ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 6, Issue 7, July 2017

Calculation Algorithm Complexity of Porter's Algorithm in Information Retrieval

Lasmedi Afuan, Retantyo Wardoyo

***Abstract*— Complexity is part of a computer science related to the use of algorithms in a particular application. The complexity of the algorithm is divided into two, namely the complexity of time and space complexity. Measured time complexity of computing the number of stages required to run the algorithm as a function of a number of data n (size of the input). This study calculating complexity on one stemming algorithms commonly used in Information Retrieval that is Porter's Algorithm. The complexity of the calculations done using ruang.Tahapan complexity approach taken is to translate the flowchart of stages Porter to pseudocode, then calculate the time complexity. The final conclusion is that the time complexity of the Porter's algorithm is O (7).**

***Index Terms*—Complexity, porter's algorithm, information retrieval.**

I. INTRODUCTION

Complexity is part of a computer science related to the use of algorithms in certain applications, one of which is a calculation algorithm on a corpus stemming from information retrieval. There are various algorithms that can be used to make the process of stemming algorithms including Porter, Nazief and Adriani, Tala and other algorithms.

In this research, calculation the complexity of the porter's algorithm . Calculations conducted complexity is the complexity of the time, this is done to see how efficiently the porter's algorithm if used in the process of stemming.

II. FUNDAMENTAL

2.1 Algoritma Complexity

A good algorithm is an algorithm is not necessarily true, but it must also be efficient. The efficiency of an algorithm is measured from the time (time) execution of the algorithm and space requirements space (memory). Efficient algorithms are algorithms that minimize the need for time and space. The time and space an algorithm depends on the size of the input (n), which states the amount of data processed. The efficiency of the algorithm can be used to assess the good algorithms of various algorithms for problem solving. Quantity that is used to explain an abstract model of time or space measurement is the complexity of algorithms.

There are two kinds of algorithmic complexity, the complexity of time and space complexity [1]. Complexity time, T (n) and space complexity S (n), the large size of data input to an algorithm, n. For example, in the sorting algorithms elements of the array, n is

the number of array elements, whereas in n matrix multiplication algorithm is a matrix of size n x n. First on the complexity of the room, S (n), measured from memory used by the structure of the data contained in the algorithm as a function of the input size n. And the second on the complexity of the time, T (n), is calculated based on the number of stages of computing operations performed in an algorithm.

In practice, the operation of which is calculated only typical operations that underlie an algorithm. For example, a typical operation in search algorithms in the array is the array element comparison. Typical operation is the sorting algorithm comparison and exchange elements. The complexity of the algorithm can also predict for the future decision-making and measuring of automatic systems to a high level. Usually expressed asymptotic complexity of the algorithm with big-O notation. If the complexity of time to execute an algorithm expressed by T (n), and meet.

$$T(n) \leq C(f(n))$$

For $n \geq n0$, the complexity can be expressed by

$$T(n) = O(f(n)).$$

There are 2 types of use Big O notation, namely: 1. Infinite asymptotics 2. infinitesimal asymptotics second difference is the type of use of this notation only on the application. For example, in the infinite asymptotics with equation

$$T(n) = 2n^2 - 2n + 2$$

For large n, the growth of T (n) will be proportional to n^2 and ignoring the parts that do not dominate us, then we write

$$T(n) = O(n^2)$$

On infinitesimal asymptotics, Big O is used to explain the error in the approximation to a mathematical function, for example

$$e^x = 1 + \frac{x}{1} + \frac{x^2}{2} + O(x^3), \qquad x \to 0$$

The mistake has a difference

$$e^x - (1 + \frac{x}{1} + \frac{x^2}{2})$$

2.2 Stemming

Stemming is a process that maps the variant form words into basic word or root word [1]. There are two approaches to the process of stemming ie turan dictionary approach and the approach [3].

Several studies discussing the algorithms stemming using the approach rules, among others [4] uses an Porter's Algorithm for stemming a text document in Indonesian language, Nazief and Adriani Algorithm [5] for stemming check the title and abstract Thesis, besides [3] using Tala's algorithms adapted from Porter's algorithm.

2.3 Porter's Algorithm

Porter's common steps of the algorithm can be seen in fig 1.

