Journal of Computer Science

Original Research Paper

Tree Rotations for Dependency Trees: Converting the Head-Directionality of Noun Phrases

Ika Alfina, Indra Budi and Heru Suhartanto

Faculty of Computer Science, Universitas Indonesia, Depok, Indonesia

Article history
Received: 02-09-2020
Revised: 12-11-2020
Accepted: 17-11-2020

Corresponding Author:
Ika Alfina
Faculty of Computer Science,
Universitas Indonesia, Depok,
Indonesia
Email: ika.alfina@cs.ui.ac.id

Abstract: To overcome the lack of NLP resources for the low-resource languages, we can utilize tools that are already available for other highresource languages and then modify the output to conform to the target language. In this study, we proposed an approach to convert an Indonesian constituency treebank to a dependency treebank by utilizing an English NLP tool (Stanford CoreNLP) to create the initial dependency treebank. Some annotations in this initial treebank did not conform to Indonesian grammar, especially noun phrases' head-directionality. Noun phrases in English usually have head-final direction, while in Indonesian is the opposite, head-initial. We proposed a variant of tree rotations algorithm named *headSwap* for dependency trees. We used this algorithm to convert the head-directionality for noun phrases that were initially labeled as a compound. Moreover, we also proposed a set of rules to rename the dependency relation labels to conform to the recent guidelines. To evaluate our proposed method, we created a gold standard of 2,846 tokens that were annotated manually. Experiment results showed that our proposed method improved the Unlabeled Attachment Score (UAS) with a margin of 32.5% from 61.6 to 94.1% and the Labeled Attachment Score (LAS) with a margin of 41% from 44.1 to 85.1%. Finally, we created a new Indonesian dependency treebank that converted automatically using our proposed method that consists of 25,416 tokens. The dependency parser model built using this treebank has UAS of 75.90% and LAS of 70.38%.

Keywords: Dependency Parsing, Head-Directionality, Indonesian, Noun Phrases, Tree Rotations

Introduction

Syntactic parsing is "*a task of recognizing an input string and assigning a structure to it*" (Jurafsky and Martin, 2008). In general, the approaches in syntactic parsing are divided into two types: Phrase structure and typed-dependency structure. Phrase structure focuses on identifying phrases and their recursive structure, while the type-dependency structure focuses on relations between words. Phrase structure parsing is also known as constituency parsing.

The dependency parsing has gained more popularity because of its applicability to a wide range of NLP tasks such as machine translation (Čmejrek *et al*., 2004; Galley and Manning, 2009; Jiang *et al*., 2016; Gao *et al*., 2017), information extraction (Niklaus *et al*., 2018; Gashteovski *et al*., 2019), question answering (Meng *et al*., 2017; Cao *et al*., 2018) and so on. These works have motivated the conversion of the available constituency treebanks to the dependency treebanks.

Several works had built constituent-to-dependency converter for English treebank (Johansson and Nugues, 2007; Choi and Palmer, 2010; De Marneffe *et al*., 2006; Schuster and Manning, 2016). These converters accept treebanks in the Penn Treebank format as the input. The Penn Treebank (PTB) is a constituency treebank in English (Marcus *et al*., 1993). The PTB annotation guidelines are considered as a de-facto standard in building constituency treebank.

For non-English treebank, many converters of constituency-to-dependency have been built, such as for Arabic (Žabokrtský and Smrz, 2003), Spanish (Gelbukh *et al*., 2005), French (Candito *et al*., 2010) and Sanskrit (Goyal and Kulkarni, 2014). In general, these works used a rule-based approach based on the target language's morphology and syntactic in converting constituency to dependency annotation.

Indonesian, a language of the Austronesian language family, is a low-resource language for Natural Language Processing (NLP) studies. Not only are dataset limited, tools for processing datasets are also rarely available.

As far as we know, the only constituency treebank available was developed by the Universitas Indonesia (UI) as the continuation of the development of their POS-tagger corpus (Dinakaramani *et al*., 2014). This treebank was later converted to the Penn Treebank format by (Arwidarasti *et al*., 2019).

As for the dependency treebank, there are two treebanks publicly available, both provided by Universal Dependencies (UD): Indonesian-GSD (McDonald *et al*., 2013) and Indonesian-PUD (Zeman *et al*., 2017). However, according to (Alfina *et al*., 2019), these two dependency treebanks do not fully conform to Indonesian grammar, especially the tokenization and POS tagging annotation. The Indonesian-PUD recently had been revised by (Alfina *et al*., 2019; 2020). This situation motivated us to convert the only Indonesian constituency treebank (Dinakaramani *et al*., 2014; Arwidarasti *et al*., 2019) to a dependency treebank.

In this study, we present a different approach to convert constituency to dependency annotation. Unlike previous works that create the tool from scratch for the target language, we prefer to utilize the already available tools for the high-resource language like English and conducting some adjustments so that the final output will conform to the target language, in this case to Indonesian grammar.

We proposed a method to revise the output of an English NLP tool named Stanford Universal Dependencies (SUD) converter (Schuster and Manning, 2016) so that the resulting treebank conforms to Indonesian grammar. SUD converter was initially built for treebank in English. It was reported that the accuracy of this tool is more than 90% for an English treebank. However, when we use this tool for treebank in Indonesian, we found out that accuracy is very low of around 60%.

After conducting error analysis, we observed that one of the low accuracy causes is the difference in head-directionality in some noun phrases between English-Indonesian. According to (Hawkins, 1990), there are two kinds of head-directionality: Head-initial or head-final. For head-initial, the second word describes the first word and for head-final, the opposite. While English usually uses head-final direction for noun phrases, Indonesian noun phrases usually use head-initial direction with some exceptions (Alwi *et al*., 2010).

Figure 1 shows an example of noun phrases in English and its corresponding noun phrases in Indonesian. For the English noun phrase, the position of *store* as the head is after the *book* as the dependent, while for Indonesian noun phrases, the position of *toko* (*store*) as the head is before *buku* (*book*).

Fig. 1: Head-directionality of the noun phrase in English (head-final) Vs. Indonesian (head-initial)

To revise the dependency tree to have noun phrases with the correct head-directionality, we proposed a variant of the tree rotations algorithms for dependency trees. Our tree rotations algorithm will change the shape of the tree where some tokens get promoted to be the head and other being demoted to be the dependent of the new head. We named our proposed tree rotations algorithms for dependency trees as the *headSwap* algorithm.

