIFT: pop the front word from the buffer, and push it onto the stack.
- REDUCE-L/R-X: pop the top two constituents off the stack, combine them into a new constituent with label X, and push the new constituent onto the stack.
- UNARY-X: pop the top constituent off the stack, raise it to a new constituent with label X, and push the new constituent onto the stack.
- FINISH: pop the root node off the stack and ends parsing

The SR parser is considered much faster than other existing parsers based on the experiments for parsing sentences in the Penn Treebank. Not only faster, but it is also considered more accurate than other parsers outside the RNN parsers. The Trance parser [4], a neural Shift-Reduce parser, was reported to achieve an F1-score of 90.68% on the Penn Treebank, and the Shift-Reduce parser of the Stanford Parser was reported to achieve the best F1-score of 88.55% for the Penn Treebank on their website.

III. Proposed Method

In this section, we present our proposed method in converting UI-CTB to the PTB format. The conversion consists of a data cleansing as a preprocessing task and the main conversion task composed of two subtasks: adjustment of the bracketing format and modification of the POS tags. The main conversion task is implemented in a program we call **PTB Converter**, while we named the resulting treebank after conversion as **Kethu**[4].

A. Data Cleansing

Prior to converting the UI-CTB to the PTB format, sentences in UI-CTB require somewhat extensive corrections. This is achieved in the data cleansing task consisting of (1) splitting/merging sentences, (2) correcting typographical errors, (3) correcting the word segmentation, (4) correcting lost punctuation, (5) correcting POS tagging, and (6) correcting the annotation for reported speeches.

First, splitting and merging sentences are necessary since the current UI-CTB contains data where more than one sentences are associated with a tree, while some trees are associated with incomplete sentences. Splitting is performed for the former and merging for the latter.

Second, we correct typographical errors by (i) deleting excessive words; (ii) correcting the letter case so that uppercase and lowercase letters are used appropriately, e.g., when writing proper names; and (iii) adding missing characters.

Third, we correct the word segmentation. This concerns several things. One, reduplicated word such as *anak-anak* (kids), should not be split into multiple tokens. Two, abbreviations like *No.*, should not be split into multiple tokens. Three, punctuation that is part of a name, such as the dash character in *APBN-P*, should not cause the word to be split. Four, bounded morpheme part of a word, like *non-*, *anti-*, *pasca-*, or *antar-*, should not cause the word to be split. Finally, we perform special handling for words ended by the clitic of *-nya* as explained in [18]. In *mencintainya (to love him/her/it)*, *-nya* should be separated from its main word, while in *terbentuknya* (the formation (of)), *-nya* should not be separated.

The fourth step is inserting lost punctuation. For example, in a sentence like ".. *Crude Palm Oil* ***CPO***...", CPO is the abbreviation of the previous words and should be wrapped with a pair of parentheses. This sentence

[4]Kethu is the name of a forest in Wonogiri, Central Java, Indonesia

should be corrected to "... *Crude Palm Oil* ***(CPO)***...". Unfortunately, the occurrence of this kind of error is abundant in UI-CTB. We revised more than 200 words that lose their parenthesis. We investigated further and found out these mistakes originated from the source of the treebank [12].

The fifth step is correcting POS tagging errors. This includes (1) revising incorrect POS tagging, (2) changing POS tag due to merging/splitting tokens, and (3) revising POS tags due to ambiguous words with more than one possible POS tags. Some words whose POS tag were changed are *ingin* (want), *perlu* (need), and *mulai* (begin), which we converted to VB (verb).

The last step is correcting the reported speech annotation. In UI-CTB, the pair of double quotes were placed outside the syntactic tag of S-TPC-1 that contains the reported speech, while in PTB format, those double quotes should be put in the S-TPC-1 fragment.

B. Main conversion task: PTB Converter

The PTB Converter performs the conversion via two steps: (1) adjusting the bracketing format and (2) changing the POS tags.

1) Adjusting the bracketing format: In this step, we change the bracketing format so that it conforms to the PTB bracketing format. Here, we also split the compound words into tokens. There are 442 words represented as compound words in UI-CTB. Since we only found examples for compound noun and verb in PTB, for other types of compound in Indonesian, we propose an annotation format exemplified in Table III, which also shows examples of our proposed mapping from UI-CTB to PTB bracketing format.

Table III
MODIFICATION IN THE BRACKETING FORMAT.

