Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.201854780
Vol. 5, No. 4, September 2018, hlm. 435-444 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 51/E/KPT/2017 e-ISSN: 2528-6579

ALGORITMA JARO-WINKLER *DISTANCE*: FITUR *AUTOCORRECT* DAN *SPELLING SUGGESTION* PADA PENULISAN NASKAH BAHASA INDONESIA DI BMS TV

Agung Prasetyo[1], Wiga Maulana Baihaqi[2], Iqbaluddin Syam Had[3]

[1,2,3]Teknik Informatika, STMIK Amikom Purwokerto
Email: [1]pras@amikompurwokerto.ac.id, [2]wiga@amikompurwokerto.ac.id, [3]deluser404@gmail.com

(Naskah masuk: 20 April 2018, diterima untuk diterbitkan: 26 Agustus 2018)

Abstrak

Autocorrect adalah suatu sistem yang dapat memeriksa dan memperbaiki kesalahan penulisan kata secara otomatis. Saat ini berbagai jenis smartphone dan aplikasi Microsoft Word terdapat fitur *autocorrect*. Sistem autocorrect dapat mengganti kata yang dianggap salah oleh sistem secara otomatis tanpa memberi tahu pengguna sehingga pengguna seringkali tidak sadar tulisannya berubah, edangkan kata penggantinya tidak selalu benar sesuai dengan yang dimaksud pengguna. Pengetahuan Microsoft Word pada fitur *autocorrect*-nya berbahasa Inggris sehingga tidak dapat diterapkan pada penulisan naskah berita di BMS TV. Setiap harinya News Director BMS TV memeriksa naskah yang akan diberitakan dimana termasuk diantaranya adalah pemeriksaan ejaan. Dengan fitur *autocorrect* dan *spelling suggestion* bahasa Indonesia diharapkan dapat membantu *News Director* BMS TV untuk memeriksa dan memperbaiki kesalahan penulisan kata secara otomatis serta memberi saran penulisan ejaan kata yang benar dalam bahasa Indonesia. Metode yang digunakan seperti algoritma Jaro-Winkler *Distance* untuk pengembangan perangkat lunak ini menggunakan dalam menghitung nilai jarak kedekatan antara dua teks. Sementara itu, *Extreme Programming* digunakan untuk pengembangan perangkat lunak. Penelitian ini menghasilkan sebuah sistem yang dapat membantu *News Director* MS TV dalam pemeriksaan kesalahan penulisan ejaan kata pada naskah bahasa Indonesia dan mempermudah *News Director* pusat dalam penghimpunan naskah dari berbagai kontributor BMS TV. Disimpulkan bahwa fitur *autocorrect* dan *spelling suggestion* digunakan untuk menengani kesalahan penulisan ejaan kata, dengan pengujian 60 kata yang terdiri dari berbagai skenario kesalahan penulisan kata fitur ini memperbaiki sepuluh kata secara otomatis dengan benar dan memunculkan saran ejaan kata pada 39 kata dengan tepat.

Kata kunci: *autocorrect, spelling suggestion*, naskah, bahasa indonesia, jaro-winkler *distance*, *stemming*

JARO-WINKLER DISTANCE ALGORITHM: AUTOCORRECT FEATURES DAN SPELLING SUGGESTION IN INDONESIAN TEXT WRITING IN BMS TV

Abstract

Autocorrect is a software system that automatically identifies and correct misspelled words. Nowadays autocorrect feature is often encountered in various devices dan applications, like on the smartphone keyboard dan Microsoft Word application. The autocorrect system instantly replaces the word that is considered wrong by the system automatically without notifying the user so that users are often not aware of writing changes while the replacement word is not always true in accordance with the intended user. The Autocorrect feature of Microsoft Word uses English so it can't be applied on writing news script in BMS TV. Every day News Director of BMS TV checks the script that would be reported where there is a spell checking included. By using bahasa in autocorrect dan spelling suggestion, it is expected to help News Director BMS TV to check dan fix the misspelled word automatically dan give suggestion for the right words spelling in bahasa. The development software method that is used is Extreme Programming dan Jaro-Winkler Distance algorithm. Jaro-Winkler is an algorithm that is applied to calculate the distance of proximity between two texts. The results of this study is a system that could help News Director BMS TV in identifying misspelled words on script in bahasa dan to make it easier for News Director center in collecting of manuscripts from various contributors of BMS TV. It can be concluded that the autocorrect dan spelling suggestion features can compound the misspelled words with a 60-word test consisting of various error scenarios. This feature can correct ten words automatically dan show correct spelling suggestion word on 39 words.

Keywords*: autocorrect, spelling suggestion, script, indonesian language, jaro-winkler distance, stemming.*

1. PENDAHULUAN

Autocorrect adalah suatu fitur yang dapat memeriksa dan memperbaiki kesalahan penulisan kata secara otomatis. Menurut (Gueddah, Yousfi dan Belkasmi, 2016), pemeriksaan ejaan terdiri dari perbandingan antara kata yang salah dengan daftar kata pada basis data dan menyarankan kata-kata yang mirip dengan kata yang salah dengan menghitung kemiripan jarak antara kata-kata tersebut.

Menurut Mustakim yang dikutip (Widianingsih, 2014) dalam unsur bahasa ragam tulis, informasi yang disampaikan secara tertulis harus jelas. Mustakim menambahkan, bahwa dalam bahasa ragam tulis unsur-unsur bahasa yang dipergunakan harus lengkap. Jika ada unsur-unsur itu tidak lengkap, maka ada kemungkinan informasi yang disampaikan tidak dapat dipahami secara tepat.

BMS TV (Banyumas Televisi) adalah stasiun televisi dengan konten seni dan budaya lokal eks-karesidenan Banyumas yang ada di Purwokerto, Kabupaten Banyumas, Jawa Tengah. BMS TV diluncurkan pada tanggal 19 Maret 2003 mengudara di kanal 49 UHF dan masih tetap eksis hingga saat ini. Jangkauan siarannya sampai saat ini sudah menjangkau wilayah Barlingmascakeb (Kabupaten Banjarnegara, Kabupaten Purbalingga, Kabupaten Banyumas, Kabupaten Cilacap dan Kebumen) serta beberapa wilayah di Kabupaten Wonosobo, kebumen, hingga sebagian wilayah Jawa Barat yaitu Ciamis, Banjar, Pangdanaran dan Tasikmalaya.

