On Proceedings of 4th International MALINDO (Malay and Indonesian Language) Workshop, 2nd August 2010

HMM Based Part-of-Speech Tagger for Bahasa Indonesia

Alfan Farizki Wicaksono
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
farizki@comlabs.itb.ac.id

Ayu Purwarianti
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
ayu@stei.itb.ac.id

Abstract—**In this paper, several methods are combined to improve the accuracy of HMM based POS tagger for Bahasa Indonesia. The first method is to employ affix tree which covers word suffix and prefix. The second one is to use succeeding POS tag as one of the feature for HMM. The last method is to use the additional lexicon (from *KBBI-Kateglo*) in order to limit the candidate tags resulted by the affix tree. The HMM model was built on 15000-tokens data corpus. In the experiment, on a 15% OOV test corpus, the best accuracy was 96.50% with 99.4% for the in-vocabulary words and 80.4% for the OOV(out of vocabulary) words. The experiment showed that the affix tree and additional lexicon is effective in increasing the POS tagger accuracy, while the usage of succeeding POS tag does not give much improvement on the OOV handling.**

Keywords : POS tagger, HMM method, affix tree, succeeding POS tag

I. Introduction

Part-of-Speech (POS) tagging is the process of assigning part-of-speech tags to words in a text [7, 20, 5]. A part-of-speech tag is a grammatical category such as verbs, nouns, adjectives, adverbs, and so on. Part-of-speech tagger is an essential tool in many natural language processing applications such as word sense disambiguation, parsing, question answering, and machine translation [12, 2].

Manually assigning part-of-speech tags to words is an expensive, laborious, and time consuming task, hence the widespread interest in automating the process. The main problems in designing an accurate automatic part-of-speech tagging are word ambiguity and Out-of-Vocabulary (OOV) word. Word ambiguity refers to different behaviour of words in different context. OOV word is a word that is unavailable in the annotated corpus.

There are several approaches on POS tagging research, i.e. rule based, probabilistic, and transformational based approach. Rule based POS tagger assigns a POS tag to a word based on several manually created linguistic rules [8]. Probabilistic approach determines the most probable tag of a token given its surrounding context, based on probability values obtained from a manually tagged corpus [7]. The transformational based approach combines rule based and probabilistic approach to automatically derive symbolic rules from a corpus [3].

Bahasa Indonesia is the national language of Indonesia which is spoken by more than 222 millions people [11]. It is widely used in Indonesia to communicate in school, government offices, daily life, etc. Bahasa Indonesia became the formal language of the country, uniting its citizens who speak different languages. Bahasa Indonesia has become the language that bridges the language barrier among Indonesians who have different mother-tongues. Even though, the availability of language tools and resource for research related to Bahasa Indonesia is still limited. One language tool that is not yet commonly available for Bahasa Indonesia is POS tagger system.

There are relatively little works on POS tagging system for Bahasa Indonesia. Pisceldo et al [7] tried to develop POS tagger for Bahasa Indonesia using Maximum Entropy model and Conditional Random Field (CRF). The best performance of Indonesian POS tagger reached in [3] is 97.57%. Triastuti [3] also developed POS tagger for Bahasa Indonesia using CRF, Transformation Based approach, and combining between CRF – Transformation based approach. The best performance in [3] reached 90.08%. Sari et al. [24] also applied Brill's transformational rule based approach in developing a POS tagger for Bahasa Indonesia on a limited tagset trained on a small manually and contextual features, they showed that the method obtained an accuracy of 88%.

However, there is no deep study about developing POS tagger Bahasa Indonesia by using Hidden Markov Model (HMM). Hidden Markov Model is the established probabilistic method for automatic POS tagger. Several languages have adapted the HMM method in building the automatic POS tagger [16, 17, 6, 1]. A POS tagger with HMM method was proved to have better running time than any other probabilistic methods [14]. In this study, we report our attempt in developing a HMM based part-of-speech tagger for Bahasa Indonesia[1]. We tried to make improvements such as using affix tree to predict emission probability vector for OOV words and utilizing information from dictionary lexicon (*KBBI-Kateglo*) and succeeding POS tag.

