Coding, Jurnal Komputer dan Aplikasi
Volume 07, No. 03 (2019), Hal 44-53 ISSN 2338-493X

PENERAPAN *JARO WINKLER DISTANCE* DALAM APLIKASI PENGOREKSI KESALAHAN PENULISAN BAHASA INDONESIA BERBASIS WEB

[1]Junar Frando, [2]Ikhwan Ruslianto, [3]Rahmi Hidayati
[1] [2] [3] Jurusan Rekayasa Sistem Komputer, Fakultas MIPA Universitas Tanjungpura

Jl. Prof. Dr. H. Hadari Nawawi, Pontianak
Telp./Fax.: (0561) 577963
e-mail: [1]junarfr@gmail.com, [2]ikhwanruslianto@siskom.untan.ac.id,
[3]rahmihidayati@siskom.untan.ac.id

ABSTRAK

Microsoft word dan open office adalah aplikasi word processor yang banyak digunakan dalam mengerjakan suatu karya ilmiah. Pada word processor umumnya sudah terdapat fitur koreksi penulisan, dimana dalam fitur ini aplikasi dapat menandai kesalahan penulisan dan juga memberikan saran perbaikannya secara otomatis. Cara kerja dalam menandai kesalahan penulisan adalah memindai kata-kata pada dokumen yang tidak terdaftar dalam kamus suatu bahasa tertentu. Fitur pemberian saran perbaikan adalah berdasarkan pencarian kata pada kamus yang paling mendekati dari kata yang salah. Pada word processor, koreksi penulisan diimplementasikan berdasarkan bahasa Inggris. Hal ini tentu saja membantu bila pengguna mengetik dokumen dalam bahasa Inggris, sedangkan koreksi penulisan yang berdasarkan bahasa Indonesia masih belum umum ditemukan. Dari permasalahan tersebut mendasari penulis membuat sebuah aplikasi cek penulisan yang dilengkapi fitur saran perbaikan berdasarkan bahasa Indonesia. Metode yang digunakan pada penelitian ini dalam memberikan saran perbaikan adalah Jaro Winkler Distance pada aplikasi berbasis web. Berdasarkan hasil pengujian menggunakan 3 sampel dokumen dengan total 37501 kata memerlukan waktu 25.5 detik dengan total persentase akurasi dalam memberikan saran perbaikan sebanyak 77.23%.

Kata kunci: *Jaro Winkler Distance, correction, Indonesian language, web, word processor.*

1. PENDAHULUAN

Karya tulis yang bersifat ilmiah memerlukan kata baku dengan Ejaan Yang Disempurnakan (EYD). Salah satu syarat EYD adalah penggunaan kata baku yang sesuai dengan Kamus Besar Bahasa Indonesia (KBBI) [1]. Ketidaksesuaian dalam penulisan kata baku tersebut bisa disebabkan beberapa faktor seperti tidak mengetahui penulisan yang benar dari kata tersebut ataupun bisa dari kelalaian dalam pengetikan. Kesalahan dalam pengetikan kata pada karya tulis memang sulit untuk dihindari. Walaupun sudah dilakukan pengecekan secara manual, terkadang kesalahan kata dalam pengetikan masih luput dari pengecekan. Beberapa hal yang menjadi penyebab kesalahan dalam mengetik seperti letak huruf pada *keyboard* yang berdekatan dan juga kurang konsentrasi dalam mengetik sering sekali dialami oleh penulis.

Saat ini terdapat banyak pilihan aplikasi *word processor* yang bisa digunakan dalam mengerjakan suatu karya ilmiah seperti *microsoft word* dan *open office*. Pada *word processor* umumnya sudah terdapat fitur koreksi penulisan, dimana dalam fitur ini aplikasi dapat menandai kesalahan penulisan dan juga memberikan saran perbaikannya secara otomatis. Cara kerja dalam menandai kesalahan penulisan adalah memindai kata-kata pada dokumen yang tidak terdaftar dalam kamus suatu bahasa tertentu. Fitur pemberian saran perbaikan adalah berdasarkan pencarian kata pada kamus yang paling mendekati dari kata yang salah. Pada *word processor,* koreksi penulisan diimplementasikan berdasarkan bahasa Inggirs. Hal ini tentu saja membantu bila pengguna mengetik dokumen dalam bahasa Inggris, sedangkan koreksi penulisan yang berdasarkan bahasa Indonesia masih belum umum ditemukan. Dari permasalahan tersebut mendasari penulis membuat sebuah aplikasi cek penulisan yang dilengkapi fitur saran perbaikan berdasarkan bahasa Indonesia.

Penelitian mengenai pencarian saran perbaikan dengan judul "Studi Perbandingan Algoritma Pencarian *String* dalam Metode *Approximate String Matching* untuk Identifikasi Kesalahan Pengetikan Teks" [2]. Dalam penelitian ini membandingkan empat perhitungan pencarian *string*, yaitu: *Hamming Distance, Levenshtein Distance, Damerau Levenshtein Distance dan Jaro Winkler Distance*. Evaluasi yang dilakukan pada penelitian ini menggunakan *user relevance judgement* yang menghasilkan nilai *Mean Average Precision* (MAP) untuk menentukan perhitungan yang terbaik. Hasil penelitian terhadap 50 kata salah menunjukan bahwa *Jaro Winkler Distance* memiliki nilai tertinggi dalam memberikan saran perbaikan dengan nilai MAP sebesar 0,87.

Penelitian mengenai penerapan *Jaro Winkler Distance* dalam pemberian kata saran dengan judul "Pengkoreksian dan *Suggestion Word* pada *Keyword* Menggunakan Algoritma *Jaro Winkler*" [3]. Pada penelitian ini *Jaro Winkler* dapat diterapkan untuk memberikan *suggestion word*, jika ada salah pengetikan atau *typo* pada pencarian bahan pustaka atau data buku.

