APLIKASI KOREKSI KESALAHAN PENULISANKATA DALAMBAHASA INGGRIS DENGAN MENGGUNAKAN ALGORITMA RABIN- KARP

[1]*Sendy Agustian,* [2]*Kenny,* [3]*Kristien Margi Suryaningrum*
[1,2,3]*Fakultas Teknologi dan Desain Universitas Bunda Mulia*
[1,2,3] *Jl. Lodan Raya No. 2 Ancol, Jakarta Utara 14430*
[1,2,3]*{agustianosendy, kenny220699, kristienmargi}@gmail.com,*

Abstrak

Mengetik merupakan suatu kebiasaan untuk melakukan suatu kepentingan berupa tugas ataupun keperluan lainnya. Sementara itu, pastinya akan ada saat dimana ada sebuah kesalahan dalam pengetikan (typo). Jika ada suatu typo terkadang akan membuat kata-kata itu menjadi ambigu, tidak benar ataupun menjadi acak. Untuk penulisan karya ilmiah atau skripsi, kesalahan pada penulisan kata mungkin sudah menjadi hal yang biasa, tetapi untuk melakukan koreksi secara manual akan memakan waktu yang lama. Maka untuk menyelesaikan masalah yang terjadi salah satunya dapat dibantu oleh bantuan program, dengan membuat suatu aplikasi yang dapat digunakan oleh semua orang sebagai alat bantu untuk memeriksa kesalahan dalam pengetikan dengan menggunakan algoritma Rabin-Karp. Cara kerja algoritma Rabin-Karp adalah dengan melakukan pencocokan string berdasarkan masing-masing nilai hash pada teks dan pattern. Pattern didapat dengan melakukan pencarian kata yang tersimpan di dalam database berdasarkan kata terdekat dari typo yang ditemukan. Berdasarkan hasil perhitungan yang telah dilakukan secara manual dan program, diperoleh hasil bahwa kata yang merupakan kata singkatan akan menghasilkan hasil dan nilai yang berbeda daripada kata yang bukan merupakan kata singkatan. Hasil untuk kata kedua dari kata singkatan akan menghasilkan nilai yang berbeda dikarenakan kata tersebut menghasilkan kata yang berbeda atau tidak sama dengan yang seharusnya.

***Kata Kunci**: Algortima Rabin-Karp, Hash, koreksi, String Matching*

Abstract

Typing is a habit to do something important such as homework or other tasks. Meanwhile, there will be definitely an error occurred while in typing (typo). If there is a typo it will make the words become ambiguous, incorrect or random. For scientific papers or thesis, errors in writing maybe have become common, but to do correction manually will take a long time .To overcome the problem one of the way can be assisted by program, by making an application that can be used by everyone as a tool to check errors in typing using Rabin-Karp algorithm. The way the Rabin-Karp algorithm works is by matching strings based on each hash value in the text and pattern. Pattern can be found by searcing word listed in database based on the closest word from the typo that was found. Based on the test results of calculation that have been done by manually and program, the results are obtained that the word which is an abbreviation word will make different results from words that are not an abbreviation word. The results for the second word of the abbreviation will make a different value because the word produce a different word or not the same as it should.

***Keywords**:Correction, Hash, Rabin-Karp Algorithm, String Matching*

PENDAHULUAN

Mengetik merupakan suatu kebiasaan untuk melakukan suatu kepentingan berupa tugas seperti laporan ilmiah atau skripsi ataupun keperluan lainnya seperti *chatting* atau berkomentar dalam media sosial. Terutama bagi kaum muda sudah sewajarnya jika setiap hari kita melakukan pengetikan baik itu melalui *smartphone* atau komputer.

Sementara itu, pastinya akan ada saat dimana ada sebuah kesalahan dalam pengetikan (*typo)*. Jika ada suatu *typo* terkadang akan membuat kata-kata itu menjadi ambigu, tidak benar ataupun menjadi acak.Kesalahan pada penulisan kata mungkin sudah menjadi hal yang biasa untuk penulisan karya ilmiah atau skripsi, tetapi untuk melakukan koreksi secara *manual*akan memakan waktu yang lama.

