Aplikasi Editor Pemeriksa Ejaan dan Rekomendasi Kata dalam Bahasa Indonesia Berbasis Android

Febria Roosita Dwi[1], Rolly Intan[2], Leo Willyanto Santoso[3]
Petra Christian University
Jalan Siwalankerto 121-131 Surabaya
+62 31 8439040, 8394830-31
febriaroosita@gmail.com[1], rintan@petra.ac.id[2], leow@petra.ac.id[3]

ABSTRAK

Bahasa Indonesia saat ini sudah semakin populer dimata dunia Internasional, contohnya Vietnam menjadikan Bahasa Indonesia sebagai bahasa kedua. Salah satu cara membantu orang lain untuk mempelajari Bahasa Indonesia adalah dengan membuat alat yang memudahkan mereka untuk mengetahui apakah bahasa yang mereka pelajar itu benar atau tidak. Skripsi ini membahas aplikasi editor Bahasa Indonesia dengan fitur pemeriksa ejaan dan rekomendasi kata dalam Bahasa Indonesia berbasis Android. Android dipilih karena merupakan salah satu sistem operasi yang jumlah penggunanya terbanyak di dunia.

Adapun proses yang dilakukan adalah sebagai berikut: pengguna membuka suatu file atau melakukan pengetikan pada editor, input kata yang dimasukkan oleh user dilakukan pengecekan oleh Apache Lucene, dan Apache Lucene akan melakukan crawling terhadap data yang sudah pernah digunakan sebelumnya, dan apabila pengguna membuka suatu file, aplikasi ini akan melakukan pengecekan ejaan dengan menggunakan algoritma Nazief dan Adriani. Hasil dari proses ini adalah hasil input yang digarisbawahi apabila tidak sesuai dengan Bahasa Indonesia, atau hasil rekomendasi sesuai dengan hasil history yang tersimpan dan hasil crawling Apache Lucene. Input berupa kata atau file .doc. Output dari aplikasi ini berupa file .doc dan hasil rekomendasi atau hasil pengecekan ejaan. Aplikasi ini dibuat dengan bahasa pemrograman Java dengan Android Studio sebagai IDE-nya.

Hasil pengujian menunjukkan bahwa akurasi proses pengecekan Bahasa Indonesia dengan menggunakan algoritma Nazief dan Adriani masih memiliki beberapa kekurangan, panjang kata dan kerumitan kata juga sangat berpengaruh terhadap waktu dalam proses pengecekan tersebut, contohnya untuk melakukan pengecekan dokumen dengan 100 kata, waktu yang dibutuhkan bevariasi, mulai dari 1.48 menit hingga 3.3 menit. Selain itu, kata-kata yang terdiri dari dua suku kata, seperti pertanggungjawaban tidak bisa di cek oleh algoritma ini.

Kata Kunci: Bahasa Indonesia, Rekomendasi, Kata, Algoritma Nazief dan Adriani, Apache Lucene, Pemeriksa Ejaan.

ABSTRACT

Indonesian has now become more popular internationally, for example, Vietnam used Indonesian Bahasa as their second language. There are several ways to helping people to learn Indonesian Bahasa, one of them is by creating a tool that allows them to determine whether the language the are currently learning is true or not. This thesis discusses the application of Indonesian editor with spell checker and diction recommendation features in Indonesian

The process is carried out as follows: users opens a file or typing in the editor, words input will be checked with Apache Lucene, and Lucene will crawling to all of the data and give the recommendation base on the weight (Leveinsthein Algorithm) and if a user opens a file, the application will perform spell checking using Nazief and Adriani algoritm. The result of this process is the result of input that is underlined words that is not in accordance with Indonesian, or suitable recommendation result by Apache Lucene. Input could be words or .doc file. As Output will be written in .doc file and the spell checking results and words recommendation will be shown in editor as the user types. This application is built using Java programming language and Android Studio as the IDE.

The test result shows that the accuracy of the Nazief and Adriani Algorthms still need to be improved, word length and word complexity also affects the time in the checking process, for example: to check documents with 100 words, there are various of time from 1.48 minutes to 3.3 minutes. Besides that, this algoritm can not check the word which had 2 syllables, like "pertanggungjawaban" because of the stemming process.