Program utama BMS TV yang bernuansa lokal diantaranya adalah *Inyong Bae Polisi* dengan logat *Ngapaknya* yang khas, Jejak Sang Petualang yang mengeksplorasi tempat wisata yang berada di wilayah Jawa Tengah. Selain itu ada program acara Klinthungan, Cemplang Cemplong, Banyumas Legend, serta Seputar Masbarlingcakeb.

Seputar Masbarlingcakeb adalah salah satu program acara yang menjadi sorotan. Seputar Masbarlingcakeb tayang empat kali dalam sehari untuk memberikan berita dari berbagai daerah di eks-Karesidenan Banyumas kepada masyarakat. Pemberitaan Seputar Masbarlingcakeb dikelola oleh *News Director*. *News Director* BMS TV mempunyai kontributor-kontributor yang tersebar di lima wilayah sekitar Barlingmascakeb yang bertugas untuk meliput berita di daerahnya. Hasil liputan ditulis menjadi sebuah naskah menggunakan aplikasi Microsoft Word kemudian dikirim beserta *video* dan *file* pendukung lainnya ke kantor pusat BMS TV melalui *email* yang selanjutnya akan diolah menjadi berita untuk ditayangkan di BMS TV. Menurut keterangan Ari Nugroho selaku penanggungjawab *News Director* pusat, pihaknya tidak menggunakan aplikasi khusus untuk mendeteksi dan menangani kesalahan pengejaan pada penulisan naskah berita. Pengetahuan aplikasi Microsoft Word pada fitur *autocorrectnya* berbahasa Inggris, sehingga tidak cocok digunakan untuk menulis naskah beritanya. Fitur tersebut mengganti kata yang dianggap salah oleh sistem secara otomatis dan seringkali perubahan tersebut tidak sesuai dengan maksud pengguna. Setiap harinya *News Director* pusat memeriksa naskah berita yang masuk dimana termasuk diantaranya adalah pemeriksaan penulisan ejaan kata.

Tidak sedikit jumlah naskah berita yang mengalami kesalahan ejaan kata saat proses penulisan berita. Kesalahan pengetikan mengakibatkan kata baku berubah menjadi kata tidak baku karena pengejaan kata yang tidak sesuai. Beberapa faktor yang menyebabkan terjadinya kesalahan pengetikan adalah letak huruf pada *keyboard* yang berdekatan, kesalahan karena slip pada tangan atau jari, kesalahan yang disebabkan oleh ketidaksengajaan (Rochmawati dan Kusumaningrum, 2016).

Salah satu fitur yang dapat digunakan untuk mendeteksi kesalahan dan memberikan sugesti kata yang benar adalah fitur *spelling corrector* atau *spelling checker* atau *spelling suggestion*. Fitur ini berfungsi sebagai pendeteksi kesalahan dan memberikan panduan bagi penggunanya dengan mendanai kata-kata yang tidak terdaftar dalam kamus suatu bahasa tertentu. Fitur ini juga disertai dengan sugesti kata yang berfungsi menyediakan rekomendasi kata-kata yang mendekati kata yang dimaksud (Mutammimah; Sujaini, Herry; Nyoto, 2017). Menurut (Gueddah, Yousfi dan Belkasmi, 2016) pemeriksaan ejaan terdiri dari perbandingan antara kata yang salah dengan daftar kata pada basis data dan menyarankan kata-kata yang mirip dengan kata yang salah dengan menghitung kemiripan jarak antara kata-kata tersebut.

Dewasa ini fitu *autocorrect* memang sering ditemui pada berbagai perangkat dan aplikasi, misalkan pada papan ketik smartphone dan aplikasi misalkan sebut saja Microsoft Word. Fitur autocorrect tersebut langsung mengganti kata yang dianggap salah oleh sistem secara otomatis tanpa memberi tahu pengguna sehingga seringkali membuat pengguna tidak sadar tulisannya berubah. Menurut (Madison, 2011), fitur autocorrect bisa menjadi berkah, juga sering menjadi kutukan. Fitur tersebut sering mengubah kata-kata tanpa alasan, sehingga jika terburu-buru menekan tombol "kirim", hal ini dapat menyebabkan hasil yang lucu, membingungkan, dan yang lainnya.

Penelitian tentang *autocorrect* ataupun *spelling suggestion* telah dilakukan oleh beberapa peneliti, diantaranya adalah (Suryaningrum dan T, 2016) yang mengimplementasikan algoritma Jaro-Winkler ke dalam fitur yang mampu memberikan koreksi dan suggestion word jika terjadi kesalahan pengetikan pada keyword bahan pustaka atau pencarian data buku.

Penelitian lainnya dilakukan oleh (Hancox dan Polatidis, 2012) dengan "*Infobot*" yang merupakan sistem jawaban pertanyaan dari bahasa alami secara

otomatis berskala kecil. Fitur utamanya adalah ekstraksi makna dari kueri pengguna tanpa menggunakan representasi sintaksis atau semantik. Dua pendekatan untuk mengidentifikasi makna yang dimaksud pengguna diselidiki: sistem *keyword-based* dan *Jaro-based string similarity algorithms*. *Correct/spelling error* memeriksa penulisan ejaan kata dari *input*-an pengguna untuk mengetahui makna yang sebenarnya. Tingkat kebenaran terbaik yang dapat dihasilkan *Infobot* adalah 78,57%.