II. Methods

A. *Underlying Model*

Both first order (bigram) and second order (trigram) Hidden Markov Model are used for developing the tagger system. Hidden states of the model represent tags and observation

[1] We provide API (Application Programming Interface) of our tagger. This API is available at http://students.itb.ac.id/home/alfan_fw@students.itb.ac.id/IPOSTAgger or http://www.informatika.org/~ayu/IPOSTAgger

states represent words. Transition probabilities depend on the states, thus pairs of tags. Emission probabilities only depend on current tag. Equation (1) is used to find the best POS tag sequence in first order HMM (bigram) case.

$$t_{1-n} = \arg\max_{t_1 \ldots t_n} P(t_1) \times \prod_{i=2}^{n} P(t_i \mid t_{i-1}) \times \prod_{i=1}^{n} P(w_i \mid t_i) \quad \text{.................} (1)$$

Fort the trigram case, the calculation is in equation (2).

$$t_{1-n} = \arg\max_{t_1 \ldots t_n} P(t_1) \times P(t_2 \mid t_1) \times \prod_{i=3}^{n} P(t_i \mid t_{i-1}, t_{i-2}) \times \prod_{i=1}^{n} P(w_i \mid t_i) \quad \text{.....} (2)$$

t_{1-n} is the best POS tag sequence for the input words, $w_1 \ldots w_n$ are sequence of input word, and $t_1 \ldots t_n$ are elements of the tag set. $P(t_1)$ in equations (1) and (2) are not an ordinary unigram probability. $P(t_2 \mid t_1)$ in equation (2) is also not an ordinary bigram probability. $P(t_1)$ and $P(t_2 \mid t_1)$ in both equations are conditioned on being the first token in a sentence. In the implementation, we add a dummy POS tag <START> before the first word (two dummies for trigram case). Therefore, $P(t_1)$ in equation (1) is calculated using $P(t_1 \mid start)$. And also, $P(t_1)$ and $P(t_2 \mid t_1)$ in equation (2) are calculated respectively using $P(t_1 \mid start, start)$ and $P(t_2 \mid t_1, start)$.

Transition and emission probabilities are estimated from a tagged corpus. We use maximum likelihood probabilities (MLE) method to estimate this value.

Another issue is on the estimation of transition probability: $P(t_i \mid t_{i-1})$ or $P(t_i \mid t_{i-1}, t_{i-2})$ where the particular bigram or trigram in not available in the training corpus. It causes sparse-data problem [2]. Here, we use smoothing method to solve this problem. For trigram case, we use deleted linier interpolation method [2] to estimate the zero trigram transition probability of $P(t_i \mid t_{i-1}, t_{i-2})$ such as shown in equation (3).

$$P(t_i \mid t_{i-1}, t_{i-2}) = \lambda_1 P'(t_i) + \lambda_2 P'(t_i \mid t_{i-1}) + \lambda_3 P'(t_i \mid t_{i-1}, t_{i-2}) \quad \text{.....} (3)$$

In equation 3, $\lambda_1 + \lambda_2 + \lambda_3 = 1$; P is probability distribution and P' is the maximum likehood estimation of the probability. For bigram case, we use Jelinec-Mercer smoothing [21] to estimate $P(t_i \mid t_{i-1})$ such as in equation (4).

$$P(t_i \mid t_{i-1}) = \lambda P'(t_i \mid t_{i-1}) + (1-\lambda) P'(t_i) \quad \text{.....} (4)$$

B. *Affix Tree for Handling OOV Words*

We use affix tree to obtain emission probability vector of OOV word like Schmid did [18, 19] in his research. We make an adaptation of Schmid's method for Bahasa Indonesia. We build three types of tree which cover the capitalized words, the uncapitalized words, and the cardinal words (*ke-5, 100,* etc). For example, if the OOV word is a capitalized word, the searching process only involves the tree constructed from capitalized word in lexicon.