Penelitian lain mengenai penerapan *Jaro Winkler Distance* dengan judul "Implementasi Algoritma *Jaro Winkler Distance* untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia" [4]. Hasil ujicoba dalam penelitian ini aplikasi yang dibangun berhasil mendeteksi kemiripan teks pada dokumen yang identik apabila dengan urutan teks yang tidak berbeda.

Penelitian lain mengenai kesalahan penulisan yang berjudul "Implementasi Algoritma *Levenshtein Distance* dan Metode *Empiris* untuk Menampilkan Saran Perbaikan Kesalahan Pengetikan Dokumen Berbahasa Indonesia" [5]. Pada penelitian ini aplikasi yang dibangun menggunakan algoritma *Levenshtein Distance* dan metode *Empiris* untuk mencari saran perbaikan. Selain itu aplikasi yang dibangun dalam penelitian ini tidak mendukung dalam pengoreksian dokumen.

Berdasarkan uraian penelitian mengenai kesalahan penulisan dan *Jaro Winkler Distance* yang telah dilakukan sebelumnya, pada penelitian membahas mengenai penerapan *Jaro Winkler Distan* dalam aplikasi pengoreksi kesalahan penulisan bahasa Indonesia berbasis web. Penelitian ini membuat aplikasi yang dapat mengoreksi kesalahan penulisan bahasa Indonesia pada dokumen *.docx* dan memberikan saran perbaikan dari tiap penulisan salah menggunakan *Jaro Winkler Distance*.

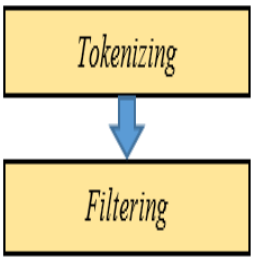
2. LANDASAN TEORI

2.1. Bahasa Baku

Bahasa baku adalah ragam bahasa yang cara pengucapan dan penulisannya sesuai dengan kaidah-kaidah standar. Kaidah standar dapat berupa pedoman ejaan (EYD), tata bahasa baku, dan kamus umum. Sebaliknya, bahasa tidak baku adalah ragam bahasa yang cara pengucapan dan penulisannya tidak memenuhi kaidah kaidah standar tersebut. Penggunaan ragam bahasa baku dan tidak baku berkaitan dengan situasi dan kondisi pemakaiannya. Ragam bahasa baku biasanya digunakan situasi resmi, seperti acara seminar, pidato, temu karya ilmiah, dan lain-lain. Adapun ragam bahasa tidak baku umumnya digunakan dalam komunikasi sehari-hari yang tidak bersifat resmi [1].

2.2. *Text Preprocessing*

Text preprocessing adalah tahap awal dari pengolahan teks. Tahap ini mencakup semua rutinitas, dan proses untuk mempersiapkan data yang akan digunakan pada operasi *knowledge discovery* sistem pengolahan teks [6]. Tujuan dari tahap *text preprocessing* pada penelitian ini adalah memecah teks menjadi *array* kata sehingga tiap kata pada *array* tersebut bisa dilakukan pengecekan pada kamus. Tahapan dari *text preprocessing* yang digunakan pada penelitian ini dapat dilihat pada Gambar 1.

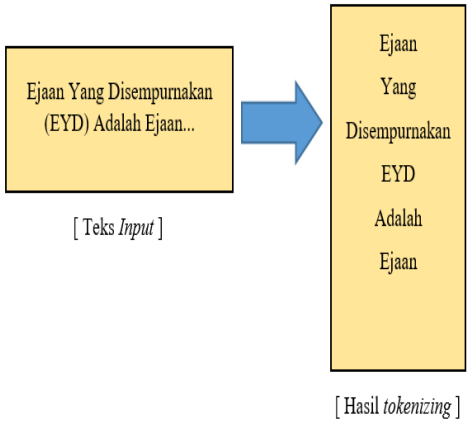


Gambar 1. Tahapan *Text Preprocessing*

2.2.1 Tahap *Tokenizing*

Tahap *tokenizing* merupakan tahapan untuk memisah-misahkan setiap kata penyusun token pada teks dokumen. *Tokenizing*

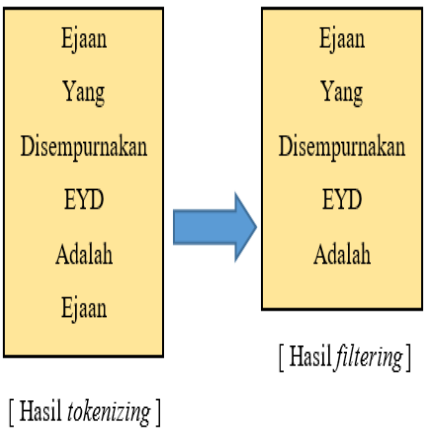
mengandalkan karakter di luar alfabet pada dokumen untuk melakukan pemisahan kata. Karakter di luar alfabet digunakan sebagai *delimiter* yang berfungsi untuk memisahkan kata perkata menggunakan fungsi *split*. *Output* dari proses *tokenizing* adalah berupa *array*. Contoh tahap *tokenizing* dapat dilihat pada Gambar 2.



Gambar 2. Tahap *Tokenizing*

2.2.2 Tahap *Filtering*

Tahap *filtering* adalah tahap menghilangkan kata duplikat dari hasil *tokenizing* sehingga kata yang sama tersebut tidak perlu dilakukan pengecekan pada kamus ataupun pengecekan *Jaro Winkler Distance*. Contoh tahap *filtering* dapat dilihat pada Gambar 3.