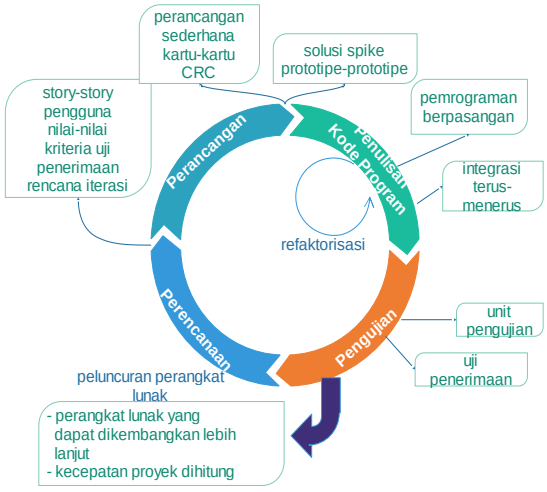
(Rochmawati dan Kusumaningrum, 2016) mengujicobakan 50 kata yang salah sebagai masukan untuk proses identifikasi kesalahan pengetikan dan saran perbaikan berdasarkan algoritma *Hamming Distance*, *Levenshtein Distance*, *Damerau Levenshtein Distance* dan *Jaro-Winkler Distance*. Hasilnya menunjukkan bahwa algoritma *Jaro-Winkler Distance* memiliki nilai tertinggi dibandingkan ketiga algoritma yang lain dengan nilai MAP 0,87 yang terbagi ke dalam empat jenis kesalahan penulisan yaitu jenis kesalahan penghapusan huruf 0,92, jenis kesalahan penambahan huruf 0,90, jenis kesalahan penggantian huruf 0,70 dan jenis kesalahan penukaran huruf 0,95.

Berdasarkan permasalahan tersebut penulis membuat sebuah fitur *autocorrect* dan *spelling suggestion* pada penulisan naskah di BMS TV yang diharapkan dapat memeriksa dan memperbaiki kesalahan penulisan kata secara otomatis serta memberi saran penulisan ejaan kata yang benar dalam bahasa Indonesia. Algoritma yang digunakan adalah *Jaro-Winkler Distance*. *Jaro-Winkler Distance* adalah algoritma untuk menghitung nilai jarak kedekatan antara dua teks. Menurut (Winkler, 1999), *Jaro-Winkler Distance* mempunyai 3 komponen dasar: (1) menghitung panjang *string* atau kata, (2) mencari nomor huruf pada kedua kata, dan (3) mencari transposisi. Semakin tinggi jarak *Jaro-Winkler Distance* antara dua teks berarti semakin ada kemiripan.

2. METODE PENELITIAN

Metode pengembangan sistem yang digunakan pada penelitian ini adalah metode *Extreme Programming* (XP) yang merupakan salah satu metode yang tergolong kedalam *Agile Methodology* yang menggunakan permodelan *Unified Modeling Language* (UML) atau disebut juga sebagai permodelan visual. Gambaran proses *Extreme Programming* dapat dilihat pada Gambar 1.

Penulis menggunakan metode pengembangan sistem *Extreme Programming* karena XP menyederhanakan berbagai tahapan dalam proses pengembangan tersebut sehingga menjadi lebih adaptif dan fleksibel (Pressman, 2010). Tahap perencanaan, perancangan, penulisan kode program dan pengujian dilakukan pada setiap iterasi. *Client* turut aktif pada setiap proses pengembangan sistem dan apabila ada perubahan konsep ataupun desain dari *client* dapat segera diperbaiki tanpa harus mengulang dari awal tahap perencanaan sehingga waktu pengembangan sistem menjadi lebih cepat.



Gambar 1. Proses extreme programming (Pressman dan Maxim, 2015)

2.1. Perencanaan

Tahap *planning* dilakukan dengan membuat sebuah "*user stories*" yang menjelaskan *output*, fitur dan fungsional dari sistem yang dibuat. Pada tahapan perencanaan akan dilakukan analisis terhadap nilai atau *score* yang dihasilkan algoritma Jaro-Winkler *Distance* dari perhitungan jarak kedekatan antar kata untuk menentukan nilai minimal *autocorrect* dijalankan ataupun memunculkan kata *spelling suggestion* pada sistem. Proses ini bertujuan untuk menentukan kebutuhan sistem, hal ini sangat penting mengingat sistem tidak akan berjalan jika apa yang dibutuhkan sistem tersebut tidak terpenuhi.

2.2. Perancangan

Perancangan pada *extreme programming* dilakukan dengan mengikuti prinsip "sederhana". Untuk desain yang sulit, *extreme programming* akan menggunakan *spike solution* dimana pembuatan desain dibuat langsung ke tujuan. XP juga mendukung adanya refactoring dimana *software system* dapat diubah sedemikian rupa dengan cara mengubah struktur kode dan menyederhanakan kode.

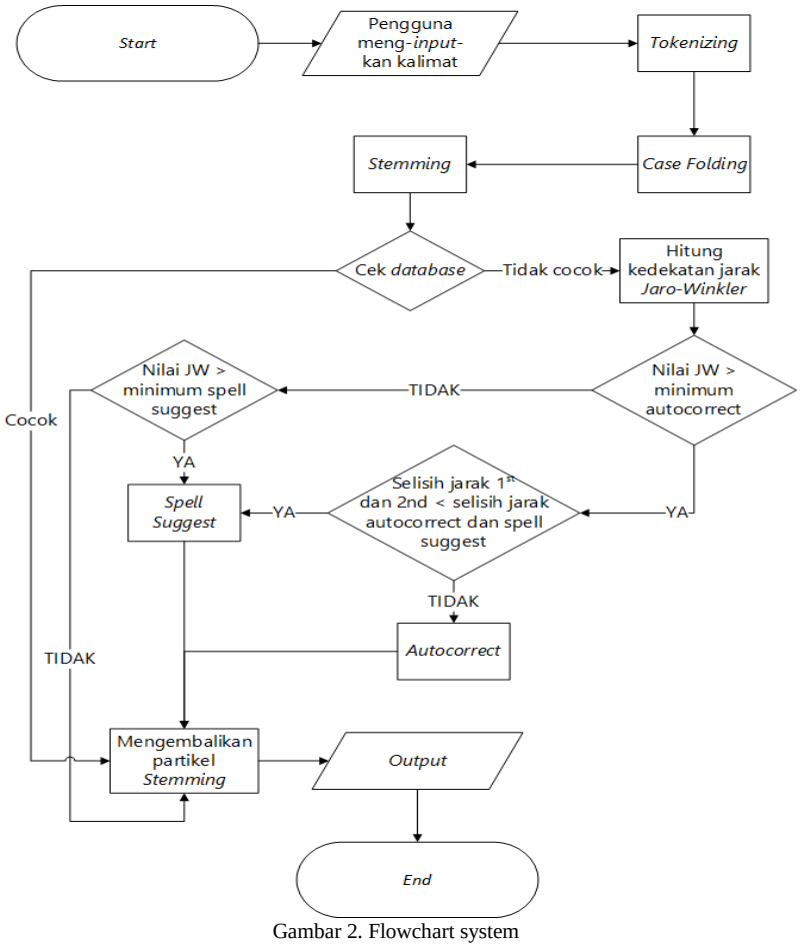
Perancangan sistem digambarkan ke dalam empat bentuk diagram pemodelan, yakni *Use Case Diagram*, *Activity Diagram*, *Sequence Diagram*, dan *Class Diagram*. Pada tahapan ini juga dibuat desain antarmuka sistem.

