Utilizing Morphological Features for Part-of-Speech Tagging of Bahasa Indonesia in Bidirectional LSTM

I Nyoman Prayana Trisna*, Aina Musdholifah, and Yunita Sari
Departement of Computer Science and Electronics
Universitas Gadjah Mada
email*: nyomanprayana@mail.ugm.ac.id

2020 6th International Conference on Science in Information Technology (ICSITech) | 978-1-7281-7349-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICSITech49800.2020.9392076

***Abstract*—Research in the area of Part of Speech (PoS) Tagging has been widely explored especially for high resource language, such as English. However, there are only a small number of studies that have been conducted for Bahasa Indonesia. In this study, we present our experiment on utilizing morphological features for PoS tagging of Bahasa Indonesia in Bidirectional Long Short Term Memory architecture. Three different features including prefix, suffix, and capitalization have been examined. The results of our study show that combining morphological features with word embedding is effective for improving the tagger performance. Our study also provides more detailed explanation on which morphological features are useful for the PoS tagging task.**

***Index Terms*—POS Tag, morphological feature, bidirectional LSTM, word embedding, light-stemmer, fixed character affix**

I. INTRODUCTION

Part of Speech (PoS) Tagging is a Natural Language Processing (NLP) task that assigns a label to each word based on its grammatical function in a sentence [1]. The task of PoS Tagging is considered as a sequence labeling that obtains sentence as the input and sequence of words with their corresponding tag as the output [2]. This task plays an important role in many applications, including word parser, word sense disambiguation, and question answering machine [3].

The majority of PoS tagging work is focused on English and other high resource languages. In this study, we are focusing on Bahasa Indonesia, the official language of Indonesia, spoken by over 100 million people. There has been some work on developing PoS tagging for Bahasa Indonesia. Several approaches have been applied including rule-based [4], probabilistic [5], guided method [6] and neural network-based approach [7].

Research by Abka [7] attempt to use word embedding to transform input words into continuous vector representations in neural network architecture. Word embedding is considered a better approach since it does not omit the relation of each word in the sentence. However, this model ignores the morphological structure of the word that could improve overall performance of the model.

Previous studies show that utilizing morphological features in PoS tagging improved the tagger performance. Prior research using the probabilistic model by Muljono et al. [8] in Bahasa Indonesia proves that morphological features boost the model's performance. A similar result is reported by Zalmout and Habash [9] who used this feature for PoS Tagging in Arabic language. However, the neural model for PoS tagging in Bahasa Indonesia that utilizes morphological features is yet to be explored.

This study proposes a novel method that combines morphological features into the Bi-Directional Long Short Term Memory (Bi-LSTM) model for PoS tagging in Bahasa Indonesia. Two-layers Bi-LSTM model is used because it's considered as better approach for sequence labelling [10]. Our contributions are two-fold: first, we prove that morphological features are effective to improve PoS tagging performance especially in Bahasa Indonesia. Second, we provide a detailed study on which type of morphological features are best for this task.

II. RELATED WORK

The research around PoS Tagging of Bahasa Indonesia has been done previously. Widhiyanti and Harjoko [4] develops PoS Tagger in Bahasa Indonesia by employing the combination of rule-based method and probabilistic model of Hidden Markov Model. In similar research, Yuwana et al. [11] examine the use of common probability model for PoS Tagging in Bahasa Indonesia. The result by Yuwana et al. [11] shows that the best approach is with Maximum Entropy model. The accuracy by Maximum Entropy outperforms the accuracy of the model of previous research [4]. The PoS Tagger is further improved by Yuwana et al. [12] by using deep neural networks. The utilization of two layers deep neural network significantly increases the accuracy to 94.5%. The accuracy by Yuwana et al. [12] exceeds the result from prior mentioned researches [4][11].