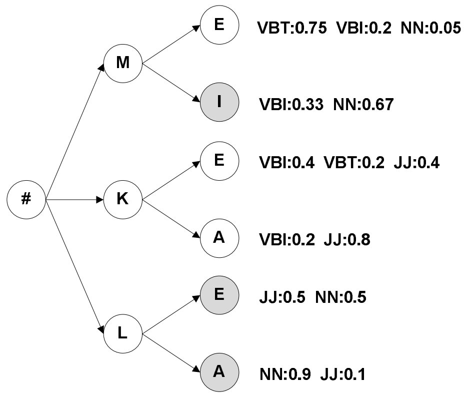

FIGURE 1. A SAMPLE PREFIX TREE OF LENGTH 2

As explanation in [18, 19], the affix tree is automatically built from training corpus. Affix tree is constructed from the affixes (prefix or suffix) of length 3 of all words in the corpus. Tag frequencies are counted for all affixes and stored at the corresponding tree nodes. Then, an information measure $I(S)$ is calculated for each node of the tree such as in equation (5).

$$I(S) = -\sum_{pos} P(pos \mid s) \times \log_2 P(pos \mid s) \quad \text{.......} (5)$$

Here, S is the affix which correspond to the current node and $P(pos \mid s)$ is the probability of tag pos given a word with affix s . Using this information measure, the affix tree is pruned. For each leaf, the weighted information gain $G(aS)$ is calculated such as in equation (6).

$$G(aS) = F(aS) \times (I(S) - I(aS)) \quad \text{............} (6)$$

Equation (6) is suitable for suffix case. To apply it to prefix tree, the Sa is employed to replace the aS . Here, S is the affix (prefix or suffix) of the parent node, aS (Sa) is the suffix (prefix) of the current node, and $F(aS)$ ($F(Sa)$) is the frequency of suffix aS (prefix Sa).

If the information gain at some leaf of the affix tree is below a given threshold [2], the leaf is deleted. The deleted node will become default node of its parent node (the shaded node in figure 1 is the sample of default node). If the default node is only the remaining node, the parent node becomes a leaf and is also checked whether it can be deleted or not.

To illustrate the process, consider the following example (prefix case), where *m* is the prefix of the parent node, *mi* is the prefix of one child node, and *me* is the prefix of the other child node. Sample tag frequencies of the nodes are given in table 1.

TABLE 1. SAMPLE TAG FREQUENCIES

tag	*Prefix m*	*Prefix me*	*Prefix mi*
VBT	75	75	0
VBI	20	19	1
NN	7	5	2
total	102	99	3

The information measure for parent node is calculated by equation (7).

$$I(m) = -\frac{75}{102}\log_2\frac{75}{102} - \frac{20}{102}\log_2\frac{20}{102} - ... \approx 1.0523 \quad \text{... (7)}$$

The corresponding values for the child nodes are 0.97 for *me* and 0.91 for *mi*. Then, we determine the weighted information gain at each of child nodes such as in equation (8) and equation (9).

$$G(me) = 99(1.05 - 0.97) = 7.92 \quad \text{..........................(8)}$$

$$G(mi) = 3(1.05 - 0.91) = 0.42 \quad \text{...............................(9)}$$

Value of $G(me)$ is above a threshold of 3. On the other hand, value of $G(mi)$ is below a threshold. And therefore, node *mi* should be deleted from the tree and becomes a default node.

The tree is searched during a lookup along the path, where the nodes are annotated with the letters of the word affix in reversed order. If a leaf is reached at the end of the path, the corresponding tag probability vector is returned. If no matching child node is found at some node on the path, tag probability vector returned is from the summation of all tag frequency vectors from all default nodes. If no default node exists, our tagger system just return the current tag probability vector.