Aplikasi koreksi kesalahan kata dapat membantu untuk menemukan letak kesalahan dan memberi sugesti dengan menggunakan algoritma *Rabin-Karp.* Algoritma ini merupakan salah satu algoritma *string matching* dengan melakukan pencocokan *string* berdasarkan masing-masing nilai *hash* pada teks dan *pattern*.

Penelitian mengenai aplikasi koreksi kesalahan berbasis pada tulisan berbahasa Indonesia untuk meningkatkan kualitas penulisan karya ilmiah menggunakan metode *N-Gram*. Aplikasi koreksi dari penelitian ini selain melakukan pengecekan kata yang sesuai dengan EyD juga dapat melakukan pengecekan penggunaan tanda baca yang tidak sesuai dalam sebuah kalimat yang ada pada dokumen yang diuji. Dari proses pengecekan penggunaan tanda baca, sistem dapat melakukan koreksi secara otomatis terhadap kalimat-kalimat yang tidak sesuai tanda bacanya. Hasil dari penelitian tersebut adalah sebuah aplikasi koreksi yang dapat mendeteksi kesalahan-kesalahan yang terjadi pada dokumen-dokumen Bahasa Indonesia dan dapat melakukan perbaikan secara otomatis terhadap kata dan kalimat yang tidak sesuai dengan EyD [1].

Penelitian selanjutnya menggunakan metode *Levenshtein Distance* dan *N-Gram* untuk identifikasi kesalahan penulisan kata (*Typhographical Error)* pada dokumen berbahasa Indonesia. Berdasarkan hasil pengujian dan analisis pada identifikasi *typographical error* pada dokumen bahasa Indonesia menggunakan metode *N-gram* dan *Levenshtein Distance* dapat diambil kesimpulan bahwa metode pendekatan *Dictionary Lookup* pada proses identifikasi *typographical error* pada dokumen bahasa Indonesia dapat diterapkan dengan baik untuk mencari kata *typographical error* dalam dokumen data input. Untuk menentukan kandidat kata, metode *Levenshtein Distance* dapat menghasilkan kandidat kata yang sesuai dengan nilai aktual yang diharapkan *user*. Namun untuk kata *typographical* error tertentu, jumlah kandidat kata yang ditampilkan dalam sistem terlalu banyak. Hasil presisi dan *recall* pada penelitian ini memiliki

nilai yang beragam pada setiap skenario pengujian. Nilai presisi terbaik yang dihasilkan sistem sebesar 0.97 pada skenario pengujian *typographicalerror* jenis *insertion.* Sementara itu, nilai *recall* terbaik yang dihasilkan sistem sebesar 1 pada skenario pengujian *typographicalerror* jenis *substitution* [2]. Penelitian lain mengenai implementasi algoritma *Rabin-Karp* pada pendeteksian pengulangan kata kerja dalam bahasa Inggris. Hasil dari penelitian tersebut yaitu suatu aplikasi yang dirancang telah mampu memberikan solusi dengan meng-implementasikan algoritma *Rabin Karp StringMatching* untuk melakukan pencarian *string* dan dapat menambahkan kosa kata yang baru pada aplikasi[3].

Algoritma *Rabin-Karp* merupakan salah satu algoritma *String Matching* dengan membandingkan nilai *hash*. Terdapat rumus untuk mencari nilai *hash* yang ditunjukkan pada persamaan (1) [4].

$$H = (c_1 * b^{k-1} + c_2 * b^{k-2} + c_3 * b^{k-3} + \ldots + c_n * b^{k}) \quad (1)$$

H = nilai *hash*
c = nilai ascii karakter
k = banyak karakter
b = basis (bilangan prima)

Text mining merupakan proses pencarian pola atau penggalian informasi dari data teks untuk menghasilkan informasi baru. Tujuan dari *text mining* adalah menemukan informasi yang penting dari teks dengan meng-ubah teks menjadi data yang dapat digunakan untuk analisis yang lebih lanjut.