***Keywords**: Indonesian Language, Recommendation, Word, Nazief and Adriani Algorithms, Apache Lucene, Spelling Checker.*

1. INTRODUCTION

Dewasa ini, Bahasa Indonesia merupakan salah satu bahasa yang dilirik oleh dunia Internasional. Indonesia yang memiliki jumlah penduduk yang terbesar ke 4 di dunia [9] dan menyumbang 40% dari jumlah penduduk di ASEAN, juga menjadi salah satu Negara yang dilirik oleh investor [11]. terlebih dalam menyambut Pasar Bebas ASEAN pada tahun 2016. Hal ini menyebabkan banyak Negara di dunia yang mulai menggunakan Bahasa Indonesia dalam sebagai bahasa kedua seperti Vietnam, dan di Australia Bahasa Indonesia menjadi Bahasa popular keempat.

Hal tersebut membuka peluang bagi Bangsa Indonesia untuk lebih dikenal oleh dunia dan Internet merupakan salah satu sarana bagi orang asing untuk mempelajari Bahasa Indonesia. Salah satu contoh dari sarana pembelajaran Bahasa Indonesia yang terdapat di Internet adalah Kamus Bahasa Indonesia. Kamus ini sangat berguna bagi orang-orang yang ingin mempelajari bahasa Indonesia, namun, karena bahasa Indonesia memiliki jumlah lema (kata-kata di dalam kamus) lebih dari 20.000 hal itu sungguh menyulitkan seseorang yang baru belajar bahasa Indonesia dalam mengerjakan tugas dan memeriksa apakah kata-kata bahasa Indonesia yang mereka tuliskan benar atau tidak (pemeriksa ejaan)

Selain itu, selama ini tidak ada aplikasi di Indonesia yang dapat digunakan sebagai editor untuk Bahasa Indonesia dan berbasis

Android. Pengguna Android, sistem operasi yang merupakan produk Google, ini telah mencapai lebih dari 750 juta perangkat [8] dan rata-rata pengguna perangkat telekomunikasi dengan sistem operasi Android menggunakan perangkat telekomunikasi tersebut untuk melakukan aktifitas sehari-hari, salah satunya digunakan untuk mengerjakan tugas kantor atau tugas sekolah seperti membuat laporan, menyusun proposal dan lain-lain. Berdasarkan penjelasan di atas, penulis akan mengembangkan aplikasi editor yang dapat memberikan rekomendasi atau pengecekkan Bahasa Indonesia berbasis Android.

2. LANDASAN TEORI

2.1 Bahasa Indonesia

Bahasa Indonesia adalah bahasa resmi Republik Indonesia dan bahasa persatuan bangsa Indonesia. Bahasa Indonesia diresmikan penggunaannya setelah Proklamasi Kemerdekaan Indonesia, tepatnya sehari sesudahnya, bersamaan dengan mulai berlakunya konstitusi

Menurut Pedoman Umum Ejaan Bahasa Indonesia halaman 21 yang diterbitkan pada tahun 2000, kata dasar merupakan kata yang paling sederhana yang belum memiliki imbuhan. Kata dasar juga dapat dikelompokkan sebagai bentuk asal (tunggal) dan bentuk dasar (kompleks). Bentuk asal adalah satuan yang paling kecil yang menjadi asal sesuatu kata kompleks, sebagai contoh pada kata berpakaian terbentuk dari bentuk asal pakai mendapat bubuhan afiks -an menjadi pakaian, kemudian mendapatkan bubuhan afiks ber- menjadi berpakaian. Contoh lain, misalnya kata berkesudahan. Kata ini terbentuk dari bentuk asal sudah kemudian mendapat bubuhan afiks ke-an menjadi kesudahan, kemudian mendapatkan bubuhan afiks ber- menjadi berkesudahan. [2]