Implementation of neural network in PoS Tagging is also observed in various languages. Plank et al. [13] explore the utilization of BiLSTM in PoS Tagging in 27 different languages as model. The research experiments various embedding approaches as input vectors. The proposed models are compared to other methods, such as TnT and Conditional Random Field (CRF) as the baseline models. The result shows that the proposed models yield better results than the baseline models. The proposed models by Plank et al. [13] also show that the usage of word embedding in input layer is better than character embedding only. Additionally, the use of word embedding outperforms the use of the combination of character embedding and word embedding. Similar models are also observed by Horsmann and Zesch [14]. Horsmann and Zesch explore 27 datasets from 21 languages. The research confirms a prior conclusion from Plank et al. [13] that BiLSTM is a remarkable model for PoS Tagging.

978-1-7281-7349-8/20/$31.00 ©2020 IEEE

The applications of morphological features for PoS Tagging have been researched in several languages with diverse models. Nedjo et al. [15] develop PoS Tagging for Oromo Language by applying Maximum Entropy Markov model (MEMM). The classifier of the model uses specialized features that employs morphological features. Some of them are n first character as prefix and n last character as suffix. The performance with specialized features for the model surpasses the baseline model from prior research [16]. Another research is examined by Muljono et al. [8] in Bahasa Indonesia. The research utilizes Hidden Markov model and morphological features by Larasati et al. [17]. The research by Muljono et al. [8] yields better performance compared to the baseline model from prior model by Wicaksono and Purwarianti [18].

The utilization of morphological features in LSTM model for PoS Tagging also has been examined, but not in Bahasa Indonesia. The research in Ughyur language by Maimaiti et al. [19] generates the engineered features from the morphological feature. These engineered features prove an improved result of the tagger. Similar research by Zalmout and Habash [9] in Arabic language experiments two kinds of embedding: word embedding and character embedding. For the morphological features, six configurations are attempted: (1) without morphological feature, (2) with fixed character affixes, (3) with light-stemmer, (4) with a morphological dictionary, (5) with the combination of fixed character affixes and morphological dictionary, and (6) with the combination of light-stemmer and morphological dictionary. Although the morphological dictionary gives the best result, it is impractical to use unless there is a dictionary or resource available. Ignoring the morphological dictionary, word embedding with light-stemmer gives better a result compared to the others.

III. Methodology

This research aims to develop the PoS Tagger for Bahasa Indonesia utilizing the morphological features of the word. The obtained dataset is sliced into two parts. The training part of the data is used to develop the model and the testing part of the data is used to evaluate the model. Both of training data and testing data's features are extracted. The model is then built with the training part to yield the PoS tagger model in Bahasa Indonesia. The tagger is then evaluated by the testing part, where the results are measured and analyzed.

A. Dataset

Dataset is obtained from research by Dinakaramani et al. [1] as is used in previous research [20]. The dataset contains tag list of 23 tags. The tags and their corresponding occurrence are displayed in Figure 1. There are 256,622 pairs of words and tags within 10,030 sentences. In this research, we reproduce the setup from Abka [7] which splits the dataset into 80% for data training and 20% for data testing. From data training, 80% of data is used for model training and 20% is used for model validation.

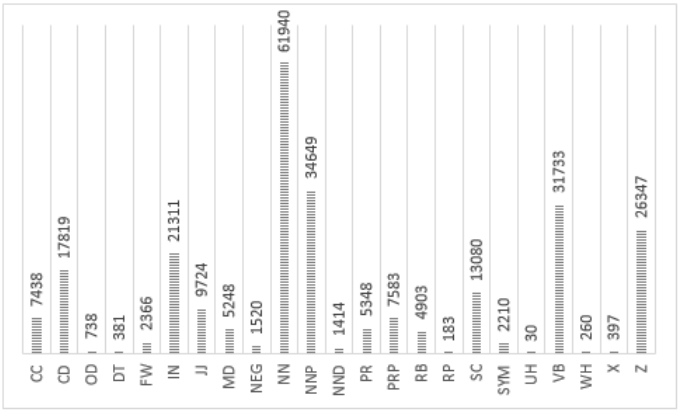

Fig. 1: Tags with their amounts in the dataset [1]

TABLE I: Basic affixes [21] that is used for this research

Prefix	di, ke, se, me, pe, be, te, ber, bel, ter, per, pel, pem, pen, mem, men, meng, meny, peng, peny, kese, sepe, menge, penge, diper, diber, keber, keter,seper, berse, pemer, teper,memper, member, mempel, pember, berpen
Suffix	kan, kah, lah, tah, pun, nya an, ku, mu, i

B. Feature Extraction

Morphological features are extracted for each word. There are three proposed features in this research that are extracted: (1) prefix of word, (2) suffix of word, and (3) capitalization of word.

