Using Pattern Matching for Root Word Extraction in Bahasa Indonesia

Alvin Natawiguna - 13512030[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13512030@std.stei.itb.ac.id

***Abstract*—Bahasa Indonesia is one of the languages that uses certain affixes to modify the meaning of root words. In this paper, I describe how pattern matching methods can be used to extract the root word from the compound word.**

***Index Terms*—Bahasa Indonesia, pattern matching.**

I. Introduction

Human language commonly uses affixes to apply modifications in the meaning of its words. These kind of modifications allow the speakers to convey certain meaning(s) in the sentence. In general, affixes can be used to derivate a certain word, i.e. turning verbs into nouns, or to inflect word to a certain grammatical category or parts of speech. These include [2]

1. tense, which describes the time of event or state denoted by the verb in relation to some other temporal reference point,
2. mood (or modality), the manner of how the speaker conveys his/her thought,
3. voice, the expression of semantic functions attributed to the referents of the clause; i.e., whether a subject is an actor, a patient, or a recipient,
4. aspect, which describes the flow of time associated with the word,
5. person, the distinction of the subjects involved in the event (first/second/third person, archaic/not)
6. number, the distinction of counts (one/more than one)
7. gender, which describes the masculinity/femininity of the subjects, and
8. case, the context of which the language is used.

These are sketches of the precise definitions. The real definition vary from language to language; not every language apply these things. For example, in Bahasa Indonesia that follows *Ejaan Yang Disempurnakan*[1] spelling system, there is no gender category when describing a subject, i.e. 'dia' in Bahasa Indonesia is used to replace the subject 'he' (masculine), 'she' (feminine), and 'it' (neuter/inanimate) in English.

This poses a challenge when we are trying to algorithmically disambiguate and classify these words. Language used typically in human speech are always context-bound. In order to get the accurate meaning, i.e. the context, it is important to extract and classify the words used to the appropriate grammatical category. I explain how current pattern matching algorithm can be applied to extract root words and classify them in the appropriate categories.

This paper focuses on the discussion of verb extractions.

II. Terms and Definitions

A. Pattern Matching and Boyer-Moore Algorithm

Given a *text*, a string with n characters, and a *pattern*, a string with the length of m characters (m <= n), pattern matching is the process of finding the location of the occurrence of aforementioned pattern in the text. [1] Pattern matching is heavily used in the field of Natural Language Processing, since it involves finding a certain pattern in a string.

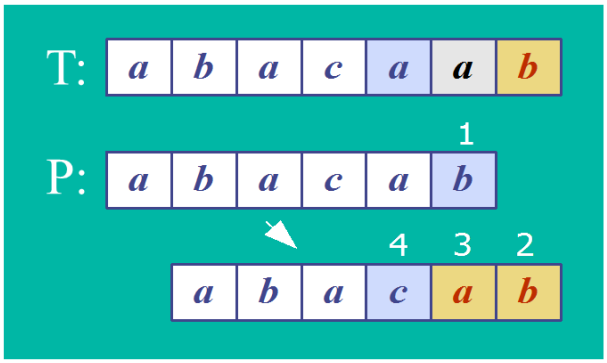

Figure 1. Example of pattern matching, using Boyer-Moore algorithm. Source: [1]

One of the most commonly used pattern matching algorithm in the Computer Science course is the Boyer-Moore Algorithm. The Boyer-Moore algorithm uses two main techniques for matching the pattern. First, it uses the looking-glass technique, which matches the pattern in the text by matching the last letter in the pattern first. Next, it

[1] Ejaan Yang Disempurnakan (EYD) is the spelling system introduced in 1972 that is meant to replace the old Soewadi Spelling System (SSS).

utilizes the character jump technique, which 'moves' the pattern across the text if a mismatch is detected. There are three conditions to this:

1. If the mismatched character is in the pattern, the pattern is right-shifted until the last occurrence of the matching character is aligned with the text.
2. If the mismatched character is in the pattern but a right shift to the character is not possible, then the pattern is shifted to the right by one character.
3. If the mismatched character is not in the pattern, the pattern is shifted until it skips the mismatched character.

In order to achieve this, Boyer-Moore algorithm utilizes the last occurrence function, which is defined in the following pseudocode:

```
function lastOccurrence(pattern:
string, c: character) → integer
{ returns the index of the last
occurrence of c in pattern, or -1 if
not found.
  Character index is from 0 to n-1,
with n as the length of string. }
LOCAL DICTIONARY
i, ret : integer

ALGORITHM
ret ← -1
i ← last index of character in
pattern
while ret = -1 and i =/= ret do
      if p[i] = c then
            ret ← i
      else
            i ← i - 1
      endif
endwhile
→ ret
```

Figure 2. Pseudocode of the last occurrence function.