In our tagger, there are 3 types of tree configuration used for handling OOV case. Each of these configurations was tested to find the best configuration for Indonesian POS tagger system:

1. *Prefix Tree*
   This configuration uses the first N character of word to create the tree. Prefix of OOV word is used to obtain emission probability vector of OOV word.
2. *Suffix Tree*
   This configuration uses the last N character of word to create the tree. Suffix of OOV word is used to obtain emission probability vector of OOV word.
3. *Prefix – Suffix Tree*
   This configuration combines prefix and suffix feature of word. The result of this configuration is the combination of emission probability vector produced by prefix tree and suffix tree. Each corresponding vector entry in both trees is summed and renormalized in order to be combined.

C. Succeeding POS Tag

Basically, for a HMM based POS tagger, succeeding POS tag is unknown when decoding process is being done (using viterbi algorithm [2, 21]). When decoding process reaches the *n*th *word* in the sentence, the tagger does not know the POS tag of *(n+1)*th *word* (succeeding POS tag).

Nakagawa et al [13] tried to develop POS tagger system using SVM and two-pass method. This two-pass method enables us to use succeeding POS tag. Our work tried to use this two-pass method for developing Indonesian POS tagger. We did some adaptations in order this two-pass method is able to be implemented in our HMM-based system. Below is the process of our two-pass method :

1. *first pass*
   In the first pass, POS tags of the input words are predicted ordinarily without using the information of succeeding POS tag. This process is done using equation (1) or (2). Affix tree is used to predict the emission probability vector of OOV words.

2. *second pass*
   In the second pass, the tagging process is repeated from the first input word. This time, the tagger uses information from succeeding POS tag if current word is OOV in decoding process. The tagger does not use succeeding POS tag if the current word is known. The POS tag of succeeding word (succeeding POS tag) is obtained from the first pass.

In the first pass, the tagger just calculate the probability $P(t_i \mid t_{i-1})$ (bigram case) and $P(t_i \mid t_{i-1}, t_{i-2})$ (trigram case) for all states in the decoding process. Meanwhile, in the second pass, if the tagger meets with OOV word, the transition probability $P(t_i \mid t_{i-1})$ is replaced with $P(t_i \mid t_{i-1}, t_{i+1})$ ($P(t_i \mid t_{i-1}, t_{i-2})$ with $P(t_i \mid t_{i-1}, t_{i-2}, t_{i+1})$). t_{i+1} is obtained from the first pass. It means that the transition probability is not only conditioned on

[2] We used a gain of threshold 3

being preceding POS tag but also succeeding POS tag. The tagger uses maximum likehood estimation to determine the value of $P(t_i \mid t_{i-1}, t_{i+1})$ or $P(t_i \mid t_{i-1}, t_{i-2}, t_{i+1})$.

D. *Lexicon from KBBI-Kateglo*

We use lexicon from *KBBI* - *Kateglo* [9, 10] for reducing entry in emission probability vector produced by affix tree. Different POS tag set between corpus and *KBBI – Kateglo* is the main problem. POS tag set in tagged corpus is the specialized form of POS tag set in *KBBI* and *Kateglo*. Therefore, we make a table called *category table* that maps POS in *KBBI* - *Kateglo* into POS in corpus. Table 2 shows the example of category table for some POS tags in *KBBI – Kateglo*.

TABLE 2. SAMPLE OF CATEGORY TABLE

NO	***POS tag KBBI***	***POS tag corpus (specialized)***
1	NN	NN NNP NNG
2	CD	CDO CDC CDP CDI
3	PR	PRP WP PRN PRL

If POS tag **A** (corpus) in emission probability vector is not the specialized form from one of the POS tag (*KBBI – Kateglo*) produced from lexicon for corresponding word, this POS tag **A** will be deleted from the probability vector.

For example, there is OOV word *bisa* in input sequence. Emission probability vector produced by affix tree for this word is {MD:0.05, NN:0.1, VBI:0.6, CC:0.25}. POS tag set produced from *KBBI – Kateglo* lexicon is {MD, NN}. VBI is not specialized form for either MD or NN, CC is not either. Therefore, VBI and CC are deleted from the probability vector. Our current probability vector is {MD:0.05, NN:0,1}.