No	UI-CTB	Kethu
1	(NN (laporan))	(NN laporan)
2	(NN (tanggung jawab))	(NP (NN tanggung) (NN jawab))
3	(VB (bertanggung jawab))	(VP (VB bertanggung) (PRT (RP jawab)))
4	(JJ (luar biasa))	(ADJP (PRT (RP luar)) (JJ biasa))
5	(RB (sama sekali))	(ADVP (PRT (RP sama)) (RB sekali))
6	(CD (salah satu))	(NP (PRT (RP salah)) (CD satu))
7	(CD (salah seorang))	(NP (PRT (RP salah)) (DT seorang))
8	(CC (Lagi pula))	(ADVP (RB Lagi) (PRT (RP pula)))
9	(IN (di mana))	(ADVP (WRB di) (PRT (RP mana)))
10	(NP-SBJ (*))	(NP-SBJ (-NONE- *))
11	(-LRB- (Z (&brl;)))	(-LRB- -LRB-)
12	(-RRB- (Z (&brr;)))	(-RRB- -RRB-)

Example 1 shows the modification to a single token, examples 2–9 are for compound words modification, and examples 10-12 are for other bracketing issues. In example 2 and 3, we adopted PTB's approach in handling compounds of noun and verb. Examples 4–9 are our proposed annotation for compound adjective and others.

Example 10 is for the case where the segment *(*)* is considered a syntax error in PTB since it has no POS tag. In PTB, these tokens are given the POS tag *-NONE-*, so we change every occurrence of *(*)* to *(-NONE- *)*.

Example 11 and 12 are for the tokens *-LRB-* (left parenthesis) and *-RRB-* (right parenthesis). In UI-CTB, both tokens are treated as a syntactic tag, while in PTB they are a POS tag. Moreover, these labels only occur 3 times in UI-CTB as syntactic tags, while in Kethu we have 255 pair of them, as the POS tags.

At the end of this step, we have converted the bracketing format of UI-CTB into PTB. Now, UI-CTB can be processed by any tool that expects treebank in PTB format. However, we want to make sure that our new treebank, Kethu, also complies with the POS tagset of PTB. So, we need to do the next step, mapping the POS tagset of UI-CTB to PTB.

2) Mapping the POS Tagset from UI-CTB to PTB: We analyze the differences between UI-CTB POS tagset [6] and PTB tagset [10]. Table IV shows our proposed mapping of specific tags in UI-CTB to the associated POS tags in PTB.

Table IV
THE MAPPING OF UI-CTB'S SPECIFIC TAGS TO PTB TAGS.

UI-CTB	Description	PTB
NEG	negation	RB
NND	classifier noun	NN
OD	ordinal number	JJ, CD
PR	demonstrative pronoun	DT
PRP	pronoun	PRP, PRP$, DT
SC	subordinating conjunction	IN
VB	verb	VB, VBZ
WH	question word	WP, WP$, WRB, WDT, IN
X	unknown	FW
Z	punctuation	-LRB-, -RRB-, comma (,), period (.), :, ", "

For OD in UI-CTB, only those that describes a noun is converted to JJ (adjective) such as in *"anak kedua" (the second child)*, in *"kedua anak" (both children) kedua* will be labeled as CD.

For PRP in UI-CTB, we converted the tokens of possessive personal pronoun to PRP$. Some tokens with PRP but we considered the semantic are as the determiner were converted from PRP to DT (determiner), such as in *(NP (NN hal) (DT nya)) (the case)*.

In UI-CTB there is only one type of verb. We decided to differentiate verb (VB) into VB and VBZ. VBZ will be used to label copula verbs that have special characteristic compare to other verbs in Indonesian. According to [7], copula verbs in Indonesian are *adalah* and *ialah*.

In UI-CTB, there is only one label for question word, WH. Since in PTB, there are four types of question words, we mapped WH in UI-CTB to 5 labels: WP, WP$, WRB, WDT and IN. WP for *apa (what)* and *siapa (who)*; WP$

for *siapa* following noun; WRB for *di mana (where), kapan (when), ketika (when)* and *bagaimana (how)*; WDT for *yang (which, that)* or *yang mana (which)*; and finally IN, if those question words are used as the subordinating conjunction.

Furthermore, we also found that in UI-CTB, words like *beberapa (some), seorang (a), semua (all)* are labeled as CD (cardinal number). Since in PTB these words are tagged as a determiner (DT), we also revised them.

At the end of this step, the treebank conversion is done, and we obtain a new version of UI-CTB that conforms to the Penn Treebank format.