We also use this algorithm to implement a rule to convert the head-directionality of noun phrases that were initially labeled as a compound. Upon applying this rule, we achieved an improvement of around 32% for UAS (Unlabeled Attachment Score). This result shows the effectiveness of our proposed method.

The contributions of our work are three-fold:

1. We propose a variant of tree rotations algorithms named *headSwap* that works on the dependency trees to swap the head between two nodes
2. We present a case in which the *headSwap* algorithm can be applied: Revising the head-directionality of noun phrases that initially labeled as compound for Indonesian treebank
3. We produced a new dependency treebank for Indonesian that had been made public[1]

We believe our proposed *headSwap* algorithm can also be applied not only for noun phrases but also for other phrases such as verb phrases, prepositional phrases, etc. Since the head-directionality differences do not only happen between English and Indonesian, the *headSwap* algorithm can also be applied for dependency trees of other languages.

The rest of the paper is organized as follows: Section 2 describes the related work, section 3 presents differences between Indonesian and English noun phrases; section 4 describes our proposed method; section 5 discusses the experiments and results and finally, section 6 presents the conclusions and future work.

Related Work

In this section, we discuss dependency trees, Universal Dependencies (UD) and Stanford UD converter.

[1]https://github.com/ialfina/hd-converter

Dependency Trees

Dependency parsing is an approach to represent the syntactic structure of sentences in natural language using dependency grammar (Jurafsky and Martin, 2008). For dependency grammar, a sentence's syntactic structure is described in terms of the words and associated set of directed binary grammatical relations among the words. The arguments to this binary relation consist of a head and a dependent. Also, a label that describes the kinds of grammatical relation between the dependent and its head can be added.

Dependency graphs and dependency trees are used to represent the sentences for dependency parsing. In (Kübler *et al.*, 2009), a dependency graph/tree is defined as follows:

- A sentence is a sequence of tokens denoted by $S = w_0w_1\ldots w_n$ where w_0 is an artificial ROOT token.
- Let $R = \{r_1,\ldots, r_m\}$ be the dependency relation type set
- A dependency graph $G = (V, A)$ is a labeled directed graph consists of nodes V and arch A, such that for sentence $S = w_0w_1\ldots w_n$ the following holds:
  1. $V \subseteq \{w_0, w_1,\ldots,w_n\}$
  2. $A \subseteq VxRxV$
  3. if $(w_i, r, w_j)\in A$ then $(w_i, r_0, w_j) \notin A$ for all $r' \neq r$
- Any dependency graphs that is a directed tree originating out of node w_0 and has a spanning node set $V = V_S$ are called dependency trees
- A dependency tree $G = (V, A)$ satisfies the single-head property

Figure 2 shows a dependency graph for a sentence of "He worked for the BBC for a decade." and Fig. 3 shows the corresponding dependency tree.

Universal Dependencies

Universal Dependencies (UD) is currently the de-facto standard in annotating the dependency treebank. Before, some treebanks have their own annotation guidelines that made it difficult for cross-lingual parsing. A consistent and universal annotation guideline is needed for multilingual syntactic analysis.

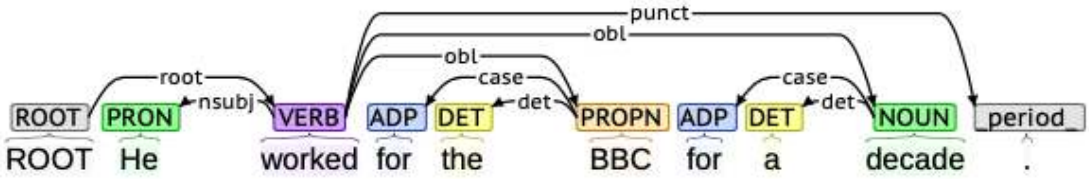

Fig. 2: A dependency graph for a sentence of "He worked for the BBC for a decade." (Zeman *et al.*, 2017)

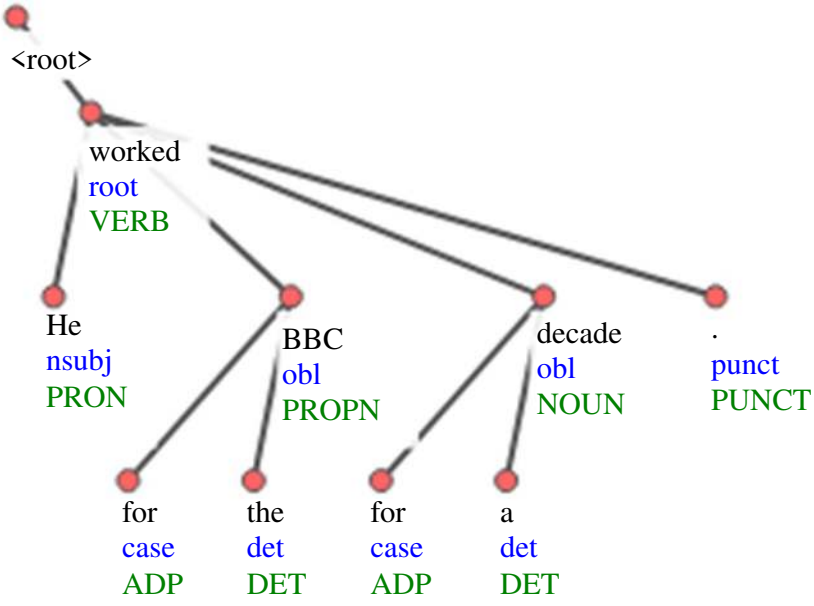

Fig. 3: A dependency tree for a sentence of "He worked for the BBC for a decade." (Zeman *et al.*, 2017)

De Marneffe *et al*. (2006) designed type dependency for English while conducting a project to convert the constituency to dependency treebank. This dependency type design later was developed into Stanford typed dependencies (De Marneffe and Manning, 2008). Stanford dependencies scheme was designed to represent English grammatical relations between words in a sentence. In 2014, the Stanford dependencies were adopted to create universal dependencies that can be applied to other languages to support cross-linguistically parsing, named Universal Stanford Dependencies (USD) (De Marneffe *et al*., 2014).

Finally, several initiatives agreed to create a new standard named Universal Dependencies by introducing annotation guidelines and a set of treebanks called Universal Dependencies v1 (UD v1) (Nivre *et al*., 2016). The recent version of the annotation guidelines is the UD v2 (Nivre *et al*., 2020), which has a tagset of 17 POS tags and 37 universal dependency relations with additional subtypes to adjust to the specific features of certain language.