Gambar 3. Tahap *Filtering*

2.3. *String Matching*

String matching dapat diartikan sebagai sebuah cara untuk mencari *string* yang sama dalam sebuah kumpulan teks (dokumen) atau *database*. Konsep *string matching* secara garis besar dapat dibedakan menjadi dua konsep besar, diantaranya [7]:

1. *Exact string matching*, merupakan pencocokan *string* secara tepat dengan susunan karakter dalam *string* yang dicocokkan memiliki jumlah dan urutan karakter dalam *string* yang sama. Contoh, kata "PULPEN" akan menunjukan kecocokan hanya dengan kata "PULPEN".
2. *Inexact string matching*, merupakan pencocokan *string* secara samar, yaitu pencocokan *string* dimana *string* yang dicocokkan memiliki kemiripan namun keduanya memiliki susunan karakter yang berbeda. Contoh, "TEMPAT" dengan "TEMPAD" terdapat penulisan karakter yang berbeda. Jika perbedaan karakter ini dapat ditoleransi sebagai sebuah kesalahan penulisan maka dua kata tersebut dapat dikatakan cocok. Berdasarkan bentuk kemiripannya, *inexact string matching* juga dapat dibedakan menjadi dua, yaitu:

a. *Approximate string matching*, merupakan pencocokan *string* berdasarkan kemiripan dari segi penulisan yang meliputi jumlah dan susunan karakter di dalam dokumen.
b. *Phonetic string matching*, merupakan pencocokan *string* berdasarkan kemiripan dari segi pengucapan, meskipun terdapat perbedaan penulisan kedua *string* tersebut jika dibandingkan.

2.4. *Jaro Winkler Distance*

Jaro Winkler Distance pada dasarnya adalah metode yang membandingkan kesamaan karakter penyusun dari dua buah string. *Jaro* menggunakan rumus untuk menghitung jarak (*dj*) antara dua *string* yaitu s_1 dan s_2, dengan *m* adalah jumlah karakter yang sama persis dan *t* adalah setengah dari jumlah karakter yang tertukar (transposisi) [8]. Berikut adalah persamaan untuk menghitung nilai *Jaro Distance*:

$$dj = \frac{1}{3} \times \left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m} \right) \quad (1)$$

Keterangan :
dj : nilai *Jaro Distance*.
m : jumlah karakter yang sama persis.
$|s_1|$: adalah panjang *string* 1.
$|s_2|$: adalah panjang *string* 2.
t : setengah dari jumlah karakter yang tertukar (transposisi).

Jarak teoritis dua buah karakter yang disamakan dapat dibenarkan jika tidak melebihi:

$$\left(\frac{\max|s_1|,|s_2|}{2}\right) - 1 \qquad (2)$$

Jaro Winkler Distance menggunakan *prefix scale* (*p*) yang memberikan tingkat penilaian yang lebih, dan *prefix length* (*l*) yang menyatakan panjang awalan yaitu panjang karakter yang sama dari string yang dibandingkan sampai ditemukannya ketidaksamaan. Bila string s_1 dan s_2 yang diperbandingkan, maka nilai *Jaro Winkler distance* (*dw*) adalah [9]:

$$dw = dj + (l \times p \times (1 - dj)) \qquad (3)$$

Keterangan :

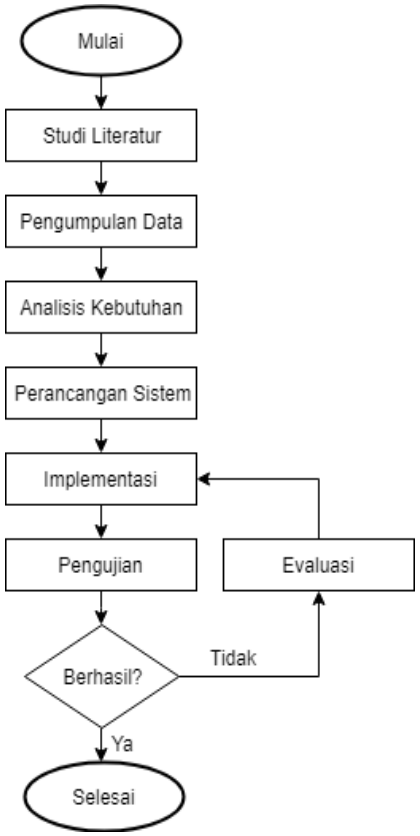
dw : nilai *Jaro Winkler Distance.*
dj : nilai *Jaro Distance.*
l : panjang *prefix* umum di awal *string* nilai maksimalnya 4 karakter dalam suatu kata (panjang karakter yang sama sebelum ditemukan ketidaksamaan).
p : konstanta *scaling factor*. Nilai standar untuk konstanta ini menurut Winkler adalah $p = 0.1$.

2.5. *Length based filters*

Length based filters adalah filter yang berbasiskan nilai panjang karakter sebagai acuan. Proses filter dengan membanding perbedaan jumlah panjang karakter yang besar seperti kata "J" dengan "JERAPAH", bila dihitung untuk mendapatkan nilai *Jaro Winkler Distance* sebesar 1 tentu tidak akan terpenuhi [9]. Proses filter lain dengan kata kunci "FAKKTOR" dengan semua kata yang ada pada daftar kamus. Kata kunci yang dapat dijadikan perbandingan adalah kata kunci dengan panjang karakter (*P*) antara $P_{katakunci}$ – 3 sampai $P_{katakunci}$ + 3. Karena "FAKKTOR" memiliki panjang 7 karakter, maka yang menjadi kata kunci pembanding mulai dari kata dengan panjang karakter 4-10 karakter [10]. Misalnya, tiga kata kunci yang dijadikan sebagai perbandingan yaitu kata "FAKTOR" dengan panjang 6 karakter, "FAKTUAL" dengan panjang 7 karakter, "INISIATOR" dengan panjang 9 karakter, "FAKIR" dengan panjang 5 karakter, dan "FAKTUR" dengan panjang 6 karakter.