2.3. Penulisan Kode Program

Pada tahapan ini sistem mengolah kata-kata atau kalimat yang di-*input*-kan pengguna untuk diidentifikasi pengejaannya. Sebelum proses pengukuran jarak, kata-kata atau kalimat pengguna diolah pada tahap *preprocessing* terlebih dahulu

sampai menjadi data yang siap diproses. Tahapan pemrosesan fitur *autocorrect* dan *spelling suggestion* terdapat pada Gambar 2.

Secara umum pemrosesan pada sistem *autocorrect* dan *spelling suggestion* dibagi menjadi dua tahap. Yakni tahap *Preprocessing* (*Tokenizing*, *Case Folding*, dan *Stemming*) dan tahap pengukuran jarak kedekatan kata yang memiliki salah ejaan menggunakan algoritma Jaro-Winkler *Distance* untuk mencari kata yang mirip.



Gambar 2. Flowchart system

2.3.1. *Preprocessing*

a. *Tokenizing*

Mendeteksi token dan batas kalimat adalah langkah preprocessing penting dalam aplikasi pemrosesan bahasa alami karena sebagian besar beroperasi baik pada tingkat kata (misalnya, silabus, analisis morfologi) atau kalimat (misalnya pemberian tag bagian-pidato, parsing, terjemahan mesin). Beberapa penelitian tentang pemrosesan bahasa menyertakan proses *tokenizing* pada tahap preprocessing, (Omar, 2018) melakukan tonizing pada tahap preprocessing untuk memperoleh token bahasa arab, (Noaman, Sarhan dan Rashwan, 2018) menyajikan model bahasa jaringan saraf berdasarkan tokenisasi kata-kata menjadi tiga bagian: awalan, batang, dan akhiran. Model yang diusulkan diuji dengan dataset pengenal ucapan AMI bahasa Inggris dan melebihi model n-gram awal.

Tugas tokenisasi adalah membagi kata dari teks menjadi set morfem yang berurutan (Aliwy, 2012). Tokenisasi adalah langkah yang diperlukan dan tidak sepele dalam pemrosesan bahasa alami (Bird, Klein dan Loper, 2009). Hal ini terkait erat dengan analisis morfologi tetapi biasanya, ini telah dilihat sebagai proses yang independen (Chanod dan Tapanainen, 1996). Sistem memecah kalimat yang di-input-kan pengguna menjadi token-token atau bagian-bagian tertentu. Misalkan tokenisasi dari kalimat "Tidur larut malam" menghasilkan tiga token, yakni "Tidur", "larut", dan "malam" (Suryaningrum dan T, 2016). Token-token tersebut yang akan diproses pada tahap berikutnya.

b. *Case Folding*

Pada tahap ini hasil dari tokenisasi diubah menjadi huruf kecil atau non-kapital semua. Hal ini karena proses Jaro-Winkler *Distance* bersifat *case sensitive* sehingga besar kecilnya huruf dapat mempengaruhi hasil perhitungan jarak kedekatannya. Contoh proses *case folding* dari token-token yang sudah didapatkan dari poses

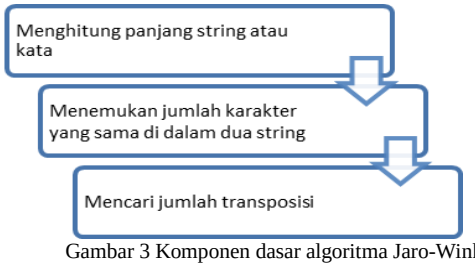
tokenizng, yaitu kata "Tidur" diubah menjadi "tidur", sedangkan kata "larut" dan "malam" tidak ada perubahan karena sudah dalam bentuk huruf kecil.

c. *Stemming*

Proses *stemming* hampir dilakukan atau digunakan pada penelitian dengan data berupa teks, seperti yang dilakukan oleh (Rahardian dkk., 2017) dalam penelitiannya yang bertujuan untuk mengklaster judul-judul majalah elektronik. Masing-masing token yang terbentuk diproses kembali untuk membentuk kata dasar. Pada tahapan ini semua jenis imbuhan kata, tdana baca "/", "//", dan tag html dihapus selama proses penghitungan Jaro-Winkler Distance berlangsung. Tujuan mendapatkan root word adalah untuk mencegah kesalahan perhitungan ketika melakukan ekstraksi fitur sintaksis (Prayogo, Mubarok dan Adiwijaya, 2018). Misalnya ketika pasangan kalimat memiliki kata root yang sama untuk semua kata-kata mereka tetapi hanya berbeda pada affix, suffix, infix atau circumfix, jika kita tidak melakukan stemming, skor jarak dari fitur sintaksis tidak akan 0, tetapi jika kita melakukan Dengan *stemming* kita akan mendapatkan skor jarak 0. Jika kita mendapatkan skor 0 artinya arti dari kalimat pasangan adalah sama, selain itu artinya pasangan memiliki arti yang berbeda apakah itu sedikit berbeda atau secara eksplisit berbeda.

2.3.2. Perhitungan Jaro-Winkler *Distance*

Setelah data siap diproses, setiap token dicek pengejaannya dengan daftar kata pada *database*. Jika ditemukan kecocokkan maka pengejaan kata tersebut dianggap benar dan tidak melalui tahap penghitungan jarak. Namun jika tidak ditemukan kecocokkan pada database maka sistem akan menghitung jarak kedekatannya dengan daftar kata yang diindikasi mempunyai kemiripan berdasarkan jumlah karakter atau huruf token tersebut. Jaro-Winkler *Distance* mempunyai 3 komponen dasar.