This research tries two methods for affix (prefix and suffix) extraction, such as light-stemmer and fixed character affix. These extraction methods are inspired by the work of Zalmout and Habash [9]. Light-stemmer employs basic linguistic rule about affix without adding excessive resources. Setiawan et al. [21] previously addressed basic affixes for Bahasa Indonesia, and they are used as affix rules for this research. The affixes are written in Table I. However, suffix *-i* is not used since many words in Bahasa Indonesia are ended with *-i*. Another affix extraction for this research is fixed character affix. Fixed character affix extracts 3 front characters of the word as prefix and 3 rear characters of the word as suffix. The capitalization of the word is also used as a feature similar to previous research [15]. Table II shows the difference between light-stemmer and fixed character affix for morphological features, as well as the capitalization of words. The usage of extracted features will be explained in the next section.

TABLE II: Example of morphological features

Word	**Feature**	**With light-stemmer**	**With fixed character affix**
'menyanyikan'	Prefix	'meny'	'men'
	Suffix	'kan'	'kan'
	Capitalization	False	False
'Diputar'	Prefix	'di'	'dip'
	Suffix	None	'tar'
	Capitalization	True	True

C. Model Construction

Model construction is completed using training data. Model construction is separated into 4 different processes: indexing,

TABLE III: Example of indexed word and features

Word	Index
padding	0
OOV	1
'monyet'	2
'yang'	3
'memakan'	4
'buah'	5
....	....

Prefix	Index
padding	0
OOV	1
'memper'	2
'member'	3
'mempel'	4
'pember'	5
....	....

Suffix	Index
padding	0
OOV	1
'kan'	2
'kah'	3
'lah'	4
'tah'	5
....	....

word padding, input and output vectorization, and model training. Each part is completed one at the time by one.

The purpose of indexing is to adjust words or features (prefix and suffix) based on their corresponding index. The indexing base is made from training data. This could make the words or features that are not available on training data (called out of vocabulary or OOV) cannot be indexed. Therefore, OOV words or features are uniformed into the same index. Table III gives example of how words and features are indexed.

The padding process is used to equalize the length of a sentence before entering into model training. The sentence size is determined by one fixed size. The number of word padding in one sentence is as many as the fixed size minus the sentence length. The padding words are also indexed as shown by the example in Table III.

Before the model is trained, either input and output have to be converted into vectors. Words in each sentence as input are transformed into word embedding vector. Word embedding is a dense vector representation of a word that captures the relative meaning of a word based on its preceding and succeeding words [22]. In this research, the size of embedding for each word as a vector is defined as 128. Then the formed vector is appended with morphological features of prefix, suffix, and capitalization. For output, tags representations are formed in one-hot encoding vector, with size of 24. It is because the padding word has specific tag, so the tag list is added by one.

Model is formed by two layers of BiLSTM like similar researches [9][12]. The model is illustrated in Figure 2. The input of the model is vector that is combined from word embedding vector and morphological features vector. The output of model is one-hot vector of 24 tags, with each index of the vector is corresponding to one tag. This research sets the size of embedding as 128. For the size of the timestep, or size of the sentence is set to the maximum size of the sentence in training data and symbolized with n_{token}. The weight size of the cell in the first layer of BiLSTM is defined as n_{first_layer}. The same thing happens to the second layer of BiLSTM, named as n_{second_layer} for the size of the second layer. For the batch size of training, we symbolize as n_{batch} The size of n_{first_layer}, n_{second_layer}, and n_{batch} are defined in the experiment in the next section.