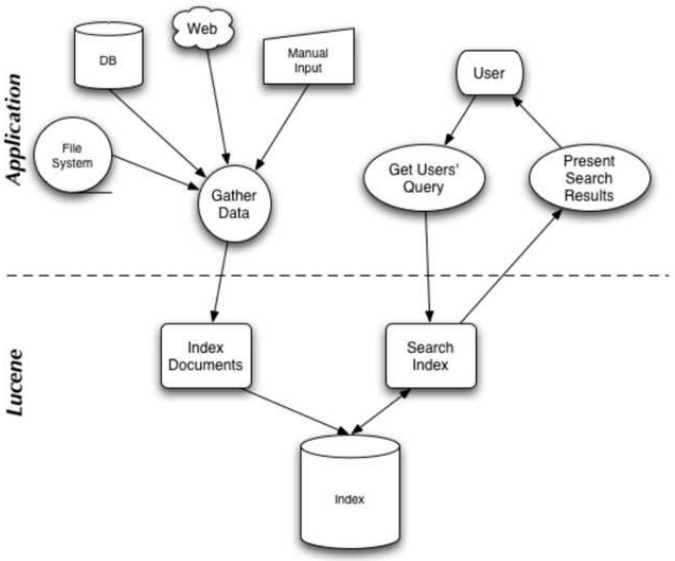
2.2 Ejaan yang Disempurnakan (EYD)

Ejaan yang Disempurnakan (EYD) dapat diartikan sebagai tata bahasa yang disempurnakan. Dalam penulisan karya ilmiah perlu adanya aturan tata bahasa yang menyempurnakan sebuah karya tulis, karena dalam sebuah karya tulis memerlukan tingkat kesempurnaan yang mendetail.

2.3 Apache Lucene

Lucene adalah Library *Information Retrieval* yang diimplementasikan pada Java. Lucene merupakan member dari Apache Jakarta, dibawah Lisensi Apache Software. Lucene sejak beberapa tahun lalu hingga saat ini adalah Library IR Java yang paling popular [5]. dan Lucene tidak peduli tentang sumber data, format file, ataupun Bahasa asalkan semua itu dapat di konversi menjadi text. Lucene sendiri bekerja dengan cara melakukan *indexing*. Bentuk pengaplikasian Lucene dapat dilihat pada Gambar 1.

Apache Lucene menggunakan algoritma Leveinstein dalam proses pencarian. Algoritma ini bekerja dengan cara membandingkan kemiripan antara 2 string, dimana *distance* diukur dari jumlah karakter yang dibuang, disisipkan, atau disubstitusi untuk mengubah string pertama menjadi string kedua [5]. Perhitungan *distance* yang digunakan oleh Apache Lucene dengan metode *fuzzyQuery* adalah hasil pengurangan antara satu dengan hasil pembagian antara *distance* dengan nilai minimum antara panjang string satu dan panjang string dua



Gambar 1. Bentuk integrasi antara aplikasi dengan Lucene

2.4 Algoritma Nazief dan Ariani

Algoritma ini merupakan algoritma *stemming* yang dikembangkan oleh Bobby Nazief dan Mirna Adriani pada tahun 1996 sebagai hasil penelitian internal Universitas Indonesia. Stemming adalah proses untuk menggabungkan atau memecahkan setiap varian-varian suatu kata menjadi kata dasar [10]. Proses *stemming* ini dilakukan dengan cara memotong imbuhan yang dilakukan secara rekursif. Serta mencari kata didalam kamus yang dilakukan sebelum tahap pemotongan. Kelemahan dari algoritma ini diantaranya tingkat akurasinya tergantung dari kamus yang dimiliki. [1]

Algoritma ini dibuat berdasarkan aturan morfologi dan mengenkapsulasi afiks yang diizinkan dan dilarang, termasuk prefiks, sufiks, infiks (penyisipan) dan konfiks (kombinasi prefiks dan sufiks). Algoritma ini juga mendukung recoding, sebuah pendekatan untuk mengembalikan sebuah huruf inisial yang telah dihapus dari sebuah akar kata sebelum mengawali sebuah prefiks. [4]

Ada beberapa aturan yang digunakan pada algoritma Nazief dan Ariani, contohnya adalah pasangan imbuhan dan akhiran yang tidak diperbolehkan (tabel 1) dan tabel disambiguitas (tabel 2).

Tabel 1. Daftar awalan dan akhiran yang tidak diperbolehkan dalam algoritma Nazief dan Ariani

Awalan	Akhiran yang tidak diperbolehkan
ber-	-i
di-	-an
ke-	-i -kan
me-	-an
ter-	-an
per-	-an