Gambar 3 Komponen dasar algoritma Jaro-Winkler

Pada algoritma Jaro-Winkler (Gambar 3), jarak antara dua kata s1 dan s2 dihitung menggunakan rumus pada Persamaan 1.

$$Jaro(s1,s2) = \frac{1}{3}\left(\frac{m}{|s1|}+\frac{m}{|s2|}+\frac{m-t}{m}\right) \quad (1)$$

dimana s1 dan s2 adalah dua teks atau kata yang dihitung jarak atau kedekatannya. m adalah jumlah karakter yang cocok antara s1 dengan s2, t adalah jumlah transposisi. Jumlah karakter pencocokan yang urutan berbeda dibagi dua menjadi nilai transposisi. Dua karakter dari s1 dan s2 dikatakan cocok hanya jika tidak lebih jauh dari $\left(\frac{\max(|s1|,|s2|)}{2}\right) < -1$ (Banerjee, Bhattacharyya dan Sanyal, 2011). Nilai Jaro yang tinggi menunjukkan kemiripan yang lebih besar antara kedua kata. Nilai Jaro 0 menunjukkan bahwa kata tidak sama dan nilai 1 menunjukkan bahwa keduanya sama (Agarwal, 2013).

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 s1 dan s2 yang diperbandingkan, maka Jaro-Winkler *distance*-nya (dw) adalah:

$$dw(s1,s2) = Jaro(s1,s2) + \left(L * p\left(1 - Jaro(s1,s2)\right)\right) \quad (2)$$

Dimana jaro merupakan Jaro *Distance* untuk string s1 dan s2, t merupakan panjang prefiks umum di awal *string* yang nilai maksimalnya adala 4 karakter, sedangkan p merupakan konstanta *scaling factor* dengan nilai standar menurut Winkler adalah 0,1 *(Pdanya dan Virparia, 2011)*.

Misalkan pengguna berniat menulis "tidur" tetapi terjadi kesalahan sehingga tertulis "tidyr". Tidak ditemukan kata "tidyr" pada daftar kata di database sehingga kata "tidyr" akan dihitung jarak kedekatannya dengan semua kata yang jumlah karakternya empat sampai enam karakter yang terdaftar di database menggunakan algoritma Jaro-Winkler Distance. Contoh: "tiban", "tidak", "tidur", "tifa", dan "tifus".

Perhitungan jarak kedekatan kata "tidyr" (s1) dengan "tiban" (s2) menggunakan algoritma Jaro-Winkler *Distance*.

$$Jaro(s1,s2) = \frac{1}{3}\left(\frac{m}{|s1|}+\frac{m}{|s2|}+\frac{m-t}{m}\right)$$

$String1\ (s1) = tidyr$

$String2\ (s2) = tiban$

$Jumlah\ karakterstring1\ |s1| = 5$

$Jumlah\ karakterstring2\ |s2| = 5$

$Jumlah\ karakter\ s1\ cocok\ dengan\ s2\ (m) = 2$

$Jumlah\ transposisi\ (t) = 0$

$$Jaro(s1,s2) = \frac{1}{3}\left(\frac{2}{5}+\frac{2}{5}+\frac{2-0}{2}\right)$$

$Jaro(s1,s2) = 0,6$

$Jaro-Winkler(s1,s2) = Jaro(s1,s2) + (L * p(1Jaro(s1,s2)))$

$Kesamaan\ awalan\ (L) = 2$

$Faktor\ penskalaran\ (p) = 0.1$

$Jaro-Winkler(s1,s2) = 0.6 + (2*0.1(1-0.6))$

$Jaro-Winkler(s1,s2) = 0.68$

$String1\ (s1) = tidyr$
$String2\ (s2) = tiban$
$Jumlah\ karakterstring1\ |s1| = 5$
$Jumlah\ karakterstring2\ |s2| = 5$
$Jumlah\ karakter\ s1\ cocok\ dengan\ s2\ (m) = 2$
$Jumlah\ transposisi\ (t) = 0$

Sehingga dapat diketahui hasil perhitungannya pada Tabel 1.

Tabel 1. Rancangan Analisis Komputasi

no	*string 1*	*string 2*	*nilai jw*
1	*tidyr*	*tiban*	*0.68*
2	*tidyr*	*tidak*	*0.813*
3	*tidyr*	*tidur*	*0.906*
4	*tidyr*	*tifa*	*0.706*
5	*tidyr*	*tifus*	*0.68*

Dari hasil perhitungan di atas, kata "tidyr" mempunyai kemiripan paling dekat dengan kata "tidur" yang memiliki nilai kedekatan 0,906. Maka kata "tidur" yang akan menggantikan kata "tidyr" pada sistem *autocorrect* dan mengembalikan partikel-partikel yang dihapus pada saat proses *stemming* jika ada.

2.4. Pengujian

Testing dilakukan dengan pengujian kode pada *unit testing*. Hasil pengkoreksian atau *autocorrect* dan pemberian saran ejaan kata diujicoba dengan beberapa kesalahan pengetikan.

3. HASIL DAN PEMBAHASAN

Metode pengembangan sistem yang digunakan pada penelitian ini adalah model *Extreme Programming* yang terdiri dari empat tahapan yaitu Perencanaan (*Planning*), Perancangan (*Design*), Penulisan Kode Program (*Coding*), Pengujian (*Testing*). Keempat tahapan tersebut dilakukan secara berurutan dalam prakteknya. Berikut ini merupakan hasil penerapan algoritma Jaro-Winkler yang telah dilakukan dalam pembuatan fitur *autocorrect* dan *spelling suggestion* pada penulisan naskah bahasa Indonesia di BMS TV:

3.1 Penulisan Kode Program (*Coding*)

Kode program utama sistem ini terdapat di halaman tulis naskah dimana fitur *autocorrect* dan *spelling suggestion* bekerja. Secara umum proses *autocorrect* terdiri dari proses *Preprocessing* (*Tokenizing*, *Case Folding*, *Stemming*) dan perhitungan algoritma Jaro-Winkler *Distance*.