IV. Experiment

Experiments are carried out by trying several input layer configurations. In the experiment, we also try to use pre-trained word embedding with word2vec [22] besides the trained embedding in the model. The word2vec vectors are

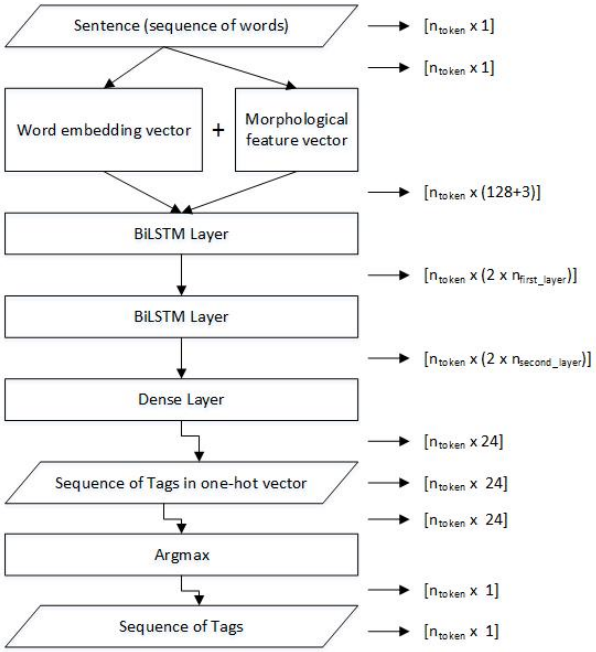

Fig. 2: Brief architecture of model with size of vector in each layer

formed by employing the Indonesia Wikipedia Dump Corpus[1]. The size of vector is set to 128, the same size as the trained embedding.

Each conducted experiment is shown in Table IV. Experiment 1 is used as baseline for trained embedding and Experiment 2 is used as baseline for pre-trained word2vec. This is due they are experiments with no utilization of morphological features.

TABLE IV: The Conducted Experiment

Experiment	Embedding	Morphological Feature (Affix Extraction)
#1	Trained embedding	No morphological feature
#2	Pre-trained word2vec	No morphological feature
#3	No word embedding	Fixed character affix
#4	No word embedding	Light-stemmer
#5	Trained embedding	Fixed character affix
#6	Pre-trained word2vec	Light-stemmer
#7	Trained embedding	Fixed character affix
#8	Pre-trained word2vec	Light-stemmer

For each experiment, we examine the use of combination of 256-128, and 128-64 as the n_{first_layer} and n_{second_layer} combinations. We set 128 and 32 as the size of the n_{batch}. Therefore there are 32 different experiments based on the experiment on Table IV, the combination of n_{first_layer}, n_{second_layer}, and n_{batch}.

V. Result and Discussion

In this research, obtained n_{token} from training is 82. Total index of each features is shown in Table V.

The result of each experiment is shown in Table VI, where MF stands for morphological feature, FCA is fixed

[1]last updated on December 2nd, 2019 20:44 at https://dumps.wikimedia.org/idwiki/latest/

TABLE V: Total index

Type of Index	Total
Word index	14670
Tag index	24
Prefix index (fixed character affix)	3320
Suffix index (fixed character affix)	3431
Prefix index (light-stemmer)	39
Suffix index (light-stemmer)	11

character affix, TE is trained embedding, and PT is pre-trained word2vec. Table VI shows Experiment 7 gives the best result based on accuracy. Experiment 7 is a model that employs word embedding and morphological features with light-stemmer. Compared to baseline Experiment 1, Experiment 7 increases the accuracy of the model by 1.44% until 1.90%. Another similar experiment is Experiment 5 that utilizes fixed character affix as morphological feature and produces better accuracy than Experiment 1 but still less than Experiment 7. This proves that the addition of morphological features increases the performance of the model, while the utilization of light-stemmer yield improved results than fixed character affix. This is equivalent to prior research [9] where light-stemmer is better. However, the baseline model as in Experiment 1 already shows impressive accuracy, as stated in previous research [12]. On the contrary, when the embedding vector is absent, the tagger with a morphological feature only provides poor performance, as shown in Experiment 3 and Experiment 4.