C. Statistics of the Treebanks

We present some statistics of both UI-CTB and the Kethu treebank to give pictures of how far we have changed the UI-CTB.

1) The comparison of the number of sentences and words: Table V shows the number of sentences, number of words and the average length of sentences in both treebanks. Although there is no change in the number of sentences, actually we did some splitting and merging on UI-CTB sentences. For the number of words, it is important to note that what we mean here by ”word” conformed to how each treebank represents the word. In UI-CTB, a compound word is considered as a word, while in Kethu, a word is a single token. So, it's not surprising that the number of words in Kethu is higher than UI-CTB. Not to mention we also added more than 200 pairs of parentheses. In the end, the average sentence length of Kethu treebank is a bit higher than the UI-CTB.

Table V
COMPARISON OF THE NUMBER OF SENTENCES AND WORDS.

Description	**UI-CTB**	**Kethu**
Number of sentences	1,030	1,030
Number of words	27,115	28,117
Avg sentence length	26.33	27.3

2) The distribution of POS tags: Table VI shows the distribution of POS tag in UI-CTB and Kethu treebank. The column with *”-”* means that the POS tag was not the member of the corresponding treebank POS tagset.

The total number of the first seven POS tags that represent the punctuation in Kethu is 3426, with a margin of 774 with Z in UI-CTB. This amount difference is contributed by the additional 255 pairs of brackets and more than 90 pairs of double quotes.

The number of CD in Kethu was decreased around 500 while number of DT was increased as the result of converting words like *beberapa (some), seorang (a), semua (all)* from CD to DT. The number of IN in Kethu was increased significantly since almost all words with SC tag in UI-CTB was converted to IN.

IV. EXPERIMENTS AND RESULTS

In this section, we present the experiments and results.

Table VI
THE POS TAG DISTRIBUTION.

POS	**UI-CTB**	**Kethu**	**POS**	**UI-CTB**	**Kethu**
-LRB-	-	255	NNP	3868	4070
-RRB-	-	255	OD	100	-
,	-	1430	PR	508	-
:	-	89	PRP	704	478
.	-	1012	PRP$	-	145
”	-	242	RB	564	732
“	-	243	RP	18	70
CC	776	802	SC	1156	-
CD	2372	1854	SYM	331	286
DT	20	594	UH	4	5
FW	423	390	VB	2885	2874
IN	2264	3349	VBZ	-	37
JJ	1073	1068	WH	8	-
MD	535	492	WP	-	13
NEG	127	-	WRB	-	11
NN	6455	7321	X	24	-
NND	148	-	Z	2752	-

A. Experiments

To evaluate the quality of the resulting treebank (Kethu), we use the Shift Reduce parser of the Stanford CoreNLP to build the Indonesian constituency parser model. The evaluation method used is 10-fold cross-validation. We divided Kethu treebank into 10 parts, each consists of 103 sentences. The training dataset consists of 8 parts of 824 sentences, the development and test dataset each consists of 103 sentences.

B. Results

After conducting the 10-fold cross validation method, we got the average of Labeled Precision (LP) of 70.37%, Labeled Recall (LR) of 71.46% and F1-score of 70.90%.

Table VII shows a summary of the F1-score of 4 studies on Shift-Reduce parsing. In general, we can see that Trance parser outperforms the Stanford parser, both for English and Indonesian. If we compare the accuracy of English and Indonesian parser, we can see the Indonesian parser trained using a variant of UI-CTB has a lower accuracy than the English parser trained using PTB.

Table VII
THE COMPARISON OF SOME WORKS ON SHIFT-REDUCE PARSING

Parser	**Treebank**	**F1 (%)**
Trance parser	PTB	90.68
Trance parser	UI-CTB revised by [8]	74.91
Stanford parser	PTB	88.55
Stanford parser	Kethu	70.90

Our result is lower than [8] that also uses a revised UI-CTB. However, since we only conducted a simple experiment to Kethu treebank without doing tree normalization or binarization like Filino & Purwarianti [8] did, we cannot compare the quality of Kethu treebank to the revised UI-CTB they use.

We suggest three reasons why Indonesian parser has lower accuracy: 1) the size of the treebank that is very small compared to PTB; 2) the quality of the treebank, and 3) special characteristics of the Indonesian grammar.

To have a better Indonesian parser, building a new treebank with a bigger size and also in better quality in word segmentation and POS tagging is needed. We also need to involve Indonesian linguists to investigate whether some special treatments are needed to compute Indonesian grammar.