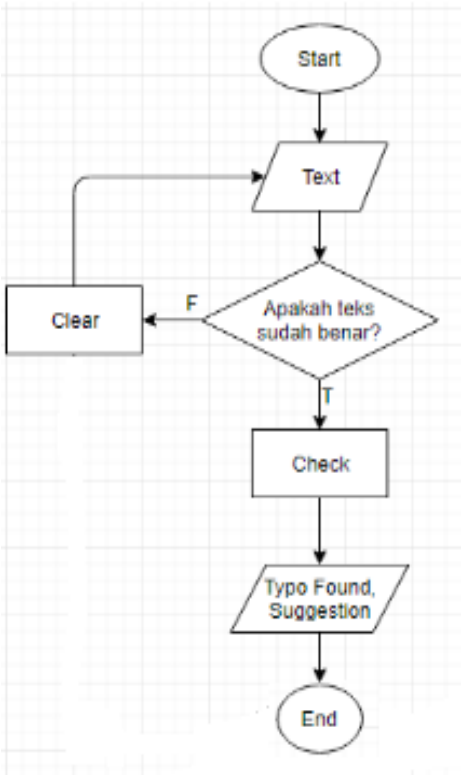
Tahapan awal yang dilakukan pada *text mining* yaitu *preprocessing* diantaranya *case folding*, *tokenizing*, *filtering*, dan *stemming* [5]

METODE PENELITIAN

Perancangan Proses

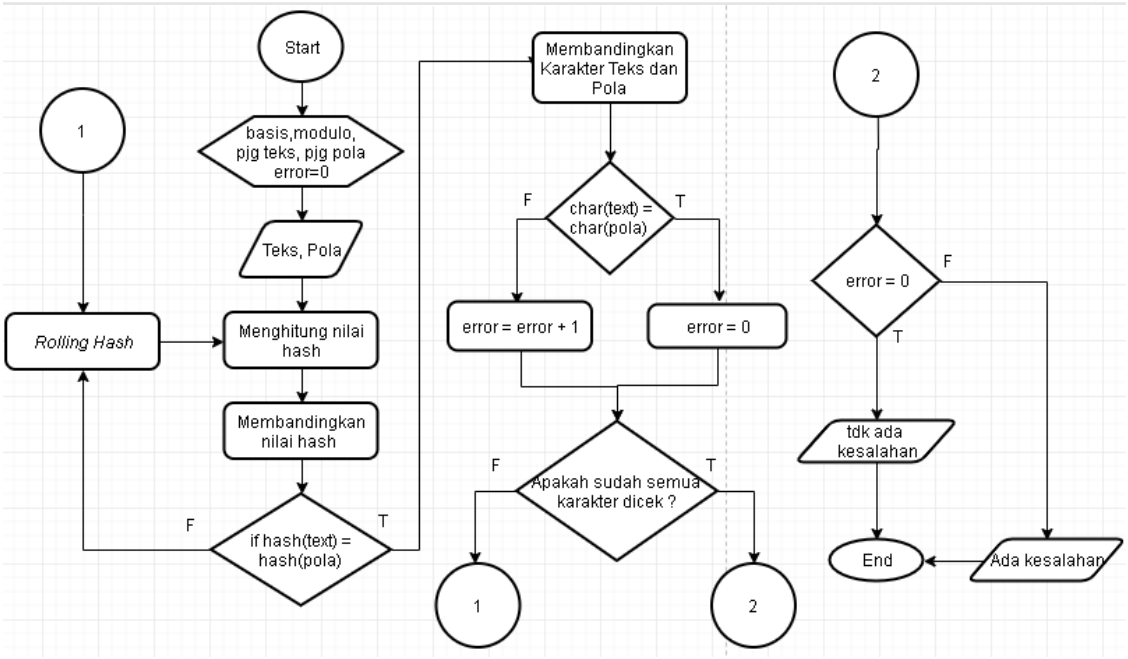
Cara kerja sistem yang ditunjukkan pada Gambar 1 adalah *user* menjalankan aplikasi koreksi kesalahan kata, *user* memasukkan teks yang ingin diperiksa, jika *user* salah memasukkan teks, maka tekan *clear* untuk menghapus teks yang sudah dimasukkan, tetapi jika teks yang dimasukkan *user* sudah benar, maka tekan *check* untuk memeriksa apakah ada kesalahan pada pengetikkan yang dilakukan *user*.