3. METODE PENELITIAN

Metode penelitian yang digunakan pada penelitian ini dapat dilihat pada Gambar 4.



Gambar 4. Alur Penelitian

Tahap pada penelitian ini dimulai dari studi kepustakaan yaitu pengumpulan bahan-bahan referensi. Literatur yang digunakan dapat berupa jurnal ilmiah penelitian sebelumnya, buku-buku, dan artikel. Selanjutnya melakukan pengumpulan data yang diperlukan dalam penelitian. Data yang dikumpulkan adalah berupa *database* Kamus Besar Bahasa Indonesia (KBBI). Langkah selanjutnya yaitu analisa kebutuhan meliputi analisa kebutuhan perangkat keras dan analisa kebutuhan perangkat lunak dalam penelitian. Adapun kebutuhan perangkat keras yang digunakan dalam penelitian ini sebuah laptop dengan spesifikasi prosessor core i7 2.2 Ghz, RAM 8.00 GB. Sedangkan kebutuhan perangkat lunak mencakup Node.js, PostgreSQL dan *Visual Studio Code.*

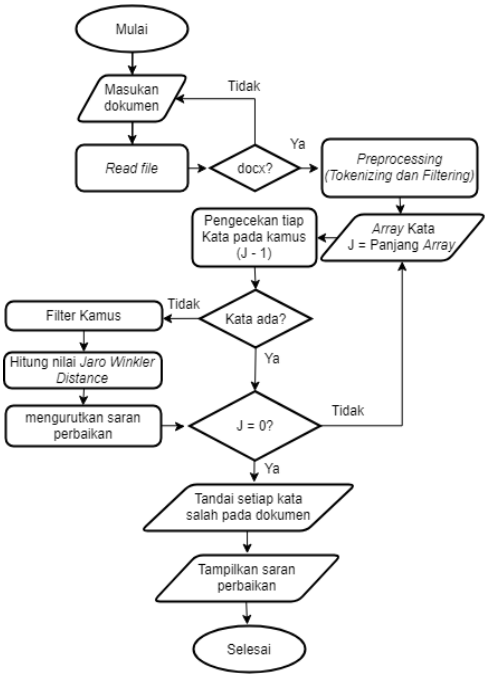
Tahap selanjutnya adalah melakukan perancangan sistem meliputi perancangan *flowchart* sistem, *unified modelling language* (UML), antar muka aplikasi, dan perancangan *database*. Selanjutnya tahap implementasi yaitu proses pembuatan aplikasi dari rancangan sistem menjadi kode-kode program. Tahap

pengujian meliputi pengujian kinerja aplikasi yang telah dirancang. Tujuan dari pengujian kinerja aplikasi adalah untuk mengetahui apakah aplikasi yang dibuat sudah berfungsi dengan yang direncanakan atau belum. Setelah dilakukan pengujian, jika terdapat *error* atau kesalahan pada aplikasi akan dilakukan evaluasi. Segala kekurangan pada aplikasi yang telah dibuat akan diperbaiki sehingga dapat berfungsi sesuai dengan yang direncanakan. Gambar dari

4. PERANCANGAN

4.1. Rancangan *Flowchart* Sistem

Flowchart sistem pengoreksi kesalahan penulisan bahasa Indonesia dapat dilihat pada Gambar 5.



Gambar 5. *Flowchart* Sistem

Pada Gambar 5 berisi *flowchart* sistem yang mana hasil akhir akan menandai setiap penulisan salah dan menampilkan saran perbaikan. Penjelasan rincian dari masing-masing tahapan akan dijelaskan sebagai berikut:

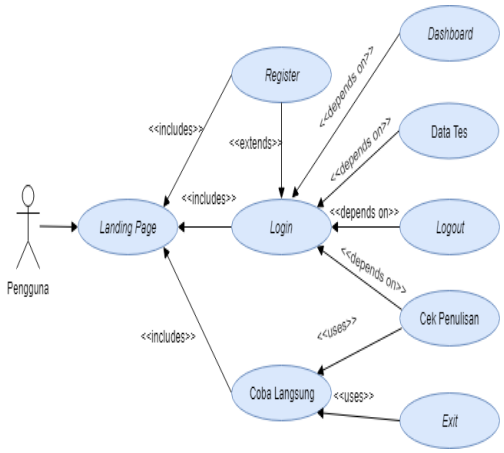
1. Proses dimulai.
2. Pengguna memasukkan dokumen berekstensi *.docx*.
3. Sistem membaca dokumen.
4. Sistem memverifikasi ekstensi dokumen apakah berupa *.docx*. Jika ekstensi sesuai maka proses akan dilanjutkan ke tahap *preprocessing* dan jika tidak sesuai maka pengguna akan diarahkan untuk memasukkan dokumen ulang.
5. Sistem melakukan *preprocessing* yang terdiri dari *tokenizing* dan *filtering*.
6. Sistem menghasilkan *array* kata dengan *j* sebagai nilai dari panjang *array* kata.
7. Sistem melakukan pengecekan keberadaan tiap kata pada *array* kata terhadap kamus dengan *looping* secara *descending* dari nilai *j* - 1. Kamus yang digunakan pada penelitian ini berjumlah 79.902.
8. Sistem memverifikasi kata yang dicek apakah berada di kamus. Jika kata yang dicek terdapat pada kamus maka proses akan dilanjutkan ke tahap pengecekan nilai *j* apakah sudah bernilai 0. Jika kata yang dicek tidak terdapat pada kamus maka kata tersebut dianggap penulisan salah, sehingga proses akan dilanjutkan ketahap filter kamus.
9. Sistem melakukan filter kamus dengan *length based filters*, memilih kata dengan maksimal selisih 3 panjang karakter dari kata salah.
10. Sistem melakukan perhitungan *Jaro Winkler Distance* terhadap seluruh hasil filter kamus untuk mendapatkan saran perbaikan.
11. Sistem mengurutkan saran perbaikan dengan maksimal mengambil tiga saran perbaikan dari nilai *Jaro Winkler Distance* tertinggi dengan *threshold 0.9*.
12. Sistem mengecek apakah *looping* sudah selesai dengan mengecek apakah nilai *j* sudah bernilai 0. Jika *looping* belum selesai maka sistem akan kembali ke *array* kata untuk melakukan pengecekan kata berikutnya. Jika *looping* pengecekan kata sudah selesai maka sistem akan menandai setiap kata salah pada dokumen.
13. Sistem menandai setiap kata salah pada dokumen.
14. Sistem menampilkan saran perbaikan.
15. Proses selesai.