Stanford Universal Dependencies Converter

Among the English NLP tools that can be used for low-resource languages like Indonesian is the Stanford UD converter (Schuster and Manning, 2016). This tool was built to convert an English constituency treebank in the Penn Treebank (PTB) (Marcus *et al*., 1993) format to a dependency treebank in the CoNLL-U format.

This converter is a revision to an initial converter (De Marneffe *et al*., 2006) that converts PTB-like treebank to dependency treebank in Stanford Dependencies (De Marneffe and Manning, 2008) representation. The SUD converter was reported to have UAS of 96.1% and LAS of 92.6% when evaluated on an English UD Treebank.

This UD converter tool is included in Stanford Core NLP (Manning *et al*., 2014). Since the constituency-to-dependency converter of the latest release of the Stanford CoreNLP only has output conforms to the UD v1 guidelines, Schuster had provided a tool[2] to convert it to the UD v2 format. We refer to the Stanford UD converter tool and its extension to UD v2 as the SUD+ converter.

Syntactic Annotation of Noun Phrases

In this section, we discuss the syntactic annotation of noun phrases. First, we present the differences between noun phrases in English and Indonesian, then we explain how UD v2 annotates noun phrases and finally, we describe how the SUD+ converter represents the noun phrases.

[2]https://github.com/UniversalDependencies/tools/tree/master/v2-conversion

Noun Phrases in Indonesian vs. English

Table 1 shows six types of Indonesian noun phrases with their corresponding head-directionality and the comparison with English. Note that on that table and the rest of this study, we use the Part-Of-Speech (POS) tagset of the Penn Treebank (PTB) (Marcus *et al*., 1993) to explain syntax or rules.

Noun phrases in line #1-#5 on Table 1 already discussed in (Alfina *et al*., 2019). We added the sixth syntax that is a special case of the 5th type (NN/NNP + NN/NNP), which involved nouns used to describe another noun's position. In (Alwi *et al*., 2010), the locative noun in Indonesian is discussed. Examples of such nouns are *atas* "*above*", *dalam* "*inside*" and *antara* "*between*". The locative nouns are used in prepositional phrase, with the syntax of *di/ke/dari* + [locative NN] + [NN/NNP] (Alwi *et al*., 2010).

Table 2 shows some examples of noun phrases with locative nouns in Indonesian and English corresponding phrases. In (Alwi *et al*., 2010), *di atas meja* is parsed into *di* and *atas meja*, not into *di atas* (*on*) and *meja* (*table*).

We can see from Table 1 that Indonesian noun phrases usually have head-initial direction, except for noun phrases with quantity determiner and locative noun.

Noun Phrases in the UD v2 Annotation Guidelines

The dependency relations (deprels) in UD annotation guidelines are divided into two groups: Universal dependency relations and language-specific relations called subtypes. UD v2 defines 37 universal deprels and many subtypes that can be defined for special construction in certain languages.

Table 1: The differences in head-directionality between Indonesian and English noun phrases

#	Syntax	ID	EN
1	NN/NNP + Demonstrative DT	Initial	N/A
2	Quantity DT + NN/NNP	Final	Final
3	NN/NNP + Possessive PRP	Initial	N/A
4	NN/NNP + JJ	Initial	N/A
5	NN/NNP + NN/NNP	Initial	Final
6	Locative NN + NN/NNP	Final	N/A

Note: The N/A values in the EN column means that such syntax doesn't exist in English

Table 2: Prepositional phrase in Indonesian vs. English

Indonesian	English
Di **atas** meja	**On** the table
Di **antara** kita	**Between** us
Ke **dalam** rumah	**Into** the house
Dari **pinggir** jalan	From **the side** of the road

Note: The bold words in the first column are locative nouns

Table 3: UD v2 dependency relation labels for noun phrases

Deprel	Type	Description	Example of NP
amod	Universal	For the adjective that describes the noun	**New** house
clf	Universal	For the classifier of a noun	*Tiga* ***buah*** *rumah* "three houses"
compound	Universal	For nominal compound word	**Ice** cream
det	Universal	For the determiner of a noun	**Some** students
flat	Universal	MWE for name, number, date, etc.	3 **hundred**
nmod	Universal	Noun modifier of a noun	Capital of **India**
nummod	Universal	For the numeric modifier of a noun	**25** books
flat:foreign	Subtype	MWE for foreign terms	-
flat:name	Subtype	MWE for name	Albert **Einstein**
nmod:poss	Subtype	The possessive determiner of a noun	**Her** book
nmod:tmod	Subtype	The time modifier of a noun	**2019** annual report

Note: Words with bold font in the 4th column are words to be annotated with the corresponding deprel label.

Table 4: The distribution of head-directionality of noun phrases of SUD+ converter output

Deprel	Freq.	H-final (%)	H-initial (%)
amod	686	1.60	98.40
compound	4331	99.54	0.46
det	132	100.00	0.00
nmod	2041	15.63	84.37
nmod:poss	145	0.00	100.00
nmod:tmod	70	14.29	85.71
nummod	1339	58.55	41.45

Among 37 universal deprels of UD v2, seven deprels are used to represent noun phrases. Table 3 shows the seven universal deprels for noun phrases and four subtypes that are usually used in the English dependency treebank. Note that the universal deprel *clf* that is used to label the classifier of a noun is rarely used in English, but this syntactic construction does exist in Indonesian and other languages.

How the SUD+ Converter Annotates Noun Phrases

In this study, we used the SUD+ converter to convert an Indonesian constituency treebank to a dependency one. We analyzed how the SUD+ converter represents the noun phrases. Table 4 shows the statistics of deprels that SUD+ converter used to annotate noun phrases in the Kethu treebank. We present the frequency of occurrences of each deprel along with the proportion of each head-directionality.

For deprel *amod* that is intended to label adjectives that describe nouns, SUD+ converter had already annotated them correctly since 98.40% of head-directionally are head-initial as expected by Indonesian grammar. For the remaining 1.60% of them are noun phrases affected by English terms such as *makro ekonomi* "*macro economy*" or some exceptions in Indonesian grammar such as for *pertama kali* "*first time*".