Hasil yang dikeluarkan oleh sistem jika menemukan *typo* adalah memberikan *output* berupa *typo found* dan *suggestion*. Jika *user* ingin melakukan pemeriksaan ulang atau pemeriksaan terhadap teks lain, maka *user* harus menekan tombol *clear*, tetapi jika *user* tidak ingin melakukan pemeriksaan ulang atau pemeriksaan terhadap teks lain,maka *user* dapat menekan tombol *exit* untuk keluar dari aplikasi.

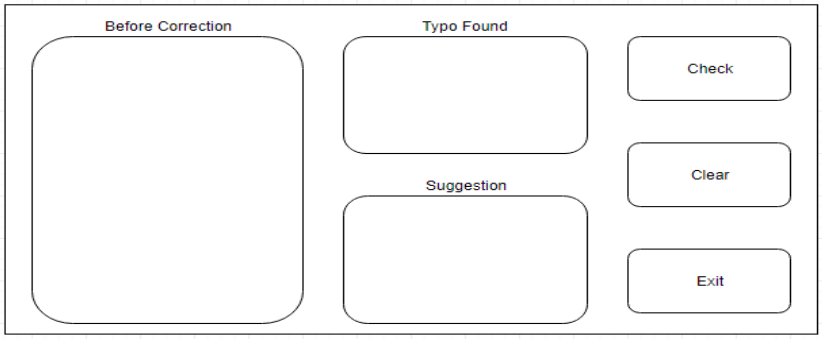


Gambar 1. *Flowchart* Aplikasi

Cara kerja dari algoritma Rabin-Karp di tunjukkan pada Gambar 2. yaitu: (1) Inisialisasi basis, modulo, pjg teks, pjg pola, error. (2) Melakukan inputan teks dan juga pola, pola didapatkan dengan pencarian kata dari database yang mirip dengan inputan teks (3) Teks akan dipotong menjadi beberapa substring berdasarkan panjang maksimal dari pola dan akan di cari nilai hash dari teks dan pola dan akan dilakukan perbandingan. (4) Jika sudah didapatkan nilai hash yang sama maka akan dilakukan perbandingan lagi terhadap substring teks dan pola. Jika karakter tidak sama maka error akan bertambah satu, jika sama maka error sama dengan 0. (5) Lalu akan dilakukan dengan pengecekan semua karakter, jika false maka akan menuju on-page connector nomor satu, dan akan melakukan rolling hash. Jika true maka akan menuju on-page connector nomor dua. (6) Setelah itu akan dilakukan pengecekan terhadap error jika tidak sama dengan 0 maka akan dicetak ada kesalahan, jika sama dengan 0 maka akan dicetak tidak ada kesalahan



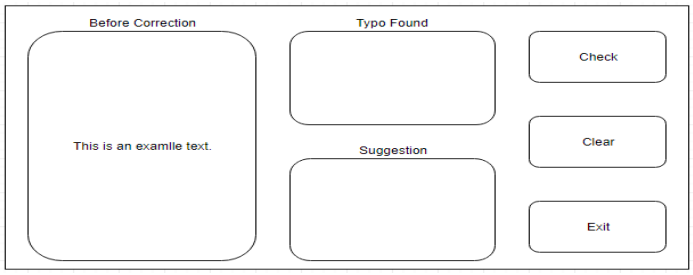
Gambar 2. *Flowchart* Algoritma *Rabin-Karp*



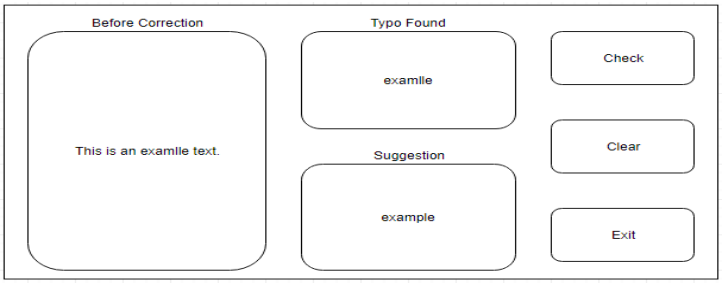
Gambar 3. Perancangan Tampilan Koreksi Kata