V. Conclusion and Future Work

We have proposed a method to convert an Indonesian constituency treebank to the Penn Treebank format. This needs to be done so that we can utilize the various tools provided for constituency treebanks such as the Stanford Parser. The conversion consists of two main tasks: data cleansing and developing the component named the PTB Converter.

We evaluated the quality of the new treebank by building the Indonesian constituency parser model using Shift-Reduced parser provided by the Stanford CoreNLP. The experiments show that our parser model has an F1-score of 70.90%.

The main contribution of our work is to provide an Indonesian constituency treebank that conforms to PTB format so that it can be used as the input of any tools that expected a treebank in PTB format.

In this work, we only revised the word segmentation, the POS tagging of UI-CTB, and correcting the annotation of reported speech. We have not reviewed other syntactic tagging of the treebank. In future work, we will conduct analysis to its syntactic annotation in order to improve the quality of the treebank.

We also recommend building a larger treebank with a bigger size and better quality so that we can produce a better Indonesian parser. The format of this treebank should also conform to the international standard format so that we can utilize tools that are already available.

Acknowledgment

This research was supported by the research grant of ”Hibah PITTA B 2019” from Universitas Indonesia, No. NKB-0508/UN2.R3.1/HKP.05.00/2019.

References

[1] D. Jurafsky and J. H. Martin, *Speech and Language Processing, 2nd Edition: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2008.

[2] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, “Building a large annotated corpus of English: The Penn Treebank,” *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[3] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, “The Stanford CoreNLP Natural Language Processing Toolkit,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2015, pp. 55–60.

[4] T. Watanabe and E. Sumita, “Transition-based Neural Constituent Parsing,” in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 1169–1179. [Online]. Available: http://www.aclweb.org/anthology/P15-1113

[5] A. Van Cranenburgh, R. Scha, and R. Bod, “Data-Oriented Parsing with Discontinuous Constituents and Function Tags,” *Journal of Language Modelling*, 2016.

[6] A. Dinakaramani, F. Rashel, A. Luthfi, and R. Manurung, “Designing an Indonesian part of speech tagset and manually tagged Indonesian corpus,” in *Proceedings of the International Conference on Asian Language Processing 2014, IALP 2014*, 2014, pp. 66–69.

[7] J. N. Sneddon, A. Adelaar, D. N. Djenar, and M. C. Ewing, *Indonesian Reference Grammar*. A&U Academic, 2010.

[8] M. Filino and A. Purwarianti, “Indonesian shift-reduce constituent parser,” in *Proceedings of 2016 International Conference on Data and Software Engineering, ICoDSE 2016*, 2016.

[9] K. M. Kurniawan, “Exploring Recurrent Neural Network Grammars for Parsing Low-Resource Languages,” Master Thesis, University of Edinburgh, 2017.

[10] A. Taylor, M. P. Marcus, and B. Santorini, “The Penn Treebank: An Overview,” in *Text, Speech and Language Technology*, 2003, pp. 5–22.

[11] S. Larasati, “IDENTIC Corpus: Morphologically Enriched Indonesian-English Parallel Corpus,” in *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, 2012, pp. 902–906.

[12] BPPT, “Research Report on Corpus Design and Collection and Cleaning Tools English to Bahasa Indonesia,” Tech. Rep., 2010.

[13] C. Dyer, A. Kuncoro, M. Ballesteros, and N. A. Smith, “Recurrent Neural Network Grammars,” in *NAACL 2016*, 2016. [Online]. Available: http://arxiv.org/abs/1602.07776

[14] D. Klein and C. D. Manning, “Accurate unlexicalized parsing,” in *ACL ’03 Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1 pp 423-430*, Sapporo, Japan, 2003.

[15] R. Socher, C.-Y. Lin, A. Ng, and C. Manning, “Parsing natural scenes and natural language with recursive neural networks,” in *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, 2011.

[16] M. Zhu, Y. Zhang, W. Chen, M. Zhang, and J. Zhu, “Fast and Accurate Shift-Reduce Constituent Parsing,” in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2013.

[17] K. Sagae and A. Lavie, “A classifier-based parser with linear run-time complexity,” in *Proceedings of the Ninth International Workshop on Parsing Technology*, Vancouver, British Columbia, 2005.

[18] I. Alfina, A. Dinakaramani, M. I. Fanany, and H. Suhartanto, “A Gold Standard Dependency Treebank for Indonesian,” in *Proceeding of the 33rd Pacific Asia Conference on Language, Information and Computation (PACLIC 33)*, Hakodate, Japan, 2019.