Table 2 Tabel Disambiguitas

Aturan ke	Jenis	Hasil
1	berV…	ber-V… | be-rV…
2	berCAP…	ber-CAP… | dimana C!='r' dan P!='er'
3	berCAerV…	ber-CAerV… | dimana C!='r'
4	belajar…	bel-ajar…
5	beC1erC2…	be-C1erC2 … | dimana C1!={'r'| 'l'}
6	terV…	ter-V… | te-rV…
7	terCerV…	ter-CerV… | dimana C!='r'
8	terCP…	ter-CP… | dimana C!='r' dan P!='er'
9	teC1erC2…	te-C1erC2… | dimana C1!='r'
10	me{l|r|w|y}V…	me-{l|r|w|y}V…
11	mem{b|f|v}…	mem-{b|f|v}…
12	mempe{r|l}…	mem-pe…
13	mem{rV|V}…	me-m{rV|V}… | me-p{rV|V}…
14	men{c|d|j|z}…	men-{c|d|j|z}…
15	menV…	me-nV… | me-tV…
16	meng{g|h|q}…	meng-{g|h|q}…
17	mengV…	meng-V… | meng-kV…
18	menyV…	meny-sV…
19	mempV…	mem-pV… | dimana V!='e'
20	pe{w|y}V…	pe-{w|y}V…
21	perV…	per-V… | pe-rV…
23	perCAP…	per-CAP… | dimana C!='r' dan P!='er'
24	perCAerV…	per-CAerV… | dimana C!='r'
25	pem{b|f|v}…	pem-{b|f|v}…
26	pem{rV|V}…	pe-m{rV|V}… | pe-p{rV|V}…
27	pen{c|d|j|z}…	pen-{c|d|j|z}…
28	penV…	pe-nV… | pe-tV…
29	peng{g|h|q}…	peng-{g|h|q}…
30	pengV…	peng-V… | peng-kV…
31	penyV…	peny-sV…
32	pelV…	pe-lV… | Exception: for "pelajar", return ajar
33	peCerV…	per-erV… | dimana C!={r|w|y|l|m|n}
34	peCP…	pe-CP… | dimana C!={r|w|y|l|m|n} dan P!='er'

V = Vokal, C = konsonan, P = bagian kecil dari kata, missal 'er'

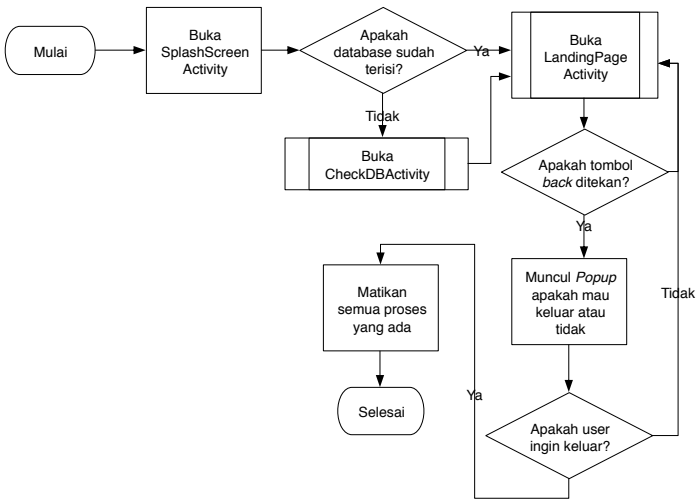
2.5 Android

Android adalah sistem operasi untuk telepon seluler yang berbasis Linux [3]. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel [7]. Android diambil alih oleh Google sebagai bagian strategi untuk mengisi pasar sistem operasi bergerak. Google mengambil alih seluruh hasil kerja Android termasuk tim yang mengembangkan Android [6].

3. DESAIN SISTEM

3.1 Garis Besar Sistem Kerja Perangkat Lunak

Sistem perangkat lunak editor Bahasa Indonesia ini dibagi menjadi dua fitur utama, yaitu sistem rekomendasi kata dengan menggunakan Apache Lucene dan pengecekkan ejaan Bahasa Indonesia dengan menggunakan Algoritma Nazief dan Ariani. Alur kerja dari perangkat lunak dapat dilihat pada Gambar 2 berikut:



Gambar 2. Diagram Alir proses kerja dari perangkat lunak

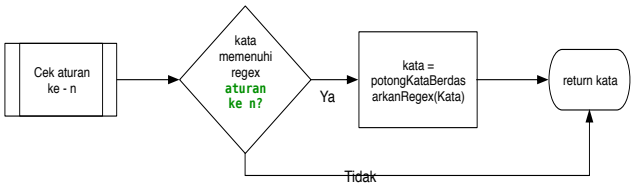
3.2 Garis Besar Algoritma Nazief dan Ariani

Fitur lain yang terdapat pada aplikasi ini adalah fitur *spelling corrector*, di mana fitur ini dapat mekakukan pengecekan apakah kata yang terdapat pada suatu *file* atau yang sedang diinputkan oleh user tersebut valid atau tidak. Jika kata tersebut tidak valid, maka akan diberi garis bawah.

