STEMMER FOR "BASA SUNDA"

Danang Junaedi[1)], I. Ovyawan Herlistiono[2)], dan Dewis Akbar[3)]

Universitas Widyatama[1)], STTIS Bandung[2)], ITB
danangjunaedi@gmail.com [1)], i_ovyawan_h@yahoo.com [2)], dewis.akbar@gmail.com [3)]

Abstract

Stemming usually be used to remove suffixes from given word(s). In this paper, we used stemming algorithm to remove suffixes from word in "*basa Sunda*", the second biggest local language in Indonesia. Although the "*basa Sunda*" is common language in Indonesia especially in Jawa Barat, we didn't find any reference about it. We begin our research by develop a software for the stemming process in order to begin milestone project of the Natural Language Processing for basa Sunda.

Keywords: Basa Sunda, Stemmer, Algorithm, Natural Language Processing

1. Introduction

Basa Sunda is a local language used by the Sundanese people in Jawa Barat, a province of Indonesia. *Basa Sunda* is the second most popular local language in Indonesia after the *bahasa Jawa*. In Jawa Barat, *Basa Sunda* taught from elementary school to the university level of education. There are even the *basa Sunda* department in the *Universitas Padjadjaran* and the *Universitas Pendidikan Indonesia*. This effort was brought up in order to prevent *basa Sunda* from extinction.

Natural Language Processing (NLP) is an application of the Artificial Intelligence field. In NLP, a computer is trained to understand human language. The first step of this process was stemming. Stemming Algorithm works by removing suffixes in order to bring back the original form of the word.

Even though *basa Sunda* is the second biggest local language in Indonesia, we can't find any NLP reference about it. So, we thought it is worth to begin our milestone of NLP application in *basa Sunda*. Hence it should be enough for originality.

2. The Problem

We begin by identifying our problem, which are: first, what kind of factors is used to build a suffixed word from original word in *basa Sunda*. And the second question is how to bring back suffixed word into its original word.

3. The Research.

3.1. The Purpose.

1. Understanding the characteristics and techniques prior to build a stemmer for *basa Sunda*.
2. Building a Stemmer specific for *basa Sunda*.
3. Calculating the accuracy of the stemmer built.

3.2. The Research Handicap

In this research we bound to covered this following:

1. The derivation word process just for the suffixes that build a specific word only.
2. A dictionary was used as the lookup function in the stemming process.
3. The suffix is bounded to affix.

3.3. The Methodology

Adriani[1] build a stemmer for *bahasa Indonesia* by removing suffixes with the accuracy up to 95.45%. *Basa Sunda* in fact had a similar form with the *bahasa Indonesia*, so we decided to used the same approach to build our *basa Sunda* Stemmer.

The process of building a stemmer for *basa Sunda* was:

1. Studying morphemically process.
2. Design derivation word rules back to its original form.
3. Design a derivation recoding rules back to its original form.
4. Collecting original form of words in *basa Sunda*.
5. Implement the Stemmer for *basa Sunda*.
6. Collecting *basa Sunda* corpus.
7. Test the stemmer using the corpus.
8. Calculating performance of the stemmer.

4. Supporting Theories

Word in *basa Sunda* was built by element(s) called *morfem*. And it could contained one or more *morfem* in one word. *Morfem* is the smallest element that define a meaning. *Morfem* can also be called as word builder that define a meaning both for lexical and grammatical. For example, the word "meuli" contain two lexical *morfem*, "beuli" which mean trade a merchandise with money and the grammatical N (which is turn into 'm') that mean 'doing'.

Word with just one *morfem* is called *kecap salancar*. The word that contains two or more *morfem* called kecap rékaan. Morfem that stood by itself in pronunciation or sentence called the free morfem. Every free *morfem* in *basa Sunda* is indeed a word.

There are three form of *morfem* in *basa Sunda*.