III. Experiments

A. *Experimental Data*

Experiments were performed using small hand tagged Indonesian corpus which consists of 35 POS tags (table 3). The tagset is modification from tagset used in [23, 7]. The corpus was developed as the correction and modification of PAN Localization corpus[3] for Bahasa Indonesia [23, 15]. Our small tagged corpus was divided into 2 sub corpora, one for training (~12000 words) and one for testing (~3000 words). We did experiments in 3 different testing corpus (each corpus contains ~3000 words). First testing corpus contains 30% unknown words. Second testing corpus contains 21% unknown words. The last testing corpus contains 15% unknown words.

TABLE 3. TAGSET USED FOR BAHASA INDONESIA

NO	***POS***	***POS Name***	***Example***
1	OP	Open Parenthesis	({[
2	CP	Close Parenthesis	)}]
3	GM	Slash	/
4	;	Semicolon	;
5	:	Colon	:
6	"	Quotation	" '
7	.	Sentence Terminator	. ! ?
8	,	Comma	,
9	-	Dash	-
10	...	Ellipsis	...
11	JJ	Adjective	Kaya, Manis
12	RB	Adverb	Sementara, Nanti
13	NN	Common Noun	Mobil
14	NNP	Proper Noun	Bekasi, Indonesia
15	NNG	Genitive Noun	Bukunya
16	VBI	Intransitive Verb	Pergi
17	VBT	Transitive Verb	Membeli
18	IN	Preposition	Di, Ke, Dari
19	MD	Modal	Bisa
20	CC	Coor-Conjunction	Dan, Atau, Tetapi
21	SC	Subor-Conjunction	Jika, Ketika
22	DT	Determiner	Para, Ini, Itu
23	UH	Interjection	Wah, Aduh, Oi
24	CDO	Ordinal Numerals	Pertama, Kedua
25	CDC	Collective Numerals	Bertiga
26	CDP	Primary Numerals	Satu, Dua
27	CDI	Irregular Numerals	Beberapa
28	PRP	Personal Pronouns	Saya, Kamu
29	WP	WH-Pronouns	Apa, Siapa
30	PRN	Number Pronouns	Kedua-duanya
31	PRL	Locative Pronouns	Sini, Situ, Sana
32	NEG	Negation	Bukan, Tidak
33	SYM	Symbols	@#$%^&
34	RP	Particles	Pun, Kah
35	FW	Foreign Words	Foreign, Word

In the first experiment, the HMM tagger using Affix Tree was compared to a baseline tagger which was trained and tested on the same data. The baseline POS tagger employed the HMM bigram tagger. This baseline method tagged tag NN (Noun) for the OOV words.

In another experiment, we compared several configurations for the HMM based POS tagger. The configuration consists of the N-gram model, the Affix Tree (prefix, suffix, prefix-suffix), and so on. It examined the effectiveness of the Succeeding POS tag and the Lexicon from *KBBI – Kateglo*.

[3] This PAN Localization project corpus is available at http://panl10n.net/english/OutputsIndonesia2.htm

B. Experimental Result

Table 4,5,6 show the performance of the POS tagger in 3 different types of testing corpus. The performance is showed in format $\frac{Overall\ Accuracy}{(Known\ Word\ Acc / Unknown\ Word\ Acc)}$.

The result shows that Affix Tree is successful in increasing the performance of OOV words tagging. It improved the accuracy of OOV words tagging by about 24%-38% from the baseline. From the experimental result table, prefix configuration gives better improvement than using suffix or prefix-suffix configuration.

Lexicon from *KBBI – Kateglo* is an important resource for handling OOV. It gives better tagging accuracy than only using affix tree. The difference of OOV word tagging accuracy between the tagger using only affix tree and affix tree+lexicon is about 2%-4% (1%-2% for overall tagging accuracy). Not all POS tags produced by affix tree appropriate with the corresponding word. This phenomena happens because affix tree method only uses information from prefix or suffix of corresponding word. Therefore, we delete inappropriate POS tags from the vector using this lexicon. Intuitively, this approach can raise the performance and the test had proved it.