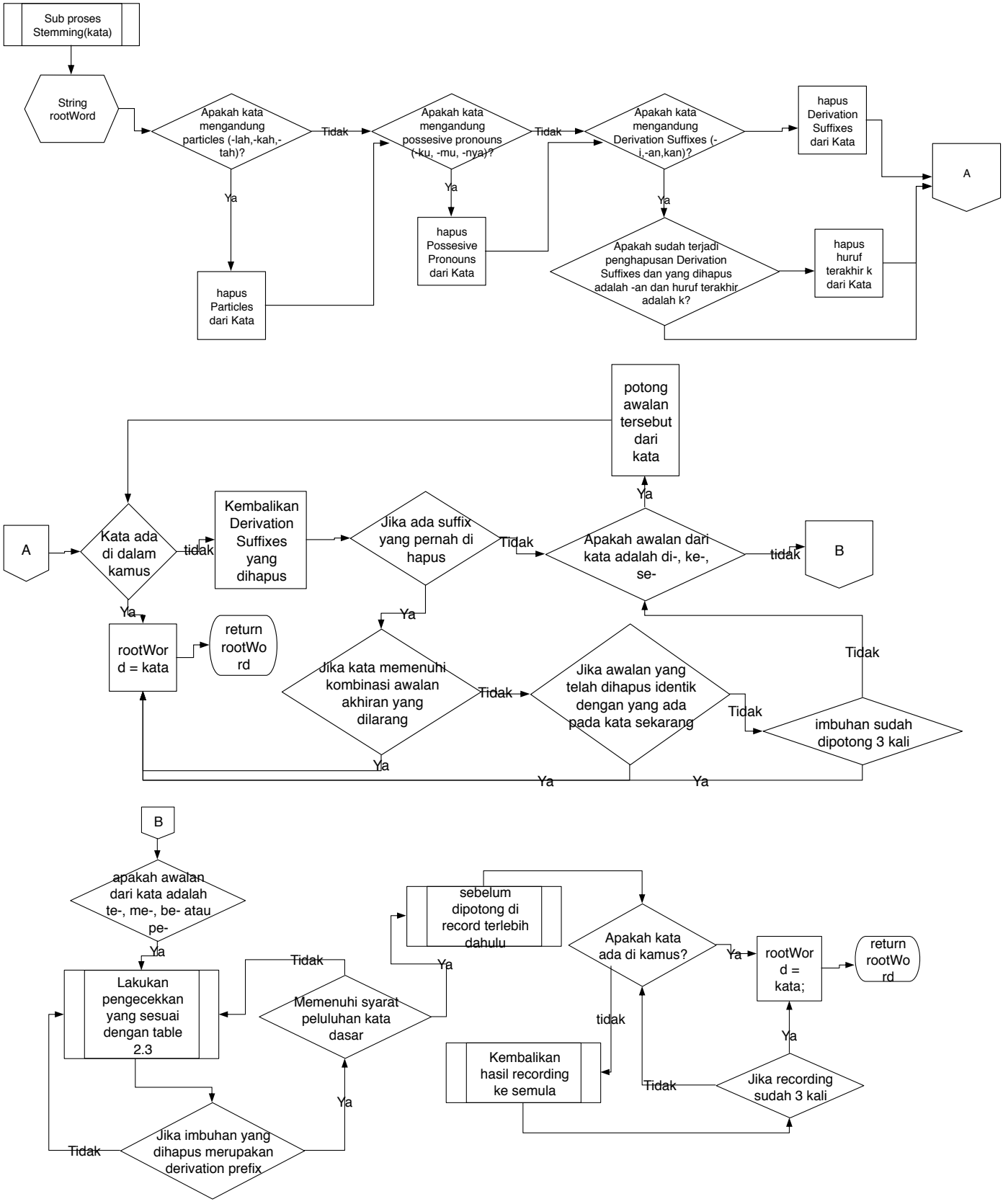
Gambar 4 menunjukkan diagram alir proses stemming yang dilakukan oleh algoritma Nazief dan Ariani. Algoritma ini melakukan proses pengecekkan dengan menggunakan bentuk-bentuk seperti yang terdapat pada Table 2.