We conducted a further analysis for deprel compound that occurred 4331 times (around 15% of total 28,262 tokens). Table 5 shows the 11 noun phrases syntax where deprel compound is used for one of its tokens along with the example in Indonesian, the expected head-directionality and the expected deprel. Table 4 shows that 99.54% of head-directionality of deprel compound are head-final, but the head-directionality of 9 noun phrases syntax for Indonesian are head-initial. Only the fourth syntax in Table 5 aligns with the SUD+ converter's output that was initially designed for the English treebank. Based on the Indonesian dependency treebank's annotation guidelines (Alfina *et al*., 2019; 2020), almost all relations that initially tagged as a compound by SUD+ converter have the incorrect label.

For deprel *det*, we found that the SUD+ converter only labels quantitative determiners correctly since its syntax is the same as English. All demonstrative determiners are labeled as dep, a label used by UD v2 for unknown relation.

For deprel *nmod*, 84.37% of tokens are already tagged correctly since the expected head-directionality is headinitial in Indonesian grammar. We found out that most of the errors caused by SUD+ that uses nmod as the default value when there are options either to use *nmod* or *obl*.

For deprel *nmod:poss*, SUD+ converter has been correctly labeled the relation between the noun and possessive pronoun with 100% accuracy. However, it fails to recognize the possessive relationship between noun and noun. For deprel *nmod:tmod* and *nummod*, since both head-directionality are possible for them, we can not evaluate the correctness of SUD+ annotation using this data.

We can see that among the seven deprels used by the SUD+ converter to represent noun phrases, the deprel *amod* is the one that best fits Indonesian grammar, while the deprel *compound* is the least compliant. The deprel *det* and *nmod:poss* are also already 100% correct, but other cases that should be label det or *nmod:poss* are still tagged with deprel dep.

Based on this analysis, we decided to propose a method on how to revise the annotation for the deprel compound. Specifically, we propose the method to convert its the head-directionally from head-final to head-initial.

Table 5: List of noun phrases' syntax with label compound on one of its tokens produced by the SUD+ converter and the expected direction and deprel

#	Syntax	Example	Expected direction	Expected deprel
1	NN + NN (compound)	Air **mata**	initial/final	compound
2	NN + modifier NN	*toko* ***buku*** "*book store*"	initial	nmod
3	NN + possessor NN	bulu **kucing**	initial	nmod:poss
4	Locative NN + NN	**atas** meja	**final**	nmod:lmod
5	NN + NNP	*sepatu* ***Adidas*** "*Adidas shoe*"	initial	nmod
6	NNP + NN	*Sabtu malam* "*Saturday* ***night***"	initial	nmod
7	NNP + NNP	*Bill* ***Gates***	initial	flat:name
8	CD + CD	*5* ***juta*** "*5 million*"	initial	flat
9	FW + FW	*net* ***buy***	initial	flat:foreign
10	FW + NN	*rating* ***lembaga*** "*institutional rating*"	initial	nmod
11	FW + NNP	*rating* ***LBPN***	initial	nmod

Note: words with bold font in the 3rd column are words to be annotated with the corresponding expected deprel label

The Proposed Method

This section presents our proposed method of conducting tree rotations for dependency trees to convert the head-directionality of noun phrases.

Tree Rotations for Swapping the Head

Our proposed tree rotations algorithm's objective is to swap the head between two nodes in a dependency tree. If node A is initially the head of node B, we want to change the tree, so that node A becomes the dependent of node B. The tree rotations should preserve the requirement for a dependency tree (Kübler *et al*., 2009). We named this proposed tree rotations algorithm the *headSwap* algorithm.

To illustrate the *headSwap* algorithm, we will use a sentence as the example: "*Pemkot Delhi berencana mendatangkan monyet dari negara bagian Rajasthan*." (*The Delhi city government plans to bring in monkeys from the state of Rajasthan*.). Figure 4a shows the initial dependency graph given by the SUD+ converter to this sentence and Fig. 4b is the expected dependency graph.

There are three dependency relations with label compound in Fig. 4a: Between token *Pemkot* and Delhi where Delhi becomes the head, between token *negara* and *bagian* where bagian becomes the head and finally between token *bagian* dan Rajashtan. We want to swap the head-directionality of those three pairs of tokens, as shown in Fig. 4b, so that in the relation between *Pemkot* and Delhi, token *Pemkot* will become the new head. The same situation applies to the other two pairs. Note that we also need to swap the parent and the dependents of the respected tokens. The parent and the children of the old head become the parent of the new head.

In this study, we work on the dependency trees in the CoNLL-U format[3]. Among the ten fields in the CoNLL-U format, we will utilize only five fields: ID, FORM, UPOS, HEAD and DEPREL. Furthermore, we designed a data structure for token data with five attributes: ID, FORM, UPOS, HEAD and DEPREL. We define four arguments for the *headSwap* procedure, as shown in the **Algorithm 1**, as follows:

1. tokenList, contains the tokens data in a dependency tree
2. oldHeadID, the ID of the old head
3. newHeadID, the ID of the new head
4. moveDepFlag, the boolean flag whether the dependent(s) of the old head need to be moved to the new head

Algorithm 1: *headSwap*

Input: tokenList,oldHeadID,newHeadID,moveDepFlag
Output: the revised tokenList

```
oldDependentList ← []
foreach token in tokenList do
    if token.HEAD == oldHeadID and
     token:ID ≠ newHeadID then
        oldDependentList:append(token)
    end
end
oldHead ← tokenList[oldHeadID]
newHead ← tokenList[newHeadID]
label ← newHead:DEPREL
newHead:HEAD ← oldHead:HEAD
newHead:DEPREL ← oldHead:DEPREL
oldHead:HEAD ← newHeadID
oldHead:DEPREL ← label
if moveDepFlag == TRUE then
    foreach token in oldDependentList do
token:HEAD ← newHeadID
end
end
```