TABLE VI: Conducted experiment in this research and their corresponding accuracy

Experiment	MF	Embedding	Weight size = 256 - 128		Weight size = 128 - 64	
			n_{batch} = 128	n_{batch} = 32	n_{batch} = 128	n_{batch} = 32
#1	-	TE	94.34%	94.91%	94.69%	94.95%
#2	-	PT	92.32%	92.53%	92.02%	92.01%
#3	FCA		79.42%	78.31%	74.80%	76.34%
#4	LST		61.32%	60.38%	59.11%	61.52%
#5	FCA	TE	95.38%	95.31%	95.83%	95.45%
#6	FCA	PT	94.66%	94.01%	94.06%	93.15%
#7	LST	TE	96.25%	96.40%	96.38%	96.39%
#8	LST	PT	94.70%	95.18%	94.95%	95.15%

Average f-measure of each tag in each experiment is also observed and shown in Table VII. Similar to previous discussion, Experiment 7 gives the highest average f-measure compared to other experiments in every hyperparameter modification. Another similarity is shown by the result of Experiment 3 and Experiment 4 which are below the baseline experiments. However, the result from Experiment 5 and Experiment 6 yield show contrasting result compared to baseline experiments. Based on accuracy in Table VI, Experiment 5 and 6 are better than baseline experiments, however based on average f-measure in Table VII Experiment 5 and 6 are worse. The utilization of fixed character affix as additional features tends to improve the most majority tags but decline the minority. This cause overall accuracy of all tags is increased, but several f-measure of each tag are decreased, resulting lower average f-measure.

As in Table VI, experiments with morphological features

TABLE VII: Conducted experiment in this research and their corresponding average f-measure

Experiment	MF	Embedding	Weight size = 256 - 128		Weight size = 128 - 64	
			n_{batch} = 128	n_{batch} = 32	n_{batch} = 128	n_{batch} = 32
#1		TE	89.03%	90.26%	91.65%	91.24%
#2		PT	88.45%	88.99%	88.38%	87.12%
#3	FCA		60.02%	58.95%	50.30%	54.24%
#4	LST		33.06%	35.41%	30.26%	34.12%
#5	FCA	TE	90.60%	87.67%	86.62%	87.42%
#6	FCA	PT	85.98%	86.77%	83.85%	83.58%
#7	LST	TE	91.44%	92.00%	92.84%	92.47%
#8	LST	PT	90.09%	90.61%	90.20%	89.52%

but without word embedding vector such as Experiment 3 and Experiment 4 yield results that are far below the baseline experiment. In both of Experiment 3 and Experiment 4, the vector is formed from morphological features index like in Figure 3, whereas in the baseline experiment the vector is made with word embedding. This explains that morphological features unable to fully represent a word in a vector, even really distinct words with different tags can be represented similarly.

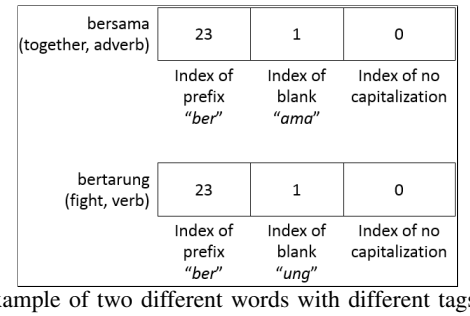

Fig. 3: Example of two different words with different tags as vector using light-stemmer as morphological features for affix

The demonstration of morphological feature only as vector is shown in Figure 3. There are two word vectors with light-stemmer as morphological features for affix, *bersama* (together, adverb) and *bertarung* (fight, verb). Based on Cosine similarity, the obtained similarity of those words is exactly 1. This is due they are the same vector, even though the words *bersama* and *bertarung* have nothing similar in the meaning nor the tag. Morphological feature such as affix is used to boost the tagging performance from the baseline model. However, morphological features can not be used as base representation of words like word embedding.

The utilization fixed character affix with the absent of word embedding as word vector is demonstrated in Figure 4. Similar to Figure 3, the word vectors in Figure 4 have high similarity based on Cosine similarity. However, the similarity of two vectors using fixed character affix is less similar than light-stemmer. This is because fixed character affix is more diverse than light-stemmer. Using Equation **??**, the similarity of the vectors in Figure 4 is 0.98, which is less than similarity in Figure 3.