3.3 Garis Besar Pengecekkan Untuk Tabel Disambiguitas

Secara umum, aturan yang terdapat pada algoritma Nazief dan Adriani dapat dijadikan sebuah *flowchart* singkat, yaitu seperti pada Gambar 4. Proses pemotongan kata berdasarkan regex yang terdapat pada flowchart tersebut nantinya akan menyesuaikan dengan aturan hasil dari ke-34 aturan tersebut, contohnya untuk aturan ke-15 akan ada penambahan huruf t karena aturan tersebut dapat memenuhi untuk kata-kata yang meluluh jika ditambahkan awalan.



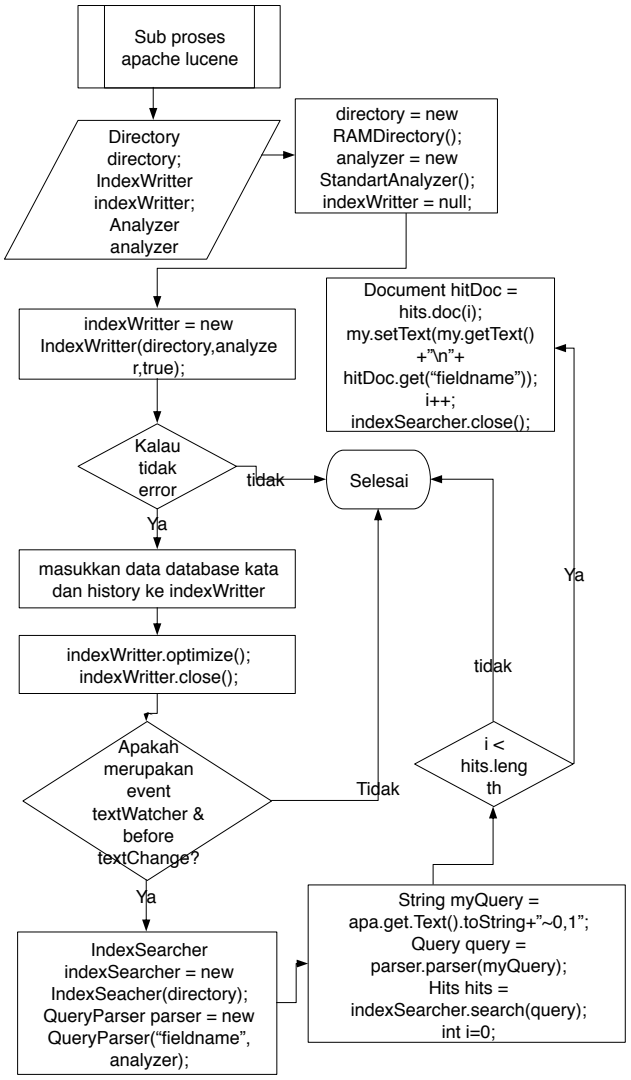
Gambar 3. Diagram alir proses yang dilakukan untuk pengecekkan Tabel Disambiguitas



Gambar 4. Diagram alir proses algoritma Nazief dan Ariani

3.4 Garis Besar Fitur Rekomendasi Kata

Fitur lain yang terdapat pada aplikasi ini adalah fitur rekomendasi kata, di mana fitur ini dapat memberikan hasil rekomendasi berdasarkan hasil crawling yang dilakukan oleh Apache Lucene. Garis besar fitur untuk proses yang dilakukan oleh Apache Lucene dapat dilihat pada Gambar 5.



Gambar 5. Diagram alir proses yang dilakukan oleh Apache Lucene

4. IMPLEMENTASI SISTEM

Pertama-tama ketika *user* membuka halaman editor, jika user membuka suatu *file* yang akan dilakukan oleh program ini adalah mendeteksi apakah *file* yang dibuka sudah pernah dibuka sebelumnya atau belum. Hal ini bertujuan untuk melakukan menampilkan list *file* yang sudah pernah dibuka oleh *user*. Setelah itu, program akan melakukan proses pembacaan dari isi *file* .doc yang dibuka oleh user.