On the other hand, information from succeeding POS tag (bipass) does not give much improvement on the tagging accuracy. Many cases in table 4,5,6 show that succeeding POS tag decrease the tagging accuracy. But, there are some cases where using information from succeeding POS tag successfully increases the tagging accuracy. Word *berdiri* in the sentence below is the example.

saya ***berdiri*** *di jalan .*

From the sentence, the POS tag of word *berdiri* has to be VBI (intransitive verb). Furthermore, the tagger correctly tags the known words {*saya*, *di*} by {PRP, IN} because there is no ambiguity in those words. If the tagger just uses affix tree, the tagger will choose VBT (transitive verb) as the POS tag of word *berdiri* because the probability of bigram <PRP-VBT> is bigger than bigram <PRP-VBI> in the training corpus. If the tagger uses succeeding POS tag information, the computation is focused on trigram <PRP-VBT-IN> and <PRP-VBI-IN>. In this case, the POS tag IN is the succeeding POS tag. From the training corpus, the probability of trigram <PRP-VBT-IN> is less than trigram <PRP-VBI-IN>. Therefore, the tagger chooses VBI as the POS tag.

TABLE 4. RESULT ON 15% OOV TESTING CORPUS

NO	***Configuration***	***Affix Tree Configuration***		
		PREFIX	***SUFFIX***	***PRE-SUFF***
1	Baseline	*90.65% (99.42%/42.34%)*		
2	Bigram	95.67% (99.43%/75.00%)	94.32% (99.39%/66.44%)	95.36% (99.43%/72.97%)
3	Trigram	95.29% (99.18%/73.87%)	94.29% (99.22%/67.12%)	95.01% (99.22%/71.85%)
4	Bigram+succeeding POS	95.57% (99.43%/74.32%)	94.56% (99.35%/68.24%)	95.36% (99.43%/72.97%)
5	Trigram+succeeding POS	94.94% (99.02%/72.52%)	94.04% (99.06%/66.44%)	94.70% (99.02%/70.95%)
6	Bigram+Lexicon	96.30% (99.43%/79.05%)	95.01% (99.43%/70.72%)	96.23% (99.43%/78.60%)
7	Trigram+Lexicon	95.98% (99.18%/78.38%)	94.94% (99.26%/71.17%)	95.95% (99.26%/77.70%)
8	Bigram+succ+Lexicon	96.36% (99.43%/79.50%)	95.36% (99.43%/72.97%)	96.50% (99.43%/80.41%)
9	Trigram+succ+Lexicon	95.78% (99.02%/77.93%)	95.08% (99.06%/73.20%)	95.91% (99.06%/78.60%)

TABLE 5. RESULT ON 21% OOV TESTING CORPUS

NO	***Configuration***	***Affix Tree Configuration***		
		PREFIX	***SUFFIX***	***PRE-SUFF***
1	Baseline	*85.92% (99.10% / 37.72%)*		
2	Bigram	93.27% (99.18%/71.64%)	92.40% (99.15%/67.71%)	93.01% (99.18%/70.42%)
3	Trigram	93.12% (99.29%/70.56%)	92.31% (99.07%/67.57%)	93.12% (99.22%/70.83%)
4	Bigram+succeeding POS	93.41% (99.22%/72.18%)	92.54% (99.15%/68.39%)	93.36% (99.18%/72.05%)
5	Trigram+succeeding POS	93.04% (99.18%/70.56%)	92.22% (99.15%/66.89%)	93.07% (99.18%/70.69%)
6	Bigram+Lexicon	93.82% (99.18%/74.22%)	93.27% (99.18%/71.64%)	93.97% (99.18%/74.90%)
7	Trigram+Lexicon	93.59% (99.26%/72.86%)	93.33% (99.18%/71.91%)	94.00% (99.22%/74.90%)
8	Bigram+succ+Lexicon	93.97% (99.18%/74.90%)	93.56% (99.18%/73.00%)	94.46% (99.18%/77.20%)
9	Trigram+succ+Lexicon	93.59% (99.15%/73.27%)	93.24% (99.11%/71.78%)	94.06% (99.07%/75.71%)