Compared to light-stemmer or fixed character affix only as word representation, word embedding tends to give more

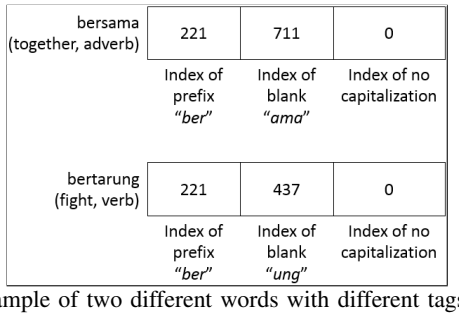

Fig. 4: Example of two different words with different tags as vector using fixed character affix as morphological features for affix

accurate similarity or dissimilarity. The example is written in Table VIII, which W.E. stands for word embedding. The similarity of two words vectors using both light-stemmer and fixed character affix tend to be high, even if the words are having a different tag. From example 1 until example 5, the high similarity is correct, because the two words have the same tag. However, example 6 until example 10 shows high similarity even the words are different in matter of tag. On the other hand, the similarity of two word vectors adapts based on the tag. If the tag of two words is the same, then the similarity is high and vice versa.

TABLE VIII: Example of similarity measure of vectors using word embedding, light-stemmer, and fixed character affix

#	Word 1	Word 2	Cosine similarity		
			W.E.	LST	FCA
1	*Ujar (VB)*	*Tidur (VB)*	0.761	1	0.395
2	*Dibuat (VB)*	*Tidur (VB)*	0.593	0.729	0.819
3	*Buku (NN)*	*Kursi (NN)*	0.536	0.781	0.446
4	*Lari (VB)*	*Melarikan (VB)*	0.666	0.746	0.936
5	*Saya (PRP)*	*Kami (PRP)*	0.929	1	0.964
6	*Bersama (JJ)*	*Bertarung (VB)*	0.057	1	0.983
7	*Perbuatan (NN)*	*Dibuat (VB)*	-0.168	0.964	0.888
8	*Perbaikan (NN)*	*Sebanyak (CD)*	-0.228	0.999	0.974
9	*Suara (NN)*	*Menyuarakan (VB)*	-0.067	0.781	0.912
10	*Bertemu (VB)*	*Pertemuan (NN)*	-0.221	0.994	0.481

Light-stemmer employs strict rules of affixes. This causes many words can not be applied in those rules and causes many words to have blank features as figured in Figure 3. On the other hand, fixed character affix does not use rules for extraction, so the indexed feature of fixed character affix is more than light-stemmer, as shown in Table V. This makes morphological features with fixed character affix more diverse than light-stemmer and makes two distinct words less similar, as stated before and illustrated in Figure 4. This is proved by comparing the unique word on testing data with the total combination of fixed character affix and light-stemmer as in Figure 5. There are 6050 (82.78% compared to total unique word) combination of affix using fixed character affix, and only 131 (2.17%) combination using light-stemmer.

The diversity of fixed character affix vector compared to light-stemmer affect the result, as can be seen in Table VI. The light-stemmer produces a better result than fixed character affix when the word embedding is exist, either with trained embedding (comparing Experiment 5 and 7) or with pre-trained word2vec (comparing Experiment 6 and 8). However,

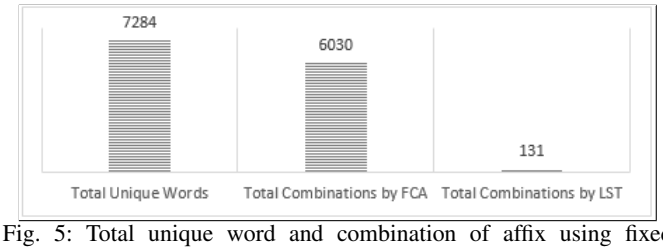

Fig. 5: Total unique word and combination of affix using fixed character affix and light-stemmer

the result is different if the word embedding is absent, as in Experiment 3 and Experiment 4, where fixed character affix as morphological features is better than light-stemmer when the word embedding vector is absent.

The experiments using pre-trained word2vec yield inconsiderable results compared to trained embedding in the model. This is clear due to word2vec is more general than the model's trained embedding. The model's trained embedding is particularly used in the tagger model alone, while the word2vec embedding could be used in other natural language tasks.