1. The free *morfem*. Free *morfem* who had a lexical meaning called the lexical free morfem (the original word). Examples of this morfem are *kuring, baju*, and *kamari*.
2. The half free *morfem*, which means the *morfem* who had both lexical and grammatical meaning. The half free *morfem* who had the lexical meaning and become the foundation of building a word is called *bakal kecap* (base word). Examples of the base word are *cokot* and *sepak*.
   Meanwhile the half free *morfem* who had a lexical meaning and attached to another *morfem* and also had a pair that builds a free *morfem* is called *klitik*. Examples of this *morfem* are *pun+*, *sim+* and *tuang+*.
   The half free *morfem* who had grammatical meaning and following some word and become of a sentence is called *partikel* or *kecap pancen*. Example: *di, ka, jeung, arék, pikeun* and *jeung teh*.
3. *Morfem* kauger is a *morfem* who cannot stand by itself in the sentence and it has to follow by another *morfem*. *Morfem kauger* who had a lexical meaning and becoming a base of word developing called *cakal*. And *morfem* kauger who had grammatical meaning and attached to base *morfem* called *rarangken*.

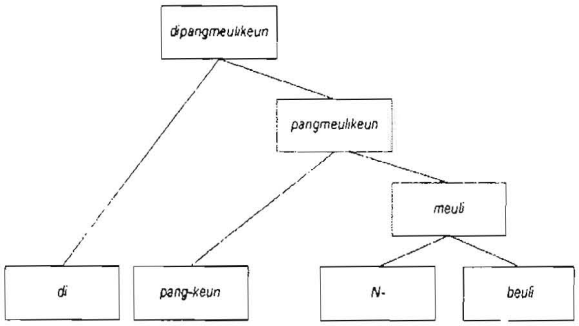

Figure 1. "Dipangmeulikeun" Word Derivation

There are two kinds of word, base word and original word. The original word is the form of single word that becomes base word. It means that every base word is indeed original word. For example the word '*dipangmeulikeun*' (Figure 1) which is based on the word '*meuli*'. The word '*meuli*' itself is build from the word '*beuli*'. We can see that in the word '*pangmeulikeun*' contain suffixes that attached to the word '*beuli*'.

Prefix	Infix	Suffix	*Barungan*	Rajegan (combination)
ba-	-ar-	-an	ka-an	di- + -ar-
di-	-in-	-eun	kapi-	di- + -an-
ka-	-um-	-keun	pa-an	di- + -ar- + -an
Ti-		-na	pang-na	di- + -keun
N-		-(N)ing	pang-keun	di- + -ar- + -keun
pa-			-pi-eun	di- + pi-
pang-			pika-	di- + pika-
per-			pika-eun	di- + pang- + -keun
pi-			sa-eun	di- + pang- + n- + -keun
sa-			sa-na	di- + pang-N- + -ar- + -keun
sang-				di- + pang-N- + -ar- + -an + -keun
si-				N- + -ar-
ti-				N- + -an
ting-				N- + -ar- + -an
				N- + -keun
				N- + -ar- -keun
				N- + pi-
				N- + pika-
				N- + pang- + -keun
				ka- + -keun
				pa- + n-
				pang- + dipika- + -na
				pang- + n- + pika- + na
				ting- + -ar-

Figure 2. *Affix* In Basa Sunda

In *basa Sunda*, suffixes can be affix, *proleksem*, *formatif* or *klitik*. In figure 2, we can see a lot of suffixes.

5. Architecture

The architecture of the Stemmer for '*basa Sunda*' can be seen in the following figure 3. Basically it

works by removing suffixes and then finds and compares it to the word in the dictionary.

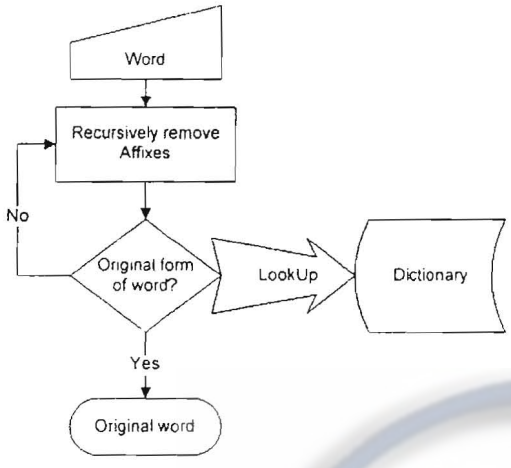

FIGURE 3. THE STEMMER OF BASA SUNDA ARCHITECTURE.