TABLE 6. RESULT ON 30% OOV TESTING CORPUS

NO	***Configuration***	***Affix Tree Configuration***		
		PREFIX	***SUFFIX***	***PRE-SUFF***
1	Baseline	*83.28% (99.01%/47.21%)*		
2	Bigram	89.32% (99.05%/67.01%)	88.98% (99.05%/65.88%)	89.84% (99.10%/68.61%)
3	Trigram	89.32% (99.01%/67.11%)	89.06% (98.97%/66.35%)	89.78% (99.01%/68.61%)
4	Bigram+succeeding POS	88.86% (99.01%/65.60%)	88.35% (99.05%/63.81%)	89.55% (99.10%/67.67%)
5	Trigram+succeeding POS	88.66% (98.89%/65.22%)	88.09% (98.85%/63.43%)	88.92% (98.97%/65.88%)
6	Bigram+Lexicon	90.32% (99.01%/70.41%)	90.15% (99.05%/69.75%)	91.18% (99.01%/73.23%)
7	Trigram+Lexicon	90.32% (99.05%/70.31%)	90.12% (99.01%/69.75%)	91.30% (99.05%/73.52%)
8	Bigram+succ+Lexicon	90.04% (98.93%/69.65%)	89.81% (99.05%/68.61%)	91.01% (99.01%/72.67%)
9	Trigram+succ+Lexicon	90.15% (98.85%/70.22%)	89.87% (98.97%/68.99%)	90.78% (98.93%/72.10%)

Unfortunately, this kind of case is very rare in the testing corpus. In addition, there are many chances that succeeding POS tag is incorrectly tagged in the first pass, especially when the succeeding word is OOV word or the OOV words appear consecutively in the testing data. These all will lead to tag the current word incorrectly and decrease the tagging accuracy. Roth and Zelenko also did similar experiment and reported the same thing with us [22]. They used a dictionary (the most frequent POS tag for each word) to get succeeding POS tag. They reported that about 2% of accuracy decrease is caused by incorrectly attached POS tags by their method [13, 22].

IV. CONCLUSIONS

Our research focused on developing Indonesian POS tagger using Hidden Markov Model and evaluating the performance of the system for each configuration. The result showed that HMM based POS Tagger for Bahasa Indonesia depends on the OOV handling method used (~99.4% accuracy for non-OOV words). If the OOV handling method is good at disambiguating OOV words, the overall performance is very high.

Affix tree and lexicon from *KBBI – Kateglo* is useful to improve the tagging accuracy for tagger on Indonesian data, especially for OOV words. Prefix is the best configuration for the affix tree since it gives better accuracy than other configurations in many cases. The best tagging accuracy for OOV words achieved ~80%. The overall tagging accuracy for each testing data achieved 91.30%(30% OOV), 94.46%(21% OOV), and 96.50%(15% OOV).

On the other hand, succeeding POS tag does not give much improvement on OOV handling. Many cases showed that succeeding POS tag decreases the tagging accuracy. Maybe, we have to use succeeding POS tag in other strategy because there are some cases where succeeding POS tag helps the tagging performance. For example, we can add some rules to decide when to use the succeeding POS tag.

In our next research, we will develop a larger Indonesian corpus. We will also conduct study on Indonesian POS tagger with another approach to investigate the best method for Indonesian POS tagger.

ACKNOWLEDGMENT

We would like to thank Dr. Hammam Riza from BPPT – Indonesia for allowing and giving access to his training data, and also for his useful guidance.

REFERENCES

[1] Azimizadeh, Ali. Mehdi, Mohammad. Rahati, Saeid. 2008. "Persian part of speech tagger based on Hidden Markov Model". JADT 2008 : 9es Journées internationales d'Analyse statistique des Données Textuelles.

[2] Brants, Thorsten. 2000. *"TnT - A Statistical Part-of-Speech Tagger"*. Proceedings of the sixth conference on Applied Natural Language Processing (2000) 224.231.