TABLE IX: Precision of tag CD compared to NN

Experiment	CD tagged as CD	CD tagged as NN
Experiment 1	3363	164
Experiment 5	3425	83
Experiment 7	3561	63

Even though the morphological feature does not significantly improve the tagger, the addition of morphological feature improves the precision of several tags significantly. Most of the inaccuracies occur in the other tags labeled as a noun (NN). In Experiment 1, the miss-tagged notably happens in cardinal (CD) tagged as NN, and verb (VB) tagged as NN. 164 CDs are tagged as NNs, and 235 VBs are tagged as NNs. In Experiment 5 and Experiment 7, the miss-tagged that happens in Experiment 1 are reduced. Table IX shows that Experiment 5 and Experiment 7 improve the accuracy of tag CD. The same thing happens to VB that is tagged to NN. Table X shows that experiments that employ morphological features like Experiment 5 and Experiment 7 reduce the number of miss-tagged of VB to one third.

TABLE X: Precision of tag VB compared to NN

Experiment	VB tagged as VB	VB tagged as NN
Experiment 1	6040	235
Experiment 5	6103	67
Experiment 7	6241	72

In Bahasa Indonesia, verb words are words that dominate the use of affixes, so that by classifying them by their affixes would be easier and more precise. This is also applicable to other tags, like cardinal and adverb. Most words like noun or adverb words will change to other tags like a verb when affixes are added. For instance, if word *jumlah* (amount) as noun is added with *se-* into *sejumlah* (a number of), it will become cardinal. Other examples are when word *merah* (red, adverb) changes to *memerah* (blush, verb) when prefix *me-* is

added, or *awan* (cloud, noun) changes to *berawan* (cloudly, adverb) by adding *ber-*. By utilizing affixes as morphological features, the tagger's precision for those tags gets better, and thus increases the overall accuracy, especially those tags are dominant according to Figure 1.

TABLE XI: Precision of tag NNP compared to NN

Experiment	NNP tagged as NNP	NNP tagged as NN
Experiment 1	5580	547
Experiment 5	6334	165
Experiment 7	6149	288

Besides VB and CD, proper noun (NNP) is another tag that is incorrectly tagged and most of the words are mistaken as NN. Even though NNP can be considered as a subclass of NN, the use of the morphological feature is able to improve the precision of tag NNP. In this case, capitalization as features contributes to consider one word is either NN or NNP. As in Table XI, the addition of morphological features especially capitalization reduces the number of incorrectly tagged of NNP words and increases the number of NNP tagged words.

VI. Conclusion

This research proposes the utilization of morphological features for BiLSTM architecture for PoS Tagging in Bahasa Indonesia. Two layers of BiLSTM with various sizes are employed as experiments to obtain which architecture is better For morphological features, this research attempts to use two kinds of morphological extractor: light-stemmer and fixed character affixes. Several modifications of models are conducted to gain the best tagger for Bahasa Indonesia, such as the representation of word vector, size of hidden neuron in each layer, and which morphological features are used.

This research shows that utilization of morphological features as addition for word embedding as input vector is yield better result than word embedding only, where light-stemmer gives better result than fixed character affix. However, if word embedding is put aside, and word index is used as word vector, fixed character affix gives better result than light-stemmer because fixed character affix can represent different words more diversely compared to light-stemmer. This research also shows that morphological feature is best used as an addition to word embedding as input feature, rather than replaced the input vector completely. The employment of self-trained embedding shows better results rather than utilizing the pre-trained model like word2vec.

References

[1] A. Dinakaramani, F. Rashel, A. Luthfi, and R. Manurung, "Designing an Indonesian part of speech tagset and manually tagged Indonesian corpus," in *2014 International Conference on Asian Language Processing (IALP)*. Singapore: IEEE, oct 2014, pp. 66–69.

[2] D. Jurafsky and J. H. Martin, *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. United Kingdom: Pearson Education UK, 2017.

[3] T. Brants, "TnT - A Statistical Part-of-Speech Tagger Thorsten," in *Proceedings of the sixth conference on Applied natural language processing* -, no. 1. Morristown, NJ, USA: Association for Computational Linguistics, 2000, pp. 224–231.