3.1.1. *Preprocessing*

a. *Tokenizing*

Setiap kalimat yang dituliskan pengguna dipecah menjadi token-token dalam variabel *array* menggunakan *function explode* pada PHP. Proses *tokenizing* memecah kata berdasarkan spasi.

```
$arr = explode(" ", $mrgtag);
```

Gambar 4. Script tokenizing

Variabel *$arr* (Gambar 4) digunakan untuk menyimpan token-token yang dihasilkan dan *$mrgtag* adalah variabel yang menyimpan kalimat yang di-*input*-kan pengguna.

b. *Case Folding*

Token atau kata yang akan diproses diubah menjadi huruf kecil atau non-kapital menggunakan *function strtolower* pada PHP.

```
$arr[$i] = stemming($i, strtolower($arr[$i]));
```

Gambar 5. Script case folding

Proses *Case Folding* (Gambar 5) dilakukan bersamaan pada saat pengiriman kata untuk diolah pada proses pembentukkan kata dasar atau pemanggilan *function Stemming.*

c. Stemming

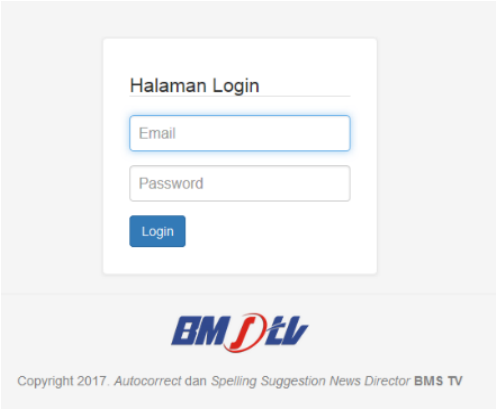
Function yang digunakan adalah *substr* dan *strlen* untuk operasi *string* atau huruf. Awalan ataupun akhiran yang dipotong disimpan dalam variabel *session* untuk dikembalikan setelah pemrosesan selesai atau tahap *output*.

3.1.2. Perhitungan algoritma Jaro-Winkler *Distance*

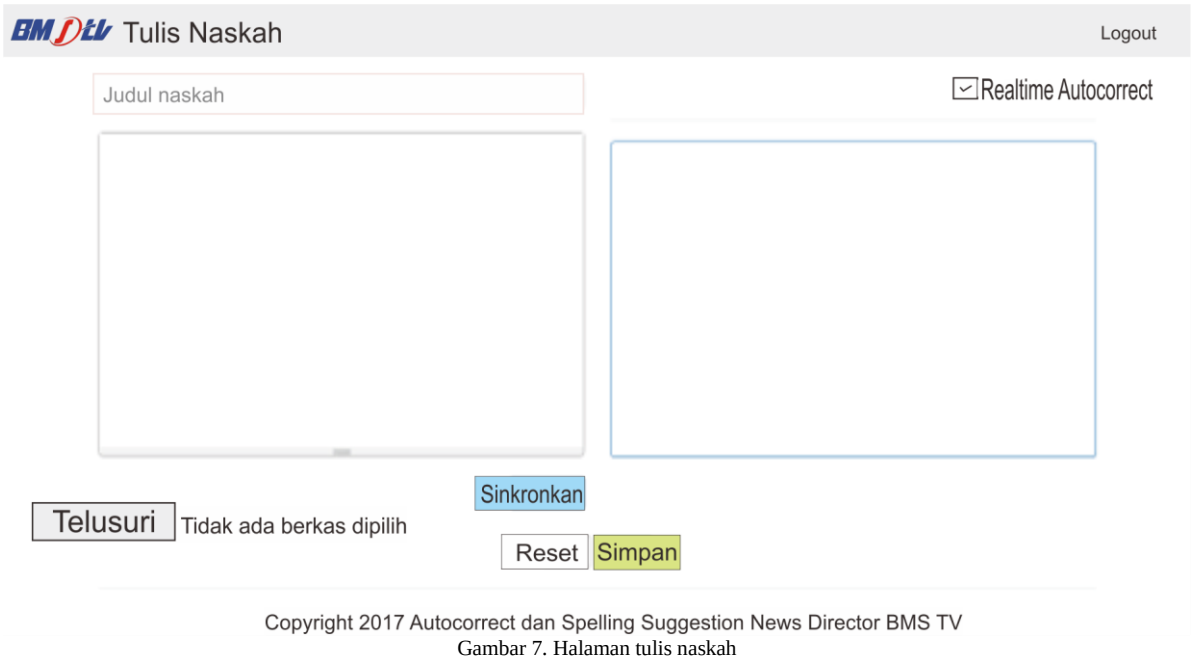
Perhitungan jarak kedekatan antara dua *string* seperti yang telah dituliskan pada Tabel 1 diimplementasikan ke dalam bahasa pemrograman PHP untuk menemukan kata yang paling mirip, dengan mencari nilai yang jarak kedekatannya paling tinggi antara sumber referensi kata yang benar dalam bahasa Indonesia dengan token atau kata yang dituliskan pengguna.

3.1.3. Implementasi Aplikasi

Antarmuka (*user interface*) sistem *autocorrect* dan *spelling suggestion* menggunakan *library* CSS *Bootstrap* yang merupakan *front-end framework* untuk mempercepat dan mempermudah pengembangan tampilan *website*. Berikut beberapa tampilan antarmuka (*interface*) yang terdapat dalam sistem *autocorrect* dan *spelling suggestion* dan telah disetujui *client* atau tim redaksi BMS TV (Gambar 6 dan Gambar 7).



Gambar 6. Tampilan halaman login



Gambar 7. Halaman tulis naskah

Fitur *autocorrect* dan *spelling suggestion* bekerja di halaman ini. Pengguna menuliskan naskahnya pada *form* yang tersedia, *form* menggunakan *plugin Summernote* dan hasil dari proses *autocorrect* dan *spelling suggestion* akan tampil di panel sebelah *form* isi naskah. Kata yang telah dikoreksi ditampilkan dengan huruf tebal, kata yang dianggap salah dan terdapat saran ejaan ditampilkan dengan huruf berwarna biru yang dapat memunculkan beberapa saran ejaan kata, kata yang dianggap salah namun tidak ada saran ejaan ditampilkan dengan huruf miring, serta kata yang dianggap sudah benar ditampilkan dengan huruf normal.

Realtime autocorrect berarti fitur *autocorrect* dan *spelling suggestion* bekerja setiap pengguna menuliskan satu kata, sedangkan pilihan lain fitur *autocorrect* dan *spelling suggestion* bekerja setelah pengguna selesai menuliskan naskah kemudian menekan tombol Sinkronasi. Pada halaman ini juga terdapat isian judul naskah, lampiran naskah, serta tombol Simpan untuk mengirim atau menyimpan naskah.