More detail about the architecture, can be seen below:

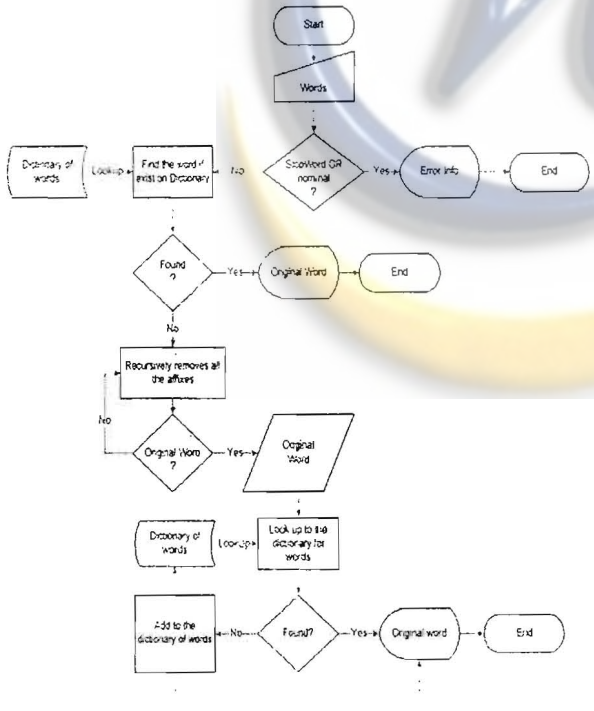

FIGURE 4. DETAIL OF STEMMER OF BASA SUNDA ARCHITECTURE.

6. Testing

In order to check whether we have develop the stemmer correctly, we conduct a testing scenario. First, we compile some document in basa Sunda from [7], [8], 17 articles and unique word about 6034 words.

Then we applied this source to our stemmer. And the result was stored in spreadsheet file, so we can assess and verified the result. As a native speaker of *basa Sunda*, we sure can assess and verified the result ourselves. By the end of testing we achieve 227 words of word with affixes that is never processed.

7. Accuracy

To our surprise, we found that the stemmer we build can achieve estimated accuracy for 94, 27 %. The lack of numbers in our dictionary caused us 227 words out of 6034 word never processed due to affixes word, people name, area name and words from the language other than *basa Sunda*.

8. Conclusions

After series of testing, we come to conclude that our "Stemmer for basa Sunda" was able to removing the 'kecap rajegan' with the accuracy estimations 94, 27%. Estimations of accuracy can be improved to 95, 15% if our dictionary is also improved.

For future work we suggest to add more '*bakal kecap*' into dictionary, fixing detection '*kata rejegan*' algorithm by adding the fuzzy similarity comparison after stemmer work. We also suggest converting structure function recursively so the failure due to the sequence precedence process can be decreased.

References

[1] Adriani, et.al. 2007. "Stemming Indonesian: A Confix-Stripping Approach", ACM Transactions on Asian Language Information Processing, Vol.6, No.4, Article13.

[2] Yayat, Sudaryat, et.al, 2009, "Tata Bahasa Sunda Kiwari", Yrama Widya, Bandung, Indonesia.

[3] Jelita, Asian, et.al, 2005, "Stemming Indonesian", Australian Computer Society, Inc., Conferences in Research and Practice in Information Technology, Vol. 38. V. Estivill-Castro, Ed.

[4] Lovins, Julie, Beth, 1968, "Development of a Stemming Algorithm", Mechanical Translation

and Computational Linguistics, vol.11, nos.1 and 2, March and June.

[5] Porter, M., F., 1980, "An Algorithm for Suffix Stripping", Computer Labotatory, Corn Exchange Street, Cambridge.

[6] Strzalkowski, Tomek, 1993, "Robust Text Processing in Automated Information Retrieval", Proc. of ACL-Sponsored workshop, on Very Large Corpora, Ohio State Univ, Colombus.

[7] ____, Majalah Sunda Mangle online, http://majalah-mangle.com, access November, 2009.

[8] ____, sunda.blogspot.com, access November, 2009.