[3]https://universaldependencies.org/format.html

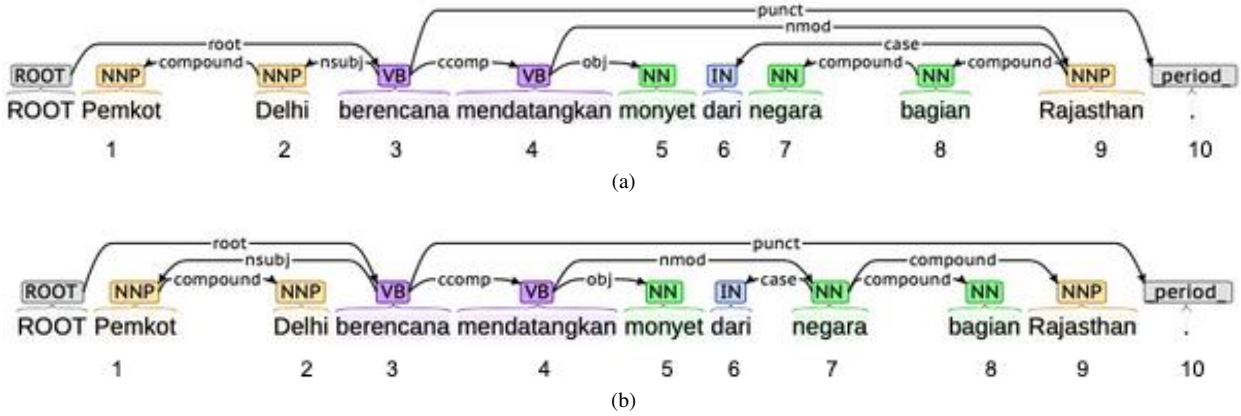

Fig. 4: An example of the initial and the expected dependency graphs for sentence "Pemkot Delhi berencana mendatangkan monyet dari negara bagian Rajasthan." (The Delhi city government plans to bring in monkeys from the state of Rajasthan.); (a) the initial dependency graph; (b) the expected dependency graph

This procedure swaps the head from the old head to the new one. If the flag is *true*, then all the old head's previous dependents will be moved to the new head.

Applying the HeadSwap Algorithm to Revise the Compound Noun Phrases

In this subsection, we explain in more detail how to convert the head-directionality for compound noun-phrases in a dependency tree. The procedure is shown in **Algorithm 2**.

Algorithm 2: compound

```
   Input: tokenList
   Output: the revised tokenList
1  phraseList ← generatePhraseList()
2  foreach phrase in phraseList do
3      skipFlag ← isException(phrase)
4      if skipFlag ≠ True then
5          old ← phrase:HEAD
6          new ← phrase:DEP[0]
7          headSwap(tokenList, old,new,True)
8          updatePhraseList(phrase)
9      end
10 end
```

To illustrate the proposed rule, we use the sentence in Fig. 4. This sentence consists of 10 tokens with ID of 1-10. First, we need to generate the list of tokens that become the head of the noun phrases contains the compound label. We named this list *phraseList* in the procedure. The *phraseList* is a list of A → B where A is the head and B is the dependent(s) of A. For our example, we have three pairs of tokens with the compound label: Pemkot-Delhi, negara-bagian and bagian-Rajashtan. For the first pair token #2 becomes the head of token #1, for the second pair token #8 is the head and for the third pair, the head is token #9. Hence, we have three phrases: 2 → {1}, 8 → {7} and 9 → {8}.

Secondly, for each phrase in *phraseList*, we need to apply the *headSwap* algorithm to change the head-directionality. However, since in Table 5, there are cases where the compound noun phrases already comply with Indonesian grammar, i.e., locative nouns case, we have to set the value of the *skipFlag* variable to *true* so that the phrase is not to be swapped. Otherwise, we applied the *headSwap* algorithm.

Finally, after applying *headSwap*, we need to update the *phraseList*. Table 6 shows how the *phraseList* was updated from the initial to final state. After each *headSwap*, we update the head and dependency information.

Revising the Dependency Relation Labels

Besides revising the head-directionality of noun phrases in the dependency tree, we also proposed a set of rules named ***rename*** to revise the dependency relation labels. These rules were designed by observing the recent version of an Indonesian-PUD treebank revised by (Alfina *et al*., 2019). Table 7 shows the design of our 32 proposed rules to improve the accuracy of dependency relation labels.

The decision to rename was made based on the information of the current deprel label of a token (old deprel), the POS tag of the token (child POS) and the POS tag of the head of the token (parent POS). In some cases, the decision was based on more detailed information. For example, for label nmod:lmod that is used to represent the locative nouns, this rule needs to have a list of locative nouns in Indonesian.

The rule *rename* currently only revised the label for a token initially labeled as compound, *compound:prt*, *dep*, *nmod*, *nmod:tmod*, *nsubj* and *obj*. In future work, we can add more labels after conducting further analysis of the annotation guidelines.

Table 6: Updating the phrases list for rule compound

Description	The state of PhraseList
Initial	2→{1}, 8→{7}, 9→{8}
After 1st iteration	1→[2}, 8→{7}, 9→{8}
After 2nd iteration	1→{2}, 7→{8}, 9→{7}
After 3rd iteration	1→{2}, 7→{8}, 7→{9}

Table 7: The rules for revising the dependency relation labels

#	Old Deprel	Child POS	Parent POS	New Deprel
1	compound	CD	CD	flat
2	compound	FW	FW	flat:foreign
3	compound	FW	NN/NNP	nmod
4	compound	NN	NN/NNP/FW	nmod
5	compound	NN (locative)	NN/NNP/FW	nmod:lmod
6	compound	NNP	NN/FW	nmod
7	compound	NNP	NNP	flat:name
8	compound:prt	RP	VB/JJ	advmod:emph
9	dep	CC any	POS	cc
10	dep	CD	CD	flat/nmod
11	dep	CD	NN/NNP/FW/SYM	nummod
12	dep	CD	VB/JJ	obl
13	dep	DT any	POS	det
14	dep	IN	NN/NNP/FW/PRP/CD	case
15	dep	IN	VB/JJ	mark
16	dep	MD any	POS	aux
17	dep	NN (locative)	NN/NNP/FW	nmod:lmod
18	dep	NN/NNP/FW/PRP/WP	NN/NNP/FW/PRP/WP/CD	nmod
19	dep	NN/NNP/FW/PRP/WP	VB/JJ	obl
20	dep	PRP$	NN/NNP/FW	nmod:poss
21	dep	RB any	POS	advmod
22	dep	RP (foregrounding) any	POS	advmod:emph
23	dep	RP (negating words) any	POS	advmod
24	dep	VB/JJ	NN/NNP/FW/PRP/CD	acl
25	dep	VB/JJ	VB/JJ	advcl
26	nmod	NN/NNP (temporal)	NN/NNP/FW/PRP	nmod:tmod
27	nmod	NN/NNP (temporal)	VB/JJ	*obl:tmod*
28	nmod	NN/NNP/FW/PRP/WP	VB/JJ	obl
29	nmod:tmod	NN/NNP (temporal)	VB/JJ	*obl:tmod*
30	nsubj	any POS	VB (passive)	nsubj:pass
31	obj	NN/NNP (temporal)	VB	*obl:tmod*
32	obj	any POS	VB (passive)	obl

For deprel *compound*, we created seven rules (#1-#7) that were aligned with the discussion in section 3, especially with Table 5 that discusses the expected deprel label for each syntax that was initially labeled as *compound*.