3.2. Pengujian

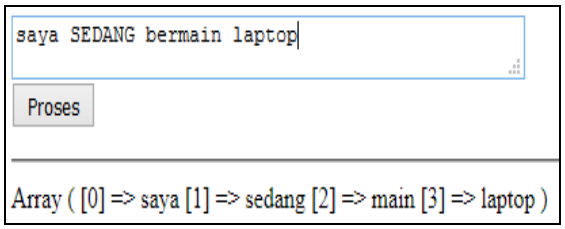
Pengujian pada tahap ini dilakukan menggunakan pengujian *unit testing* dan *acceptance testing*. Tahapan *unit testing* dilakukan untuk mengetahui apakah sistem sudah berjalan sesuai dengan yang diinginkan atau tidak dengan menguji setiap *function* pada sistem. *Acceptance testing* dilakukan untuk menentukan apakah sistem yang dibangun telah memenuhi kriteria penerimaan serta menentukan apakah sistem dapat diterima atau tidak.

3.2.1 Unit *Testing*

Unit testing dilakukan dengan pengujian terhadap setiap *Class* pembangun fitur *autocorrect* dan *spelling suggestion*. Hasil dari pengujian *unit testing* dirangkum dalam pengujian sebagai berikut:

a. Class Preprocessing()

Class Preprocessing mengolah data *input*-an pengguna pada penulisan naskah sebelum diproses menggunakan algoritma Jaro-Winkler *Distance*. Pemrosesan pada *class* ini diantaranya adalah *tokenizing*, *case folding*, dan *stemming* yang terdiri dari beberapa method. Pengujian dilakukan dengan *input*-an berupa kalimat yang *output*-nya menjadi variabel *array* per kata dari tahap *tokenizing* kemudian melewati tahap *case folding* dan *stemming*. Hasil pengujian dapat dilihat pada Gambar 8.



Gambar 8. Pengujian class Preprocessing

b. Class JaroWinkler()

Class JaroWinkler mengolah data *input*-an pengguna setelah diolah *class Preprocessing* (Gambar 9). Pada tahap ini token yang terbentuk dari *class Preprocessing* diperiksa dan dihitung kedekatan jaraknya dengan daftar kata di *database*. Pengujian dilakukan dengan *input*-an satu kata yang salah eja, kemudian dihitung kedekatan jarak antara kata yang salah eja dengan daftar kata di *database* dan ditampilkan secara urut dari *score* paling tinggi sampai socre yang paling rendah. Score yang tinggi artinya memiliki kedekatan atau kemiripan yang dapat dijadikan pengganti kata yang salah eja.

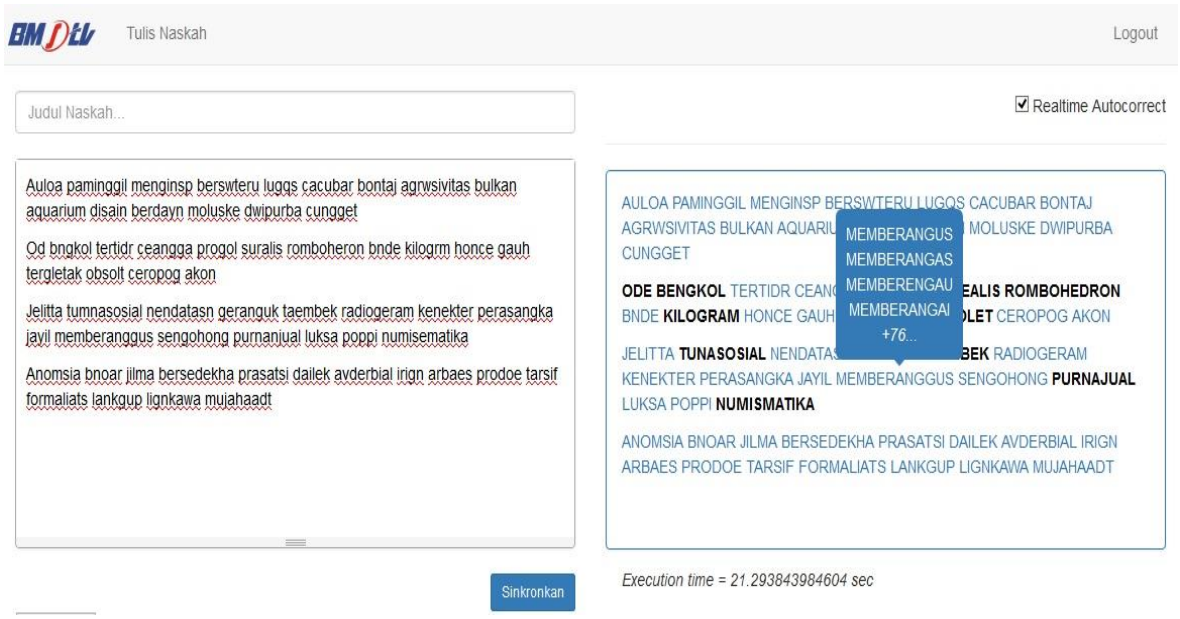
```
tidyr > tidur = 0.90666666666667
tidyr > tiri = 0.82666666666667
tidyr > tiru = 0.82666666666667
tidyr > tiroid = 0.82444444444444
tidyr > tidak = 0.81333333333333
tidyr > tadir = 0.805
tidyr > timur = 0.78666666666667
tidyr > tiner = 0.78666666666667
tidyr > tilde = 0.78666666666667
tidyr > tikar = 0.78666666666667
tidyr > tiara = 0.78666666666667
tidyr > tipar = 0.78666666666667
tidyr > tiram = 0.78666666666667
tidyr > tirau = 0.78666666666667
tidyr > tiris = 0.78666666666667
tidyr > tirta = 0.78666666666667
tidyr > tiras = 0.78666666666667
tidyr > tiran = 0.78666666666667
tidyr > tirai = 0.78666666666667
tidyr > tiada = 0.78666666666667
tidyr > tirah = 0.78666666666667
tidyr > tirus = 0.78666666666667
tidyr > utarid = 0.78055555555556
```

Gambar 9. Pengujian class JaroWinkler

c. Class Naskah()

Class Naskah adalah *class* utama pada fitur *autocorrect* dan *spelling suggestion*, *class* Naskah mengirim dan menerima pemrosesan *autocorrect* serta pengambilan dan penyimpanan naskah berita ke *database*. Pengujian *class* ini dilakukan dengan pengujian fitur *autocorrect* dan *spelling suggestion* dalam menangani kesalahan penulisan ejaan pada 60 kata yang dipilih secara acak. Skenario pengujian fitur *autocorrect* dan spelling suggestion dilakukan dengan mengujicobakan kata yang mengalami kesalahan penulisan huruf, pengurangan huruf, penambahan huruf dan transposisi huruf. Hasil pengujian dapat dilihat pada Gambar 10.