Ketika telah mendapatkan data semua kata yang tersimpan pada *file* tersebut, program akan mulai melakukan pengecekan terhadap data tersebut apakah menggunakan Bahasa Indonesia yang *valid* atau tidak. Jika program menemukan bahwa suatu kata di dalam *file* tersebut tidak valid, akan ditandai dengan memberikan garis bawah pada kata tersebut.

Ketika user mengetik pada halaman editor, program akan memproses input dari user, kemudian akan menampilkan hasil rekomendasi dari apa yang diinputkan oleh *user*. Proses hingga menampilkan rekomendasi ini menggunakan Apache Lucene.

5. PENGUJIAN SISTEM

5.1 Pengujian Aplikasi Untuk Membuka *File* dengan berbagai ukuran

Pengujian untuk melihat pengaruh jumlah kata terhadap waktu dilakukan dengan melakukan pengecekan terhadap 100 *file* dengan ukuran berbeda dan jumlah kata yang bervariasi mulai dari 100 hingga 1000 kata. Pada Tabel 3 terlihat rata-rata waktu yang dibutuhkan untuk melakukan suatu proses adalah 2 hingga 15 menit, hal itu terjadi karena jumlah kata, panjang kata dari masing-masing dokumen berbeda-beda sehingga waktu pengecekkannya pun berbeda tergantung pada panjang kata tersebut.

Tabel 3. Hasil Pengujian Terhadap Kecepatan Proses Membaca Sebuah *File* .doc

Jumlah Kata	Rata-rata waktu (menit)
100	2.315
200	3.313
300	3.099
400	3.506
500	6.208
600	8.905
700	9.54
800	14.238
900	15.674
1000	8.938

Tabel 4. Perbandingan antara menggunakan system history, tanpa pengecekkan, dan pengecekkan tanpa history.

No	Jumlah Kata	Waktu (menit)			
		Dengan Pengecekan tanpa *history* (i)	Pengecekan dengan *history* (ii)	Pengecekan dengan File Serupa Telah Dibuka Sebelumnya (iii)	Tanpa pengecekkan (iv)
1	100	1.85	1.08	0.167	0.083
2	200	1.92	1.02	0.067	0.083
3	300	2.76	1.45	0.367	0.067
4	400	2.93	2.35	0.417	0.083
5	500	6.98	1.25	0.37	0.083
6	600	7.45	7.05	0.8	0.05
7	700	5.35	4.23	1.333	0.067
8	800	10.18	8.4167	1.167	0.05
9	900	6.42	5.23	1.2167	0.067
10	1000	9.57	8.483	1.967	0.067

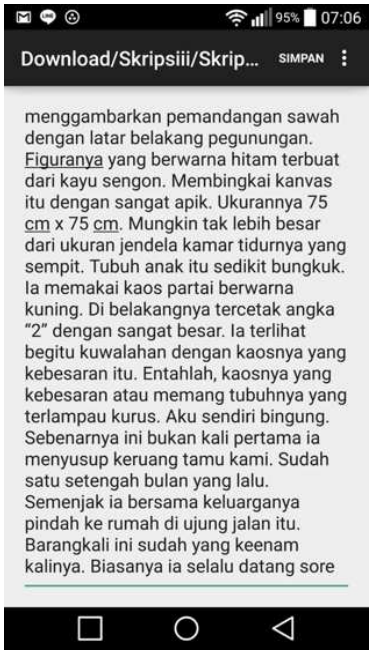
Untuk mengantisipasi hal tersebut, dibuatlah system history di mana ketika suatu kata sudah pernah dilakukan pengecekkan maka akan disimpan di dalam memory (seperti cache), sehingga aplikasi tidak melakukan pengecekan yang berulang-ulang untuk kata yang

sama. Setelah melalukan *system history* pada tabel 4 dapat dilihat perbandingan waktu yang digunakan.

Dari hasil pengujian, terlihat bahwa rata-rata yang waktu yang diperlukan untuk membuka sebuah *file* cukup berkurang dengan menggunakan metode *system history*. Pengecekkan akan berjalan lebih cepat apabila *file* tersebut sudah pernah dibuka sebelumnya.

5.2 Pengujian Pengecekan Kata dalam Bahasa Indonesia

Dari hasil pengujian pada Gambar 6, terlihat bahwa kata figuranya bukan merupakan kata yang valid. Selain itu singkatan seperti cm juga dianggap salah oleh sistem karena tidak ada di dalam kamus dan memiliki jumlah kata kurang dari 3 sehingga bukan merupakan kata berimbuhan.