[3] Brill, E.*"A simple rule-based part-of-speech taggCher"*. In: Proceedings of the Third Conference on Applied Natural Language Processing (ANLP '92), Trento, Italy (1992) 152–155.

[4] Chandrawati, Triastuti. 2008. *"Indonesian Part-of-Speech Tagger based on Conditional Random Fields and Transformation based learning methods"*.Undergraduate thesis. Depok:Fasilkom University of Indonesia.2008.

[5] Cutting, Doug, et al. *A Practical Part-of-speech Tagger*. Xerox Palo Alto Research Center. In Proceding of the third conference on applied Natural Language Processing page 133-140. 1992.

[6] Dandapat, Sandipan., Sarkar Sudeshna. Part-of-Speech Tagging for Bengali with Hidden Markov Model. In Proceedings of the NLPAI Machine Learning Contest. Mumbai, India, 2006.

[7] Femphy Pisceldo, Manurung, R., Adriani, Mirna. Probabilistic Part-of-Speech Tagging for bahasa Indonesia. Third International MALINDO Workshop, colocated event ACL-IJCNLP 2009, Singapore, August 1, 2009.

[8] G. Rubin. B. Greene .1971. "Automatic Grammatical Tagging of English". Technical Report, Department of Linguistics, Brown University, Providence, Rhode Island.

[9] *Kamus Besar Bahasa Indonesia*. http://www.pusatbahasa.diknas.go.id, access date 12 April 2010.

[10] Kateglo. Dictionary, Thesaurus, and Glosarium for bahasa Indonesia. http://www.bahtera.org/kateglo. access date 12 April 2010.

[11] Lewis, M. Paul (ed.). 2009. *"Ethnologue: Languages of the World, Sixteenth edition"*. Dallas, Tex : SIL International.

[12] Manurung, Ruli. Adriani, Mirna. 2008. *"A survey of bahasa Indonesia NLP research conducted at the University of Indonesia"*.Second MALINDO Workshop. Selangor, Malaysia: 12-13 June 2008.

[13] Nakagawa, T., Kudo, T., & Matsumoto, Y. *Unknown word guessing and part*-of- *speech tagging using support vector machines.* In Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium. 2001.

[14] Nguyen, Nam., Guo, Yunsong. 2007. Comparisons of Sequence Labeling Algorithms and Extensions. International Conference on Machine Learning.

[15] PANL10N1(PAN Localization Project), http://www.panl10n.net, access date 12 April 2010.

[16] Padr´o M. and Padr´o L. Developing Competitive HMM POS Taggers Using Small Training Corpora. *Estal*. 2004.

[17] Pajarskaite, Giedre, et al,. Designing HMM based Part-of-Speech Tagging for Lithuanian Language. INFORMATICA vol 15, no 2, page 231-242. 2004.

[18] Schmid, Helmut. Probabilistic Part-of-Speech Tagging using Decision Tree. *Proceedings of International Conference on New Methods in Language Processing*. September 1994.

[19] Schmid, Helmut. 1995. "Improvements in Part-of-Speech Tagging with an Application to German". Proceedings of the ACL SIGDAT-Workshop. March 1995.

[20] Scott M.T. M.P. Harper. 1999. "A second-order Hidden Markov Model for part-of-speech tagging". Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics. pp: 175-182.

[21] Wibisono, Yudi. HMM for Sentece Compression. Graduate Thesis. Institute of Technology Bandung. 2008.

[22] D. Roth and D. Zelenko. 1998. Part-of-Speech Tagging Using a Network of Liniear Separators. In Proceedings of the joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (ACL/COLING-98),pages 1136-1142.

[23] Adriani, Mirna. Riza, Hammam. Local Language Computing: Development of Indonesian Language Resources and Translation System. 2008.

[24] Sari, Syandra, Herika Hayurani, Mirna Adriani, and Stephane Bressan. Developing Part-of-Speech Tagger for Bahasa Indonesia Using Brill Tagger. The International Second MALINDO Workshop, 2008.