4.2. Rancangan *Unified Modelling Language* (UML)

A. *Use Case Diagram*

Proses mengoperasikan aplikasi dimulai ketika pengguna dengan membuka halaman *landing page.* Pada halaman *landing page* terdapat menu *login, register,* dan coba langsung. Bagi pengguna yang sudah memiliki akun, pengguna dapat mengakses menu *dashboard,* data tes, cek penulisan, dan *logout*

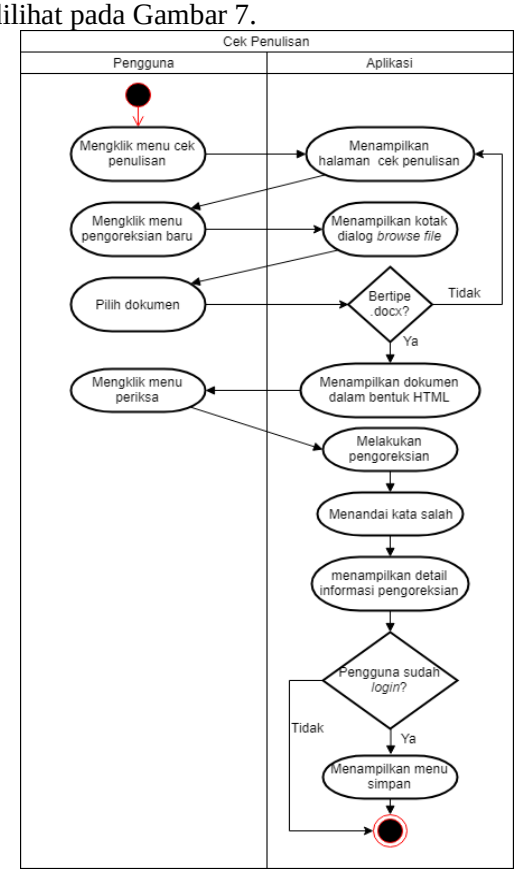
dengan melakukan *login* terlebih dahulu. Apabila pengguna belum memiliki akun, pengguna dapat membuat akun dengan mengklik menu *register*. Apabila pengguna ingin mencoba menu cek penulisan tanpa menggunakan akun pengguna dapat menggunakan menu coba langsung. Gambar dari *use case diagram* sistem dapat dilihat pada Gambar 6.



Gambar 6. *Use Case Diagram Sistem*

B. *Activity Diagram* Cek Penulisan

Activity diagram cek penulisan dapat dilihat pada Gambar 7.



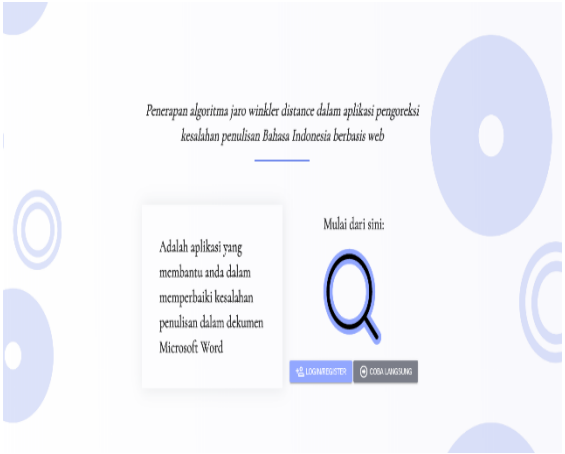
Gambar 7. *Activity Diagram* Cek Penulisan

Masukan awal pada proses cek penulisan adalah dokumen berbasis *.docx* dari pengguna, sedangkan keluarannya berupa dokumen berbasis HTML yang ditampilkan dalam aplikasi. Tahap selanjutnya pengguna mengklik menu periksa, maka aplikasi akan mengoreksi penulisan pada dokumen HTML. Setelah aplikasi selesai mengoreksi dokumen, selanjutnya aplikasi akan menandai tiap kata salah dan juga menampilkan detail informasi pengoreksian, meliputi lis penulisan salah beserta saran perbaikan, jumlah kata dalam dokumen, jumlah penulisan salah, dan waktu proses. Setelah pengoreksian bagi pengguna yang sudah melakukan *login*, aplikasi akan menampilkan menu simpan, sedangkan untuk pengguna yang tidak *login* atau sebagai tamu maka menu ini tidak akan muncul..

5. IMPLEMENTASI, PENGUJIAN DAN PEMBAHASAN

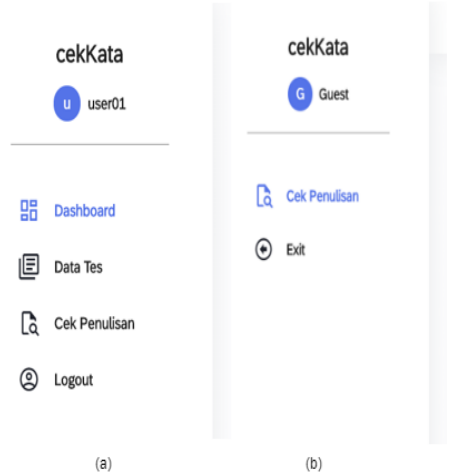
5.1. Tampilan Aplikasi

Saat aplikasi pengoreksi kesalahan penulisan Bahasa Indonesia ini dibuka, maka akan muncul halaman *landing page* atau tampilan awal aplikasi terdapat dua menu utama. Menu *login/register* ditujukan bagi pengguna yang ingin mengakses pengoreksian dengan akun yang dimilikinya, sedangkan menu coba langsung ditujukan bagi pengguna yang hanya sekedar ingin mencoba pengoreksian tanpa perlu menggunakan akun. Gambar tampilan dari halaman *landing page* dapat dilihat pada Gambar 8.