For deprel *compound:prt*, since in English, there is a unique construction of verb compound with syntax ”Verb (VB) + Particle (RP)” such as give up, take down and so on, SUD+ converter labels each occurrence of this syntax to compound:prt. However, for Indonesian grammar, such syntax is not for verb compound, but for foregrounding particles such as *lah*, *kah*, *tah* and pun (Sneddon *et al*., 2010). Rule #8 was designed for this problem.

The deprel *dep* is used to label unknown relations. The SUD+ converter used this label if it does not familiar with the syntax on the processed treebank. So, it is important to rename this label completely to other labels. To rename this label, we proposed 17 rules (#9-#25) for various cases.

We also decided to revise the label of deprel *nmod* that in UD v2 is used to represent the nominal modifier of a noun. In UD v2 annotation guidelines, the nominal modifier of a noun is labeled as *nmod*, while the nominal modifier of a predicate of verb/adjective should be labeled as *obl* (oblique modifier). However, we found out that SUD+ incorrectly labels tokens that should be labeled as *obl* as also *nmod*. Rule #28 was designed to fix this problem.

Rules #26, #27, #29 and #31 are related to noun phrases that are used as a temporal modifier. There are two subtypes for temporal modifier: *nmod:tmod* and *obl:tmod*. The label *nmod:tmod* is used if the phrase modifies a noun and *obl:tmod* is used if the phrase is to modify a predicate of verb/adjective. Rules #26 and #27

are used to revise the label for a token that initially labeled as *nmod*, rule #29 is used for the incorrect label of *nmod:tmod* that should be *obl:tmod*, while rule #31 are for tokens that is initially labeled as *obj* (the object of a transitive verb) but actually should be labeled as *obl:tmod*.

Rules #30 and #32 are related to passive verbs. Using rule #30, we revise the token that initially tagged as *nsubj* to *nsubj:pass* if its head is a passive verb and in rule#32 we change the label from *obj* to *obl* if the head is a passive verbs since grammatically passive verbs could not have an object.

Experiments and Results

This section discusses the experiments in converting the head-directionality of noun phrases and automatically building an Indonesian dependency treebank.

Dataset

In this study, we use the Kethu treebank produced by (Arwidarasti *et al*., 2019) as the initial constituency treebank to be converted to a dependency treebank. This treebank uses the same format as the Penn Treebank, both the Part-Of-Speech (POS) tagset, the bracketing and the annotation guidelines, which makes Kethu suitable as the input for the SUD+ converter that designed for the Penn Treebank. The Kethu treebank consists of 1,030 sentences and 28,262 tokens, with an average sentence length of 27.4 tokens per sentence. The genre of the sentences is news, mainly about the economy and finance.

To evaluate the proposed method, we chose 105 of 1,030 sentences of the Kethu treebank as the sample. For every 50 sentences, five first sentences are chosen to make sure a representative sample from the original treebank was created. This dataset consists of 2,846 tokens. We named this subset of Kethu treebank as the Kethu-105 treebank.

Creating the Gold Standard

We used the SUD+ converter to convert the Kethu-105 treebank to a dependency treebank in UD v2 format. We regarded the dependency treebank from Kethu-105 as the baseline treebank that will be converted using our proposed method so that the noun phrases comply with Indonesian grammar. To evaluate our proposed method, we need to create a gold standard.

The gold standard was created by revising the baseline treebank manually. Two annotators with a background in computer science and Indonesian linguistics revised the dependencies.

The gold standard creation consists of three phases that conducted iteratively: (1) Learning the UD v2 annotation guidelines; (2) Learning the proposed adjustment of UD v2 to Indonesian grammar by (Alfina *et al*., 2019; 2020); and (3) Revising the baseline treebank manually.

Several meeting was held to compare and discuss the annotation results between the two annotators until all of the inter-annotator disagreements were resolved. The resulting gold standard was named Gold Kethu-105.

Evaluating the Proposed Method

We evaluated the accuracy of our proposed method using MaltEval (Nilsson and Nivre, 2008). The quality measurements used are Unlabeled Attachment Scores (UAS) and Labeled Attachment Score (LAS) (Kübler *et al*., 2009). Table 8 shows the experiment results for three scenarios. First, the evaluation for the output of SUD+ converter as the baseline. The second scenario is by applying SUD+ converter plus the rule *compound* and the last scenario by combining SUD+ converter, rule *compound* and *rename* altogether.

We evaluated the accuracy of our proposed method using MaltEval (Nilsson and Nivre, 2008). The quality measurements used are Unlabeled Attachment Scores (UAS) and Labeled Attachment Score (LAS) (Kübler *et al*., 2009). Table 8 shows the experiment results for three scenarios. The first scenario evaluates the output of the SUD+ converter as the baseline. The second scenario is applying the rule compound to the output of the SUD+ converter. The last scenario is combining the SUD+ converter, rule compound and rename altogether.

For the baseline, the UAS and LAS are only 61.6 and 44.1%, respectively, which is very low. Since the SUD converter was reported to have an accuracy of more than 90% for an English treebank, we can see that some adjustments are needed to use this tool for non-English treebanks, especially for Indonesian in our case.

The result for the SUD+ converter plus rule *compound* is very good since the UAS improves significantly from 61.6 to 94.1% with a margin of 32.5% and the LAS improves 11.5% from only 44.1 to 55.6%. This result shows the effectiveness of our approach in converting the head-directionality of noun phrases in the treebank.

Finally, the last scenario combines the SUD+ *converter*, rule *compound* and rename to produce a dependency treebank that has UAS of 94.1% and LAS of 85.1%. This result shows that the rule *rename* has successfully improved the LAS with a margin of 29.5% from 55.6 to 85.1%.

Furthermore, we investigated what rules that have revised the deprel labels with reasonable accuracy.