Gambar 6. Hasil dari Pengecekan Kata dalam Bahasa Indonesia

6. KESIMPULAN

Berdasarkan hasil pengujian, dapat disimpulkan beberapa hal sebagai berikut:

- Fitur untuk melakukan pengecekan Bahasa Indonesia, dapat berjalan ketika pengguna membuka file dan ketika pengguna melakukan pengetikan pada halaman editor.
- File .doc hasil dari aplikasi ini dapat dibuka di Ms. Word 2013 dengan baik.
- Faktor yang mempengaruhi lama tidaknya aplikasi untuk membaca suatu file adalah panjang kata, jumlah kata berpengaruh untuk jumlah perulangan yang akan dilakukan proses stemming.
- Algoritma Nazief dan Ariani masih memiliki kekurangan yaitu pengecekan jika suatu kata memiliki dua suku kata, seperti ditandatangani, kelengkapan database kamus juga sangat mempengaruhi karena rootword akan selalu dibandingkan dengan isi kamus.
- History yang digunakan pada aplikasi ini berhasil membuat aplikasi ini membuka dokumen secara lebih cepat, dan semakin banyak history yang dimiliki kemungkinan besar akan speed akan bertambah cepat hingga semua kata yang ada dalam dokumen tersebut masuk ke dalam history.
- Fitur yang tidak compatible dengan perangkat jelly beans adalah fitur Google Drive. Pada Android versi jelly beans, fitur upload dan pengambilan data dari Google Drive tidak dapat dilakukan karena pada perangkat tersebut, tidak kompatibel dengan Intent.ACTION_GET_CONTENT dan Intent.ACTION_CREATE_DOCUMENT.

7. REFERENCES

[1] Asian, J. 2007. *Effective Techniques for Indonesian Text Retrieval*. Melbourne, Victoria, Australia: School of Computer Science and Information Technology.

[2] Department Pendidikan Nasional. 2010. *Update EYD Ejaan yang Disempurnakan Terbaru*. Jakarta, Indonesia: Department Pendidikan Nasional.

[3] Felker, D. 2011. *Android Application Development for Dummies*. Indianapolis: Wiley Publishing, Inc.

[4] Firdaus, A., Ernawati, & Vatresia, A. 2014. Aplikasi Pendeteksi Kemiripan Pada Dokumen Teks Menggunakan Algoritma Nazief & Adriani dan Metode Cosine Similarity. *Jurnal Teknologi Informasi*, Vol. 10, No. 1, p. 96-109.

[5] Gospodnetic, O., & Hatcher, E. 2005. *Lucene in Action. Greenwitch*, South East London, England: Manning Publications Co.

[6] Lengkong, H. N., Sinsuw, A., & Lumenta, A. 2015. Perancangan Penunjuk Rute Pada Kendaraan Pribadi Menggunakan Aplikasi Mobile GIS Berbasis Android Yang Terintegrasi Pada Google Maps. *E-journal Teknik Elektro dan Komputer*, Vol. 4, No. 2, p. 18-25.

[7] Meier, R. 2009. *Professional Android Application Development.* New York: Wiley Publishing, Inc.

[8] Oreskovic, A. 2013. *Google: Pengguna Android Akan Capai 1 Miliar dalam 9 Bulan.* VOA Indonesia. URI=http://www.voaindonesia.com/content/google-pengguna-android-akan-capai-1-miliar-dalam-9-bulan/1643029.html

[9] Purnomo, H. 2014. *Negara dengan Penduduk Terbanyak di Dunia, RI Masuk 4 Besar.* Detik.com. URI=http://finance.detik.com/read/2014/03/06/134053/2517461/4/negara-dengan-penduduk-terbanyak-di-dunia-ri-masuk-4-besar

[10] Radiant, I., Adelia, Mewati, A., & Rehatta, A. 2014. Implementasi Cosine Similarity dan Algoritma Smith-Waterman untuk Mendeteksi Kemiripan Teks. *Jurnal Informatika*, Vol. 10 No. 1, p. 31-42.

[11] Wahyuni, D.. *Peluang atau Tantangan Indonesia Menuju Asean Economic Community (AEC) 2015*. Institut Ilmu Sosial dan Manajemen STIAMI. URI = http://www.stiami.ac.id/download/get/28/proceeding-dian-wahyudin