[4] K. Widhiyanti and A. Harjoko, "Pos tagging bahasa indonesia dengan hmm dan rule based," *Informatika: Jurnal Teknologi Komputer dan Informatika*, vol. 8, no. 2, 2012.

[5] F. Pisceldo, M. Adriani, and R. Manurung, "Probabilistic Part of Speech Tagging for Bahasa Indonesia," in *Proceedings of the 3rd International MALINDO Workshop*, Singapore, 2009.

[6] S. Raharjo, R. Wardoyo, and A. E. Putra, "Detecting proper nouns in indonesian-language translation of the quran using a guided method," *Journal of King Saud University - Computer and Information Sciences*, pp. 0–8, 2018.

[7] A. F. Abka, "Evaluating the use of word embeddings for part-of-speech tagging in Bahasa Indonesia," in *2016 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*. Jakarta: IEEE, oct 2016, pp. 209–214.

[8] Muljono, U. Afini, C. Supriyanto, and R. A. Nugroho, "The development of Indonesian POS tagging system for computer-aided independent language learning," *International Journal of Emerging Technologies in Learning*, vol. 12, no. 11, pp. 138–150, 2017.

[9] N. Zalmout and N. Habash, "Don't Throw Those Morphological Analyzers Away Just Yet: Neural Morphological Disambiguation for Arabic," *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 704–713, 2018.

[10] N. Reimers and I. Gurevych, "Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging," 2017.

[11] R. S. Yuwana, A. R. Yuliani, and H. F. Pardede, "On part of speech tagger for Indonesian language," in *2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, vol. 2018-Janua. Yogyakarta: IEEE, nov 2017, pp. 369–372.

[12] R. S. Yuwana, E. Suryawati, and H. F. Pardede, "On Empirical Evaluation of Deep Architectures for Indonesian POS Tagging Problem," *2018 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, pp. 204–208, 2018.

[13] B. Plank, A. Søgaard, and Y. Goldberg, "Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss," 2016.

[14] T. Horsmann and T. Zesch, "Do LSTMs really work so well for PoS tagging? A replication study," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 727–736.

[15] A. T. Nedjo, D. Huang, and X. Liu, "Automatic Part-of-speech Tagging for Oromo Language Using Maximum Entropy Markov Model (MEMM)," *Journal of Information and Computational Science*, vol. 11, no. 10, pp. 3319–3334, jul 2014.

[16] G. M. Wegari and M. Meshesha, "Parts of Speech Tagging for Afaan Oromo," *International Journal of Advanced Computer Science and Applications*, vol. 1, no. 3, pp. 1–5, 2011.

[17] S. D. Larasati, V. Kubo, and D. Zeman, "Indonesian morphology tool (MorphInd): Towards an Indonesian corpus," *Communications in Computer and Information Science*, vol. 100 CCIS, pp. 119–129, 2011.

[18] A. F. Wicaksono and A. Purwarianti, "HMM Based Part-of-Speech Tagger for Bahasa Indonesia," in *4th International MALINDO (Malay and Indonesian Language) Workshop*, no. January, Sarawak, 2010.

[19] M. Maimaiti, A. Wumaier, K. Abiderexiti, and T. Yibulayin, "Bidirectional Long Short-Term Memory Network with a Conditional Random Field Layer for Uyghur Part-Of-Speech Tagging," *Information*, vol. 8, no. 4, p. 157, 2017.

[20] D. Munandar, E. Suryawati, D. Riswantini, A. F. Abka, R. Wijayanti, and A. Arisal, "POS-tagging for non-english tweets: An automatic approach: (Study in Bahasa Indonesia)," in *2017 1st International Conference on Informatics and Computational Sciences (ICICoS)*, vol. 2018-Janua. IEEE, nov 2017, pp. 219–224.

[21] R. Setiawan, A. Kurniawan, W. Budiharto, I. H. Kartowisastro, and H. Prabowo, "Flexible affix classification for stemming Indonesian Language," *2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON 2016*, 2016.

[22] T. Mikolov, W.-T. Yih, and G. Zweig, "Linguistic Regularities in Continuous Space Word Representations," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 746–751.