Perancangan tampilan

Berikut merupakan penjelasan yang di tunjukkan pada Gambar 3 (1) *Before Correction:* merupakan tempat dimana *user* menuliskan teks (2) *Typo Found*: merupakan *output* jika terdapat *typo* pada teks yang dituliskan oleh *user* (3) *Suggestion* merupakan *output* yang diberikan sistem untuk memberikan saran untuk mengganti kata yang salah. (4) *Check* merupakan tombol untuk *user* memeriksa teks yang dituliskannya. (5) *Clear* merupakan tombol untuk menghapus teks yang dituliskan *user* jika salah memasukkan teks, dan *Exit* merupakan tombol untuk keluar dari aplikasi jika *user* sudah tidak mau menggunakan aplikasi. Perancangan tampilan input di tunjukkan pada Gambar 4. *User* dapat memasukkan sebuah teks yang nanti nya akan di cek pada bagian input teks *Before Correction.*



Gambar 4. Perancangan Tampilan *Input*



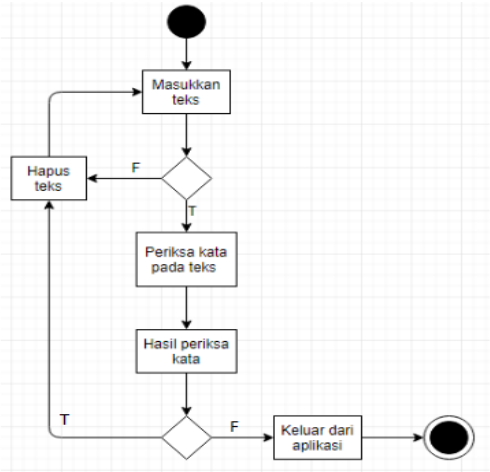
Gambar 5. Perancangan Tampilan *Output*

Perancangan tampilan output di tunjukkan pada Gambar 5 merupakan hasil setelah tombol *Check* ditekan, Sistem akan menampilkan letak kesalahan kata pada bagian *Typo Found*, dan akan menampilkan saran pada bagian *Suggestion*.

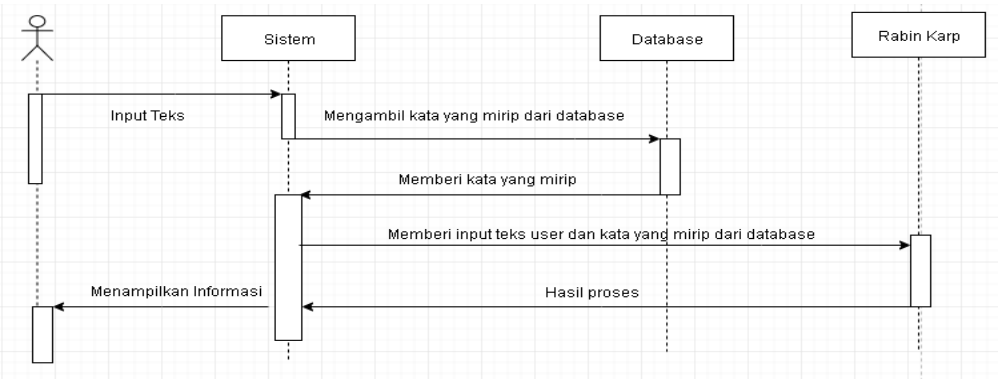
Perancangan Sistem

Activity diagram tunjukkan pada Gambar 6.*User* memasukkan teks, *user* dapat menghapus teks jika salah, jika teks sudah benar, sistem akan memeriksa kata yang terdapat pada teks. Sistem akan memberikan hasil periksa kata, jika terdapat kesalahan maka sistem akan memberitahukan letak kesalahan dan saran untuk menggantikan kata yang salah tersebut. Jika *user* ingin memeriksa teks yang lain, maka *user* harus menghapus teks untuk memasukkan teks yang lain, tetapi jika *user* sudah tidak mau memeriksa teks yang lain, maka *user* dapat keluar dari aplikasi.

Sequence diagram di tunjukkan pada Gambar 7. *User* memasukkan teks, kemudian algoritma *Rabin-Karp* melakukan pengambilan kata pada *database* yang ada pada sistem. Sistem melakukan pencocokan kata dengan algoritma, kemudian algoritma memberikan kata kerja yang sama pada sistem. Sistem menampilkan letak kesalahan kata sekaligus saran kata pengganti kepada *user*, kemudian *user* dapat memilih kata pengganti sesuai dengan yang disarankan sistem.