Table 8: Experiment results

Description	UAS (%)	LAS (%)
SUD+ (baseline)	61.6	44.1
SUD+ + compound	94.1	55.6
SUD+ + compound + rename	94.1	85.1

Table 9: The comparison of the distribution of noun phrases in the Gold Kethu-105, baseline (SUD+) and the output of the rule rename

Deprel	Gold	SUD+	Rename	Diff
amod	80	79	79	-0.01
compound	-	418	-	-
det	79	20	80	-0.01
flat	49	-	37	-0.24
flat:foreign	9	-	9	0
flat:name	175	-	157	-0.10
nmod	430	197	488	0.13
nmod:lmod	18	-	13	-0.28
nmod:poss	23	16	17	-0.26
nmod:tmod	11	7	9	-0.18
nummod	98	102	106	0.08
Total	972	839	995	

Note: The last column contains the relative differences between the count in column Rename and Gold.

Furthermore, we investigated what rules that have revised the deprel labels with good accuracy. Table 9 shows the comparison of the distribution of deprel labels between the Gold Kethu-105, the baseline (output of SUD+ converter) and the final result after applying the rule *rename*. We can see the big differences between the Gold Kethu-105 and the SUD+ converter for labels of *compound*, *det*, *flat*, *flat:foreign*, *flat:name* and *nmod*.

We suggest that besides the differences in noun phrases syntax between English and Indonesian, the SUD+ converter itself has not implemented the rule for deprel *flat*, a universal dependency relation in UD v2 annotation guidelines. The SUD+ converter labels all dependency relations that should be *flat* or its subtypes to the compound.

We also can see that the rule rename have errors less than or equal to 10% for deprel *det*, *flat:foreign*, *flat:name* and *nummod*. For deprel *flat*, *nmod:tmod* and *nmod:poss* the errors are more than 23% which are need improvement.

Building Dependency Parser

Furthermore, we built an Indonesian dependency parser using the supervised method by using the remaining 925 sentences of Kethu treebank that were not used as the gold standard on the previous experiment as the training dataset. We named this part of the Kethu treebank as the Kethu-925 dataset. We converted this dataset to a dependency treebank using the best approach we got before: The combination of SUD+ converter, the compound rule with the *headSwap* method and the relabel rules. Table 10 shows the general comparison between the gold standard (Kethu-105) and the converted Kethu-925 dataset.

We can see that the average sentence length of both treebanks is almost the same, that we can suggest the level of difficulty for conducting parsing in both treebanks are more or less the same. The converted Kethu-195 treebank has more variations on the POS tagset and deprel labels. We found that there is no token with POS SYM that is usually used for nouns like %, or $ on the gold standard, while there are 17 occurrences of them in the Kethu-195 dataset. Kethu-925 also has tokens with POS WRB that are typically used for adverbial interrogative pronouns like *how*, *when*, *where* and *why*. This POS tag is not represented in the gold standard.

After creating the converted Kethu-925, we used UDPipe (Straka *et al*., 2016), a trainable pipeline for tokenization, tagging, lemmatization and dependency parsing of CoNLL-U files to build the Indonesia dependency parser model. The evaluation results for the resulting dependency are UAS of 75.90% and LAS of 70.38%. We consider this result quite good since another work by? that also built the Indonesian dependency parser had UAS of around 82% and LAS of 79% with the treebank that fully manually annotated with the size of 19,440 tokens and the average sentence length of only 19.4 tokens per sentence.

Moreover, we also already converted the POS tagset of the Gold-Kethu-105 and the Converted Kethu-925 from PTB to UD v2, along with additional information required by the UD validation tool like sentence *id*, *the original sentence before the tokenization* and *the tag SpaceAfter = No* that is used to indicate whether after a token there is a space or not. Both datasets are already made public[4].

From these two experiments, we have shown our approach's effectiveness in converting an Indonesian constituency treebank to a dependency one using an already available tool for English treebank and our proposed headSwap rules to convert the head-directionality of noun phrases in Indonesian sentences.

[4]https://github.com/ialfina/hd-converter

Table 10: The comparison of the Gold Kethu-105 and the converted Kethu-925

Description	Gold-105	Converted-925
Sentence count	105	925
Token count	2,846	25,416
Average sentence length	27.11	27.47
PTB POS count	24	26
Deprel count	37	36

Conclusion and Future Work

We proposed an approach to revise automatically a dependency treebank of a low-resource language that was initially produced by an NLP tool of a high-resource language. In our case, we need to revise an Indonesian dependency treebank produced by converting a constituency treebank to dependency using the Stanford UD converter (SUD+) that is designed for English treebank. We proposed a variant of tree rotations algorithm for dependency trees named *headSwap* to swap the head between two nodes to fix the wrong annotations.

We applied the algorithm to revise the head-directionality of noun phrases with the dependency relation label named *compound*, a label based on our observation that the SUD+ conversion has more than 90% incorrect head-directionality for Indonesian noun phrases. The rule to revise the head-directionally of the deprel compound was named rule *compound*. Moreover, we also propose a rule called rename to revise the dependency relation labels so that the treebank conforms to the UD v2 annotation guidelines and the recent guidelines used by an Indonesian dependency treebank.

To evaluate the proposed method, we conducted experiments on an Indonesian constituency treebank named Kethu. First, we created a gold standard of 105 sentences and 2,846 tokens by annotating them manually. The evaluation shows that the rule *compound* successfully improves the UAS by a big margin of 32.5% from 61.6 to 94.1% and the LAS with a margin of 11.5% from 44.1 to 55.6%. The rule *rename* finally improve the LAS to 85.1%. These results show the effectiveness of our proposed method in revising the output of the SUD+ converter for the Indonesian treebank.

Furthermore, we also built the Indonesian parser model using a new training dataset that was built automatically using our proposed method. The training dataset consists of 925 sentences and 25,416 tokens. We built the parser using UDPipe, a trainable pipeline for dependency parsing, to build the model. Experiments show that the resulting dependency parser model has UAS of 75.90% and LAS of 70.38%, a quite good result for a small treebank that automatically converted and has an average sentence length of 27.4.

We want to build a bigger Indonesian dependency treebank using a semi-supervised approach using the treebank produced by this study as the seed for future work.

Acknowledgement

The authors thank Jessica N. Arwidarasti and Arawinda Dinakaramani, who helped us in preparing and annotating the gold standard and also to Mohamad Ivan Fanany for the insightful feedback for the draft of this study.

Funding Information

This study was supported by the research grant of "Publikasi Terindeks Internasional (PUTI) Q2 2020" Number: NKB-1475/UN2.RST/HKP.05.00/2020 from Universitas Indonesia.