Gambar 8. Tampilan Halaman *Landing Page*

Setelah masuk pada halaman utama aplikasi terdapat menu *navigasi* disebelah kiri untuk mengkases menu lain. Gambar dari tampilan menu *navigasi* dapat dilihat pada Gambar 9.

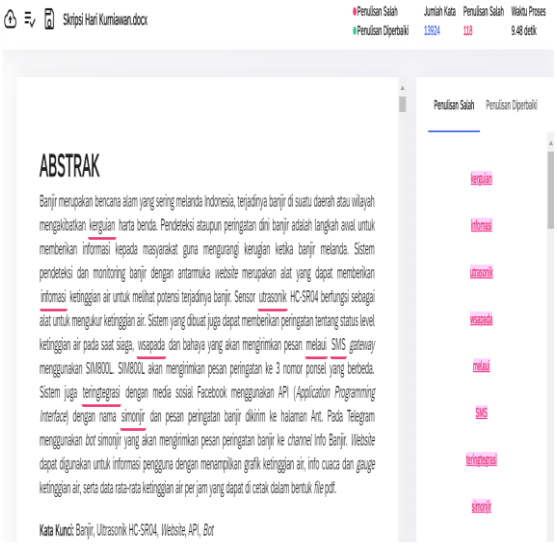


Gambar 9. Tampilan Menu *Navigasi* Pengguna dengan *Login* (a) dan tanpa *Login* (b)

Pada halaman *dashboard* untuk tampilan utamanya adalah laporan jumlah dokumen, jumlah rata-rata kesalahan penulisan dalam bentuk persentase, total kata dalam kamus, grafik persentase penulisan, dan *link* untuk mengakses aktifitas pengoreksian terbaru yaitu pada kolom aktifitas terakhir. Gambar dari tampilan halaman *dashboard* dapat dilihat pada Gambar 10.

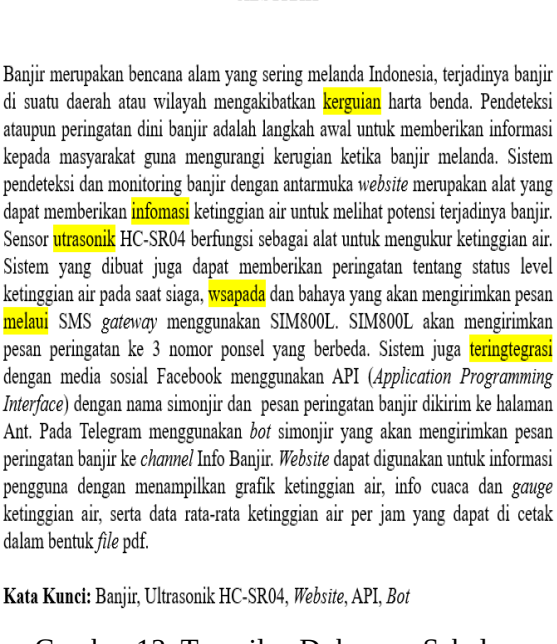
Gambar 10. Tampilan Halaman *Dashboard*

Berbeda dengan halaman *dashboard* bagi pengguna yang telah melakukan *login*, halaman cek penulisan pada Gambar 10 merupakan tampilan utama bagi pengguna atau tamu yang tidak melakukan *login* dengan mengklik menu coba langsung. Untuk tampilan utama pada bagian pengoreksian terdiri dari, dua ikon sebagai menu *upload* dan perbaiki otomatis, tardapat juga ruang untuk menampilkan dokumen HTML dari konversi dokumen *.docx*, ruang untuk menampilkan lis penulisan salah beserta lis penulisan yang telah diperbaiki, dan keterangan jumlah kata, jumlah penulisan salah, dan waktu proses. Gambar dari tampilan halaman cek penulisan dapat dilihat pada Gambar 11.



Gambar 11. Tampilan Halaman Cek Penulisan

Setelah pengguna selesai melakukan perbaikan dan menyimpan hasil perbaikan, pengguna dapat men-*download* hasil perbaikan dalam bentuk *.docx*. Gambar dari tampilan dokumen sebelum dan setelah diperbaiki dapat dilihat pada Gambar 12 dan 13.



ABSTRAK

Banjir merupakan bencana alam yang sering melanda Indonesia, terjadinya banjir di suatu daerah atau wilayah mengakibatkan kerguian harta benda. Pendeteksi ataupun peringatan dini banjir adalah langkah awal untuk memberikan informasi kepada masyarakat guna mengurangi kerugian ketika banjir melanda. Sistem pendeteksi dan monitoring banjir dengan antarmuka *website* merupakan alat yang dapat memberikan infomasi ketinggian air untuk melihat potensi terjadinya banjir. Sensor utrasonik HC-SR04 berfungsi sebagai alat untuk mengukur ketinggian air. Sistem yang dibuat juga dapat memberikan peringatan tentang status level ketinggian air pada saat siaga, wsapada dan bahaya yang akan mengirimkan pesan melaui SMS *gateway* menggunakan SIM800L. SIM800L akan mengirimkan pesan peringatan ke 3 nomor ponsel yang berbeda. Sistem juga teringtegrasi dengan media sosial Facebook menggunakan API (*Application Programming Interface*) dengan nama simonjir dan pesan peringatan banjir dikirim ke halaman Ant. Pada Telegram menggunakan *bot* simonjir yang akan mengirimkan pesan peringatan banjir ke *channel* Info Banjir. *Website* dapat digunakan untuk informasi pengguna dengan menampilkan grafik ketinggian air, info cuaca dan *gauge* ketinggian air, serta data rata-rata ketinggian air per jam yang dapat di cetak dalam bentuk *file* pdf.