Gambar 6. *Activity* diagram



Gambar 7. *Sequence* diagram

Algoritma

Terdapat fungsi algoritma *Rabin-Karp* di tunjukkan pada Gambar 8. Algoritma *Rabin-Karp* digunakan untuk mencocokan nilai *hash* dari teks dan pola. Cara kerja dari algoritma *Rabin-Karp* yaitu mencari nilai *hash* suatu pola terlebih dahulu setelah itu

mencari *N-Gram* dengan cara mengurangi panjang karakter dari teks dan pola. Lalu mencocokan nilai *hash*, jika nilai sama maka akan di bandingkan lagi terhadap karakter pola dan teks tersebut, jika tidak sama maka akan mengulangi *looping* sampai selesai, jika karakter sama maka proses akan memberi tahu letak posisi indeks yang sudah di-temukan. Fungsi *hash* dapat ditunjukkan pada Gambar 9.

Fungsi *hash* digunakan untuk mencari nilai *hash* dari suatu kata, dari mencari panjang maksimal suatu *array* dan maksimal kata yang terdapat pada *array* tersebut dan dilakukan perhitungan terhadap masing-masing karakter dengan mencari nilai ASCII dan diproses dengan rumus *hash*.

```
Function RabinKarp(input pat: string[1..m], txt: string[1..n])
h_pat <- FungsiHash(pat[1..m])
for i <- 0 to n-m do
        h_txt <- FungsiHash(txt[i..i+m-1])
        if h_pat == h_txt then
                for j <- 0 to m do
                        if txt[0].charAt(i+j) != pat[0].charAt(j) then
                        break
                        Endif
                Endfor
                if j == m then
                write("Pattern found at index")
                write(i)
                Endif
        Endif
Endfor
```

Gambar 8. Algoritma *Rabin-Karp*

```
Function FungsiHash(input txt: string[1..m])
for i <- 0 to maxarray do
        for j <- 0 to maxchar do
                maxchar <- txt[i].length()
                nilai_ascii <- txt[0].charAt(j)
                h_temp < nilai_ascii*Math.pow(q,maxchar+(-j-1))+h_temp
        Endfor
Endfor
h_temp <- h_temp % modulo
return h_temp
```

Gambar 9. Fungsi Proses Nilai *Hash*

HASIL DAN PEMBAHASAN

Terdapat contoh kalimat yang dapat diuraikan sesuai dengan masing-masing kata yang terdapat pada kalimat tersebut untuk membandingkan perhitungan nilai *hash* secara *manual* dan program.

Contoh kalimat :

"Finally my exam is over, tomorrow we'll go to some places I've never been before, with my family.Yes, it's holiday. I'm going to sleep earlier tonight!" Setelah disusun ke dalam Tabel sesuai dengan masing-masing kata maka didapat hasil seperti pada Tabel 1.

Perhitungan Secara Mnual

Perhitungan nilai *hash* diawali dari indeks pertama sampai indeks terakhir. Dengan nilai basis 3 dan q 107. Berikut merupakan contoh perhitungan *hash* dariteks "*finally*". Setelah dilakukan perhitungan sampai indeks akhir secara *manual* maka didapat hasil yang ditunjukkan pada Tabel 2.

Perhitungan dengan Program

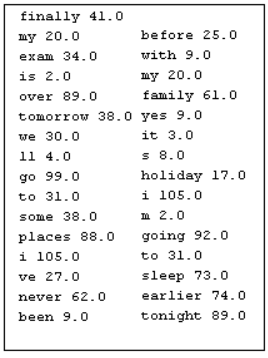
Setelah dilakukan perhitungan sampai indeks akhir dengan bantuan program maka didapat hasil seperti kata dan nilai *hash* disampingnya yang dapat di tunjukkan pada Gambar 10.