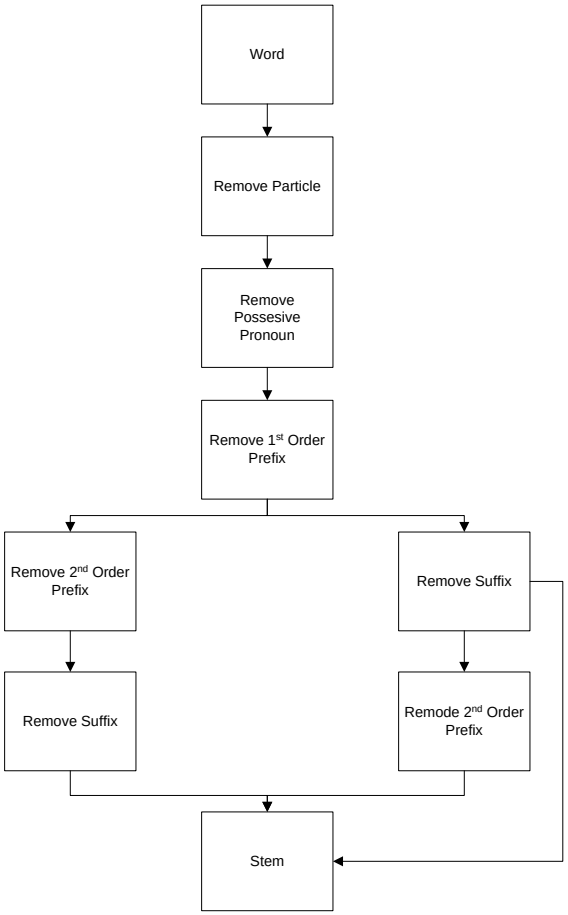

Fig 1. The steps of Porter's Algorithm

Based on fig 1, the steps of the Porter 's algorithm is as follows

1. Remove Particle.
2. Remove possesive Pronoun.
3. Remove the prefix first. If no go to step 4a, if there are looking for then go to step 4b
4. Step 4
   a. Remove the second prefix, proceed to step 5a.
   b. Remove the suffix, if it is not found then the word is assumed to be the root word. If found then proceed to step 5b.
5. Step 5
   a. Remove the suffix. Then the final word is assumed to be the root word.
   b. Remove the second prefix. Then the final word is assumed to be the root word.

III. Porter's Algorithm Complexity

Based on the steps at the Porter's algorithm in fig 1, do the translation process steps into pseudocode to calculate the time complexity of the algorithm. The following pseudocode of the algorithm Porter can be seen on table 1.

Table 1. Porter's Algorithm

```
Porter's Algorithm
Dictionary
        word, rootword : string
Algorithm
        word ← word
          word ← removeparticle(word)
            word ← removeprefix1(word)
              word ← removeprefix2(word)
                word ← removeprefix2(word)
                  word ← removesuffix(word)
                      rootword ← word
```

Table 2. Steps of Porter's Algorithm

line	Steps
1	word ← word
2	word ← removeparticle(word)
3	word ← removeprefix1(word)
4	word ← removeprefix2(word)
5	word ← removeprefix2(word)
6	word ← removesuffix(word)
7	rootword ← word

Calculation time complexity of Porter;s Algorithm
line 1 → O(1)
line 2 → O(1)
line 3 → O(1)
line 4 → O(1)
line 5 → O(1)
line 6 → O(1)
line 7 → O(1)

So, time complexity of Porter's Algorithm is O (7)
Example text for *Stemming* process.

Proses analyzing adalah proses analisa dari hasil proses tagging sehingga diketahui seberapa jauh tingkat keterhubungan antar kata- kata dan antar dokumen yang ada

Result of stemming using Porter's algorithm

Proses analyzing proses analisa hasil proses tagging tahu berapa jauh tingkat terhubung antar kata kata antar dokumen ada

IV. Conclusion

Based on research that has been done, the calculation time complexity of the algorithm generated Porter is O (7)

.

Acknowledgment

I would like thank you to all colleagues in computer science postgraduate program, faculty of mathematics and natural sciences (FMIPA), Universitas Gadjah Mada (UGM), for supporting this research.

REFERENCES

[1] S. N. B. Tjaru, "Kompleksitas Algoritma Pengurutan Selection Sort dan Insertion Sort," *Makal. IF2091 Strateg. Algoritm.*, no. 13508054, 2009.

[2] F. Z. Tala, "A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia," *M.Sc. Thesis*, vol. pp, pp. 39–46, 2003.

[3] M. S. Utomo, "Implementasi Stemmer Tala pada Aplikasi Berbasis Web," *J. Teknol. Inf. Din.*, vol. 18, no. 1, pp. 41–45, 2013.

[4] L. Afuan, "Stemming Dokumen Teks Bahasa Indonesia menggunakan Algoritma Porter," *J. Telemat.*, vol. 6, no. 2, pp. 34–40, 2013.

[5] H. R. Pramudita, "Penerapan Algoritma Stemming Nazief * Adriani dan Similarity pada Penerimaan Judul Thesis," *J. Ilm. DASI*, vol. 15, no. 04, pp. 15–19, 2012.