Kata Kunci: Banjir, Ultrasonik HC-SR04, *Website*, API, *Bot*

Gambar 12. Tampilan Dokumen Sebelum Diperbaiki

ABSTRAK

Banjir merupakan bencana alam yang sering melanda Indonesia, terjadinya banjir di suatu daerah atau wilayah mengakibatkan kerugian harta benda. Pendeteksi ataupun peringatan dini banjir adalah langkah awal untuk memberikan informasi kepada masyarakat guna mengurangi kerugian ketika banjir melanda. Sistem pendeteksi dan monitoring banjir dengan antarmuka *website* merupakan alat yang dapat memberikan informasi ketinggian air untuk melihat potensi terjadinya banjir. Sensor ultrasonik HC-SR04 berfungsi sebagai alat untuk mengukur ketinggian air. Sistem yang dibuat juga dapat memberikan peringatan tentang status level ketinggian air pada saat siaga, waspada dan bahaya yang akan mengirimkan pesan melalui SMS *gateway* menggunakan SIM800L. SIM800L akan mengirimkan pesan peringatan ke 3 nomor ponsel yang berbeda. Sistem juga terintegrasi dengan media sosial Facebook menggunakan API (*Application Programming Interface*) dengan nama simonjir dan pesan peringatan banjir dikirim ke halaman Ant. Pada Telegram menggunakan *bot* simonjir yang akan mengirimkan pesan peringatan banjir ke *channel* Info Banjir. *Website* dapat digunakan untuk informasi pengguna dengan menampilkan grafik ketinggian air, info cuaca dan *gauge* ketinggian air, serta data rata-rata ketinggian air per jam yang dapat di cetak dalam bentuk *file* pdf.

Kata Kunci: Banjir, Ultrasonik HC-SR04, *Website*, API, *Bot*

Gambar 13. Tampilan Dokumen Sesudah Diperbaiki

Bagi pengguna yang telah menyimpan hasil perbaikan. Pengguna dapat melihat daftar lis dokumen yang telah disimpan pada halaman data tes. Gambar dari tampilan halaman data tes dapat dilihat pada Gambar 14.

Gambar 14. Tampilan Halaman Data Tes

5.2. Pengujian Jaro Winkler Distance

Dari hasil pengujian aplikasi terhadap 3 dokumen diperoleh hasil sebagai berikut :

1. Pengoreksian 92 halaman terdapat 118 jumlah penulisan salah dari 13924 jumlah kata, dengan waktu proses 8.96 detik. Dari 118 jumlah penulisan salah, jumlah saran perbaikan yang berhasil ditemukan dengan *Jaro Winkler Distance* sebanyak 31 penulisan, jumlah kata non-kamus yang terdeteksi sebanyak 82 penulisan dan jumlah penulisan yang tidak ditemukan saran perbaikan sebanyak 5 penulisan. Dari 5 jumlah penulisan salah, terdapat 3 kesalahan penulisan disebabkan perhitungan *Jaro Winkler Distance* tidak mampu mendeteksi kesalahan penulisan kata berdempet. Adapun 2 kesalahan dikarenakan kurang lengkapnya kamus.
2. Pengoreksian 60 halaman terdapat 120 jumlah penulisan salah dari 9323 jumlah kata, dengan waktu proses 8.10 detik. Dari 120 jumlah penulisan salah, jumlah saran perbaikan yang berhasil ditemukan dengan *Jaro Winkler Distance* sebanyak 42 penulisan, jumlah kata non-kamus yang terdeteksi sebanyak 51 penulisan dan jumlah penulisan yang tidak ditemukan saran perbaikan sebanyak 27 penulisan. Dari 27 jumlah penulisan salah, terdapat 6 kesalahan karena kurang akuratnya pemberian saran dari hasil perhitungan *Jaro Winkler Distance*, sedangkan 9 kesalahan penulisan disebabkan perhitungan *Jaro Winkler Distance* tidak mampu mendeteksi kesalahan penulisan kata berdempet. Adapun 12 kesalahan dikarenakan kurang lengkapnya kamus.
3. Pengoreksian 113 halaman terdapat 123 jumlah penulisan salah dari 14254 jumlah kata, dengan waktu proses 8.44 detik. Dari 123 jumlah penulisan salah, jumlah saran perbaikan yang berhasil ditemukan dengan *Jaro Winkler Distance* sebanyak 22 penulisan, jumlah kata non-kamus yang terdeteksi sebanyak 83 penulisan dan jumlah penulisan yang tidak ditemukan saran perbaikan sebanyak 18 penulisan. Dari 18 jumlah penulisan salah, terdapat 2 kesalahan karena kurang akuratnya pemberian saran dari hasil perhitungan *Jaro Winkler Distance*, sedangkan 8 kesalahan penulisan disebabkan perhitungan *Jaro Winkler Distance* tidak mampu mendeteksi kesalahan penulisan kata berdempet. Adapun 8 kesalahan dikarenakan kurang lengkapnya kamus.