Tabel 1. Daftar Kata-kata

Kata - kata	
Finally	*Before*
My	*With*
Exam	*My*
Is	*With*
Over	*My*
Tomorrow	*Family*
We	*Yes*
Will	*It*
Go	*Is*
To	*Holiday*
Some	*I*
Places	*Am*
I	*Going*
Have	*To*
Never	*Sleep*
Been	*Earlier*
	Tonight

Tabel 2. Nilai *Hash* Kata

Kata	***Hash***	**Kata**	**hash**
Finally	41	*Before*	25
My	20	*With*	9
Exam	34	*My*	20
Is	2	*Family*	61
Over	89	*Yes*	9
Tomorrow	38	*It*	3
We	30	*Is*	2
Will	96	*Holiday*	17
Go	99	*I*	105
To	31	*Am*	79
Some	38	*Going*	92
Places	88	*To*	31
I	105	*Sleep*	73
Have	70	*Earlier*	74
Never	62	*Tonight*	89
Been	9		



Gambar 10. Perhitungan *Hash* dengan Program

Tabel 3. Analisa Hasil

Kata asal	Kata Hasil	*Manual*	Program
We'll	*We ll*	30 & 96	30 & 4
I've	*I ve*	105 & 70	105 & 27
It's	*It s*	3 & 2	3 & 8
I'm	*I m*	105 & 79	105 & 2

Analisis Hasil

Setelah menghitung nilai *hash* secara *manual* dan program terdapat beberapa hasil yang tidak sama yang dapat ditunjukkan pada Tabel 3.

hash(*finally*) = $((102 * 3^6) + (105 * 3^5) + (110 * 3^4) + (97 * 3^3) + (108 * 3^2) + (108 * 3^1) + + (121 * 3^0)) \ mod_{107}$

= (74,358) + (25,515) + (8,910) + (2,619) + (972) + (324) + (121)

= 112,819 mod_{107}

= 41

Terdapat beberapa nilai *hash* yang tidak sama berdasarkan analisa hasil Tabel 3 dapat di ketahui jika nilai *hash* dari kata asal dan kata hasil berbeda karena kata tersebut merupakan kata singkatan seperti "*we'll"* yang merupakan kata dari " *we will* " dan lain sebagainya.

KESIMPULAN DAN SARAN

Berdasarkan hasil perhitungan yang telah dilakukansecara *manual* dan program, diperoleh hasil bahwa kata yang merupakan kata singkatan akan menghasilkan hasil dan nilai yang berbeda daripada kata yang bukan merupakan kata singkatan. Hasil untuk kata kedua dari kata singkatan akan menghasilkan nilai yang berbeda dikarenakan kata tersebut menghasilkan kata yang berbeda atau tidak sama dengan yang seharusnya.

Adapun beberapa saran yang dapat disampaikan untuk dapat mengembangkan penelitian ini yaitu dapat dikembangkan dengan menerapkan *tenses* bahasa Inggris dan sistem aplikasi yang dirancang dapat memberikan *interface* yang lebih sederhana dan menarik.

DAFTAR PUSTAKA

[1] Andri, S. Ariana dan M. Andriani, "Aplikasi koreksi kesalahan berbasis pada tulisan berbahasa Indonesia untuk meningkatkan kualitas penulisan karya ilmiah", Dalam Prosiding Seminar Nasional Aplikasi Sains & Teknologi (SNAST), 2014, hal. A.169 – A.172.

[2] A. I. Fahma, I. Cholissodin, dan R. S. Perdana. "Identifikasi kesalahan penulisan kata (*Typographical Error*) pada dokumen berbahasa Indonesia menggunakan metode *N-gram* dan *Levenshtein Distance*", *Jurnal Pengembangan Teknologi Informasi*

dan Ilmu Komputer, vol. 2, no.1, hal. 53 – 62, 2017.

[3] H. P. F. A. Sinaga, "Implementasi algoritma Rabin KARP pada pendeteksian pengulangan kata kerja dalam penulisan bahasa Inggris", Skripsi Sarjana, Universitas Sumatera Utara, Medan, 2018.

[4] Herriyance, Handrizal, dan S. D. Faradilla, "Analisis algoritma Rabin-Karp pada kamus umum berbasis Android", *Jurnal Riset Sistem Informasi dan Teknik Informatika*, vol. 2, no.1, hal. 64 – 74, 2017.

[5] D. A. Putra dan H. Sujaini, "Implementasi algoritma Rabin-Karp untuk membantu pendeteksian plagiat pada karya ilmiah", *Jurnal Sistem dan Teknologi Informasi,* vol. 1, no. 1, hal. 1 – 9, 2015.