Performance-wise, Boyer-Moore algorithm is capable of running at most O(n+m) (worst case) if the pattern does not appear at the text. On the other hand, if the pattern does exist in the text, the Boyer-Moore algorithm is capable of running at most O(nm) (worst case).

Memory-wise, depending on the implementation, Boyer-Moore algorithm may require zero to O(m) memory capacity. This comes from storing the unique characters that exists in the pattern, including its last occurrence index. During the pattern-matching process, this will reduce the last occurrence lookup process complexity down to O(1) (if using hashing) or O(|u|) (if using an array), with u is the set of unique characters in the pattern.

B. Spell-Checking

(to be written)

III. DISCUSSION

A. Affix Ambiguity

Based on the grammatical category, affixes modify the root word based on the context. This may also involve the As mentioned before, an affix may be used to modify the meaning of the word.

For example, consider the sentence, "Saya meniduri adik." This sentence describes that the first subject, 'saya' (I) doing the action of the root word (tidur – sleep) to the second subject (adik). Connotatively, this means 'saya' is committing fornication to his/her own brother/sister (adik). 'Saya' is the dominant subject on this sentence. On the other hand, consider a similar sentence, "Saya menidurkan adik." This sentence describes that the first subject, 'saya', is trying to make the second subject, 'adik', do the action 'tidur'. This may involve cooperation or no cooperation of both subject described; 'saya' may be trying hard to make 'adik' to 'tidur', but the 'adik' does not want to.

The suffix '-kan', however, may not always convey the same modification. For example, when used in words like 'memakan', i.e. 'saya memakan daging sapi', the first subject, 'saya', becomes the dominant one. This dominance conveys that the second subject, 'daging sapi' (beef), becomes the victim of 'saya' in the action, 'makan'.

Note that the addition of 'me-' suffix in the first example replaces the first letter of the root word ('tidur' becomes 'meniduri'). This is a common occurrence in Bahasa Indonesia, especially with consonants 't' ('timpa', 'tambah', 'tulah'), 'k' ('kantuk', 'kafan'), and 'p' ('putus', 'putar', 'pulang'). In addition, the 'me-' suffix is sometimes transformed to follow the phonetics of the first letter, such as 'men-' in 'meniduri' ('tidur'), 'meng-' in 'memaku' ('paku') and 'mengantuk' ('kantuk'), and 'mem-' in 'membantu' ('bantu').

B. Extraction Strategy

In general, the extraction strategy is as follows:

1. Create a dictionary of root words of Bahasa Indonesia.
2. Create a dictionary for prefixes and a dictionary of suffixes, including its variations. It is preferable to store the dictionary as a multi-level index, with the first level index containing base prefix/suffix and the second level containing the variations of the prefix/suffix. This is similar to a tree structure. For example:

Level	Keyword				
1	me-	-	-	ke-	…
2	men-	mem-	men-	-	…
3	meng-	-	-	-	…

Table 1. Example of prefix/suffix index structure.

3. When a word is entered, the first one/two characters are assigned as a prefix pattern and compared to the matching defined prefix. If the prefix is found, the next character is concatenated to the current pattern, and then matched with the second level index. The process is repeated until there are no more matches in the tree, or the tree node is a leaf.
4. Remove the matching prefix from the entered word.
5. Repeat step 3 and 4 inversely, by matching the last character(s) with the suffix tree.
6. Using the Boyer Moore algorithm, use the trimmed word as the pattern to match with the words in the root word dictionary.
7. End.

IV. Experiment

(to be written)

V. Conclusion

(to be written)

VI. Appendix

(to be written)

VII. Acknowledgment

(to be written)

References

[1] LinguaLinks Library, "What is Grammatical Category?," SIL International, 5 January 2004. [Online]. Available: http://www-01.sil.org/linguistics/GlossaryOflinguisticTerms/WhatIsAGrammaticalCategory.htm. [Accessed 19 May 2014].

[2] R. M. Andrew Davison, "Pattern Matching," 5 May 2014. [Online]. Available: http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2013-2014-genap/Pencocokan%20String%20(2014).ppt. [Accessed 19 May 2014].

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Mei 2014

ttd

Alvin Natawiguna