5.3. Pembahasan

Pengujian metode *Jaro Winkler Distance* dengan menggunakan tiga sampel dokumen dengan total keseluruhan 37501 kata membutuhkan waktu proses 25.5 detik untuk menyelesaikan pengoreksian. Dari total jumlah keseluruhan kata dan waktu proses yang dibutuhkan maka rata-rata kecepatan aplikasi pengoreksi kesalahan penulisan Bahasa

Indonesia berbasis *web* dengan menggunakan *Jaro Winkler Distance* adalah:

$$\frac{37501}{25.5} = 1470.6 \text{ kata/detik}$$

Dari kecepatan rata-rata tersebut dapat mengoreksi 3 dokumen skripsi dengan total gabungan 265 halaman, maka tiap mengoreksi dokumen skripsi dengan 265 halaman / 3 = 88 halaman, membutuhkan waktu 25.5 detik / 3 = 8.5 detik.

Hasil pengujian dari dua sampel dokumen terdapat 95 penulisan salah yang berhasil ditemukan saran perbaikan dengan *Jaro Winkler Distance*, sedangkan total saran perbaikan yang tidak bisa diberikan dari hasil perhitungan *Jaro Winkler Distance* ada sebanyak 28 kata yang diantaranya 20 disebabkan oleh penulisan yang berdempet tanpa spasi. Adapun faktor lain yang menyebabkan aplikasi kurang dalam mengidentifikasi kesalahan penulisan adalah faktor kurang lengkapnya kamus yang ditemukan pada sebanyak 22 penulisan.

Dari total 95 penulisan yang berhasil ditemukan saran perbaikan dan 28 penulisan yang gagal ditemukan saran perbaikan, dapat ditarik nilai keakurasian dari implementasi *Jaro Winkler Distance* dalam aplikasi pengoreksi kesalahan Bahasa Indonesia berbasis web yaitu:

$$\frac{95}{95 + 28} x100\% = 77.23\%$$

Berdasarkan hasil perhitungan nilai yang didapat dari rata-rata kecepatan aplikasi dalam melakukan pengoreksian yaitu 1470.6 kata/ detik, di mana tiap dokumen dengan jumlah 88 halaman memerlukan waktu 8.5 detik untuk melakukan pengoreksian yang dapat dikategorikan cepat. Sedangkan hasil perhitungan untuk nilai persentase keakurasian yang didapat untuk memberikan saran perbaikan adalah 77.23% dimana hasil tersebut dapat dikategorikan cukup baik.

6. KESIMPULAN DAN SARAN

6.1. Kesimpulan

Berdasarkan pengujian dari hasil penerapan *Jaro Winkler Distance* dalam aplikasi pengoreksi kesalahan penulisan Bahasa Indonesia berbasis web dapat ditarik kesimpulan sebagai berikut:

1. Faktor-faktor penyebab aplikasi gagal dalam memberikan saran perbaikan yang akurat adalah kurang akurasi pada *Jaro Winkler Distance* itu sendiri dan juga tidak mampu dalam mengoreksi kata yang berdempetan. Adapun faktor lain yaitu kegagalan dikarenakan kurang lengkapnya kamus.
2. Untuk sekali mengoreksi aplikasi memerlukan 8.5 detik dalam mengoreksi dokumen skripsi yang terdiri dari 88 halaman.
3. Aplikasi berhasil mendapatkan total persentase keakurasian dalam memberikan saran perbaikan dengan menggunakan *Jaro Winkler Distance* sebesar 77.23%.

6.2. Saran

Adapun saran untuk perbaikan dan pengembangan dari penelitian ini adalah :

1. Membuat fitur admin dan *feedback* saran masukan agar daftar kata pada kamus bisa diperbaiki atau ditambah berdasarkan saran dari pengguna dengan persetujuan admin.
2. Menambahkan metode lain untuk memecah kalimat yang berdempet tanpa spasi.
3. Menambahkan berbagai jenis format dokumen pengoreksian.

DAFTAR PUSTAKA

[1] Waridah, E., & Saeful, A. (2013). *EYD dan Seputar Kebahasa-Indonesiaan.* Bandung: Ruang Kata.

[2] Rochmawati, Y., & Kusumaningrum, R. (2016). Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching untuk Identifikasi Kesalahan Pengetikan Teks. *Fakultas Sains dan Matematika, Universitas Diponegoro*, 125-134.

[3] Suryaningrum, K. M., & T, A. (2016). Pengkoreksian dan Suggestion Word pada Keyword Menggunakan Algoritma Jaro Winkler. *Jurnal Teknologi Informasi-Aiti*, 169-181.

[4] Kurniawati, A., Puspitodjati, S., & Rahman, S. (2014). Implementasi Algoritma Jaro-Winkler Distance untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia. *Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Gunadarma*.

[5] Adriyani, N. M., Santiyasa, I. W., & Muliantara, A. (2012). Implementasi Algoritma Levenshtein Distance dan Metode Empiris untuk Menampilkan Saran Perbaikan Kesalahan Pengetikan Dokumen Berbahasa Indonesia. *Fakultas Matematika Dan Ilmu Pengetahuan Alam, Universitas Udayana*.

[6] Feldman, R., & Sanger, J. (2007). *The Text Mining Handbook : Advanced Approaches.* New York: Cambridge University Press.

[7] Syaroni, M., & Munir, R. (2005). Pencocokan String Berdasarkan Kemiripan Ucapan (Phonetic String Matching) Dalam Bahasa Inggris. *Seminar Nasional Aplikasi Teknologi Informasi 2005*.

[8] Naumann, F., & Herschel, M. (2010). *An Introduction to Duplicate Detection.* Morgan & Claypool Publishers.

[9] Dreßler, K., Ngomo, N., & Cyrille, A. (2017). On the efficient execution of bounded Jaro-Winkler distances. *Semantic Web 8*, 185–196.

[10] Fadhillah , N., Azis, H., & Lantara, D. (2018). Validasi Pencarian Kata Kunci Menggunakan Algoritma Levenshtein Distance Berdasarkan Metode Approximate String Matching. *Prosiding Seminar Nasional Ilmu Komputer dan Teknologi Informasi*, 129-133.