Author's Contributions

Ika Alfina: Literature study, design and implementation, data analysis, writing the article.

Indra Budi: Drafting the article and reviewing it for significant intellectual content.

Heru Suhartanto: Reviewing the paper for significant intellectual content and giving final approval of the version to be submitted and any revised revision.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Alfina, I., Dinakaramani, A., Fanany, M. I., & Suhartanto, H. (2019). A Gold Standard Dependency Treebank for Indonesian. In in Proceeding of the 33rd Pacific Asia Conference on Language, Information and Computation (PACLIC 33).

Alfina, I., Zeman, D., Dinakaramani, A., Budi, I., & Suhartanto, H. (2020). Selecting the UD v2 Morphological Features for Indonesian Dependency Treebank. In Proceedings of the 2020 International Conference of Asian Language Processing (IALP).

Alwi, H., Dardjowidjojo, S., Lapoliwa, H., & Moeliono, A. M. (2010). Tata bahasa baku bahasa Indonesia (ketiga). Jakarta: Balai Pustaka.

Arwidarasti, J. N., Alfina, I., & Krisnadhi, A. A. (2019, November). Converting an Indonesian Constituency Treebank to the Penn Treebank Format. In 2019 International Conference on Asian Language Processing (IALP) (pp. 331-336). IEEE.

Candito, M., Crabbé, B., & Denis, P. (2010, May). Statistical French dependency parsing: Treebank conversion and first results.

Cao, Q., Liang, X., Li, B., Li, G., & Lin, L. (2018). Visual question reasoning on general dependency tree. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 7249-7257).

Choi, J. D., & Palmer, M. (2010). Robust constituent-to-dependency conversion for English.

Čmejrek, M., Hajič, J., & Kuboň, V. (2004). Prague Czech-English dependency treebank: Syntactically annotated resources for machine translation. In Proceedings of EAMT 10th Annual Conference.

De Marneffe, M. C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., & Manning, C. D. (2014, May). Universal Stanford dependencies: A cross-linguistic typology. In LREC (Vol. 14, pp. 4585-4592).

De Marneffe, M. C., MacCartney, B., & Manning, C. D. (2006, May). Generating typed dependency parses from phrase structure parses. In Lrec (Vol. 6, pp. 449-454).

De Marneffe, M. C., & Manning, C. D. (2008, August). The Stanford typed dependencies representation. In Coling 2008: Proceedings of the workshop on cross-framework and cross-domain parser evaluation (pp. 1-8).

Dinakaramani, A., Rashel, F., Luthfi, A., & Manurung, R. (2014, October). Designing an Indonesian part of speech tagset and manually tagged Indonesian corpus. In 2014 International Conference on Asian Language Processing (IALP) (pp. 66-69). IEEE.

Galley, M., & Manning, C. D. (2009, August). Quadratic-time dependency parsing for machine translation. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (pp. 773-781).

Gao, S., Yang, X., Yu, Z., Pan, X., & Guo, J. (2017). Chinese-Naxi machine translation method based on Naxi dependency language model. International Journal of Machine Learning and Cybernetics, 8(1), 333-342.

Gashteovski, K., Wanner, S., Hertling, S., Broscheit, S., & Gemulla, R. (2019). Opiec: An open information extraction corpus. arXiv preprint arXiv:1904.12324.

Gelbukh, A., Torres, S., & Calvo, H. (2005). Transforming a constituency treebank into a dependency treebank. Procesamiento del lenguaje natural, (35), 145-152.

Goyal, P., & Kulkarni, A. (2014, August). Converting phrase structures to dependency structures in Sanskrit. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers (pp. 1834-1843).

Hawkins, J. A. (1990). A parsing theory of word order universals. Linguistic inquiry, 21(2), 223-261.

Jiang, W., Zhang, W., Xu, J., & Cai, R. (2016, November). Automatic Cross-Lingual Similarization of Dependency Grammars for Tree-based Machine Translation. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (pp. 501-510).

Johansson, R., & Nugues, P. (2007). Extended constituent-to-dependency conversion for English.

Jurafsky, D., & Martin, J. H. (2008). Speech and Language Processing: International Version: an Introduction to Natural Language Processing. Computational Linguistics and Speech Recognition, Pearson.

Kübler, S., McDonald, R., & Nivre, J. (2009). Dependency parsing. Synthesis lectures on human language technologies, 1(1), 1-127.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. (2014, June). The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations (pp. 55-60).

Marcus, M., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank.

McDonald, R., Nivre, J., Quirmbach-Brundage, Y., Goldberg, Y., Das, D., Ganchev, K., ... & Bedini, C. (2013, August). Universal dependency annotation for multilingual parsing. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 92-97).

Meng, Y., Rumshisky, A., & Romanov, A. (2017). Temporal information extraction for question answering using syntactic dependencies in an lstm-based architecture. arXiv preprint arXiv:1703.05851.

Niklaus, C., Cetto, M., Freitas, A., & Handschuh, S. (2018). A survey on open information extraction. arXiv preprint arXiv:1806.05599.

Nilsson, J., & Nivre, J. (2008, May). MaltEval: an Evaluation and Visualization Tool for Dependency Parsing. In LREC.

Nivre, J., De Marneffe, M. C., Ginter, F., Hajič, J., Manning, C. D., Pyysalo, S., ... & Zeman, D. (2020). Universal dependencies v2: An evergrowing multilingual treebank collection. arXiv preprint arXiv:2004.10643.

Nivre, J., De Marneffe, M. C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., ... & Tsarfaty, R. (2016, May). Universal dependencies v1: A multilingual treebank collection. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16) (pp. 1659-1666).

Schuster, S., & Manning, C. D. (2016, May). Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16) (pp. 2371-2378).

Sneddon, J. N., Adelaar, A., Djenar, D. N., & Ewing, M. C. (2010). Indonesian reference grammar (2 nd). Crows Nest. New South Wales, Australia: Allen & Unwin.

Straka, M., Hajic, J., & Straková, J. (2016, May). UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, pos tagging and parsing. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16) (pp. 4290-4297).

Žabokrtský, Z., & Smrz, O. (2003, April). Arabic syntactic trees: from constituency to dependency. In 10th Conference of the European Chapter of the Association for Computational Linguistics.

Zeman, D., Popel, M., Straka, M., Hajic, J., Nivre, J., Ginter, F., … & Li, J. (2017). CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, number 1, pages 1-19.