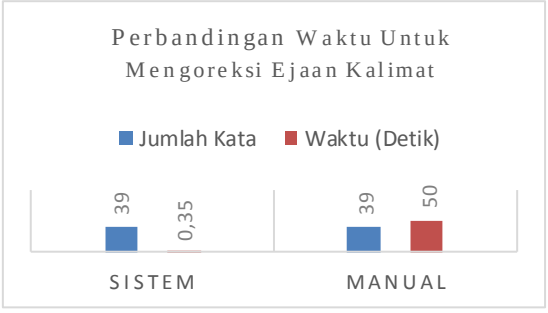
Dari hasil pengujian, fitur dapat memperbaiki sepuluh kesalahan penulisan ejaan kata secara otomatis dengan benar dan memberi saran penulisan ejaan kata pada 39 kata dengan tepat di empat saran kata teratas dalam waktu rata-rata sekitar 0,35 detik perkata.



Gambar 10. Hasil pengujian sistem

Berdasarkan hasil pengujian Class Naskah(), editor berita BMS TV sangat terbantu dengan adanya sistem *autocorrect* dan suggestion yang telah dibangun, dibandingkan dengan tidak menggunakan sistem autocorrect dan suggestion atau dengan cara manual membutuhkan waktu dan tenaga yang lebih untuk mengoreksi naskah yang ada pada Gambar 10. Berikut perbandingan waktu yang dibutuhkan dalam mengoreksi ejaan naskah pada Gambar 10 dapat dilihat pada Gambar 11.

Berdasarkan Gambar 11, jarak waktu yang diperlukan untuk mengoreksi ejaan kata pada Gambar 10 terbilang sangat jauh, sehingga dengan adanya sistem autocorrect dan suggestion ini sangat membantu editor dalam mengoreksi dan mengedit berita yang dituliskan oleh pencari berita di BMS TV.



Gambar 11 Perbandingan Waktu untuk Mengoreksi Ejaan Kata

3.2.2. *Acceptance Testing* (pengujian penerimaan)

Acceptance testing dilakukan dengan metode kuisioner, perhitungan hasil kuisioner dilakukan dengan menggunakan skala likert. Pengujian dilakukan dengan menjalankan sistem autocorrect dan spelling suggestion kemudian diberikan lembaran yang berisi enam pertanyaan kepada tim News Director BMS TV regional Banyumas yang ada di Purwokerto. News Director BMS TV di Purwokerto merupakan responden yang berkompeten untuk menjawab kuisioner guna menentukan user acceptance penelitian ini karena pada News Director BMS TV di Purwokerto terdapat News Director pusat atau admin dan kontri sehingga dapat mewakili kontributor dari daerah lain. Selain itu News Director BMS TV di Purwokerto juga terlibat dalam proses pengembangan perangkat lunak dan memahami bagaimana penggunaan sistem autocorrect dan spelling suggestion. Poin-poin pada pertanyaan tersebut berisi tentang semua fitur pada sistem autocorrect dan spelling suggestion apakah sesuai dengan permasalahan yang ada.

Setelah selesai menghitung indeks tiap-tiap aspek, diperlukan kriteria interprestasi skor berdasarkan interval (jarak). Rumus interval dalam bentuk presentase sebagai berikut:

Interval = 100 / 5 (bobot nilai tertinggi) = 20

Berdasarkan rumus tersebut, dihasilkan interval sebesar 20 (ini adalah jarak terendah 0% hingga tertinggi 100%). Berikut interprestasi skor berdasarkan interval:

Tabel 2. Interprestasi skor berdasarkan interval

Indeks	*Kategori*
80% - 100%	*Sangat Setuju*
60% - 79%	*Setuju*
40% - 59%	*Cukup Setuju*
21% - 39%	*Tidak Setuju*
0% - 19%	*Sangat Tidak Setuju*

Tabel 3. Hasil akhir pengujian

Aspek ke	*Indeks*	*Kategori*
1	*80%*	*Sangat Setuju*
2	*70%*	*Setuju*
3	*60%*	*Setuju*
4	*70%*	*Setuju*
5	*60%*	*Setuju*
6	*60%*	*Setuju*

Dari tabel tersebut dapat dihasilkan rata-rata indeks persetujuan sebagai berikut:

(80% + 70% + 60% + 70% + 60% + 60%) / 6 = 66,66%

Rata-rata indeks persetujuan dari responden sebesar 66,66%, jadi dapat disimpulkan hasil tersebut termasuk dalam kategori Setuju, yang berarti sistem autocorrect dan spelling suggestion dapat membantu News Director BMS TV dalam pemeriksaan kesalahan pengetikan ejaan kata pada naskah berita bahasa Indonesia dan mempermudah News Director pusat dalam penghimpunan naskah dari berbagai kontributor BMS TV.

4. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan di BMS TV tentang penerapan algoritma Jaro-Winkler *Distance* untuk fitur *autocorrect* dan *spelling suggestion* dapat disimpulkan bahwa fitur yang telah dibuat dapat menangani kesalahan penulisan ejaan kata pada penulisan naskah bahasa Indonesia. Dari hasil pengujian terhadap 60 kata yang terdiri dari berbagai kesalahan penulisan ejaan, fitur autocorrect dan spelling suggestion dapat menangani kesalahan penulisan ejaan pada 49 kata dengan baik. Sedangkan pada kata lainnya terjadi kesalahan dalam menampilkan saran ejaan.

Penelitian lebih lanjut digunakan untuk pengembangan sistem *autocorrect* dan *spelling suggestion*, yaitu manambahkan pembobotan kata pada daftar kata di database atau membatasi korpus dengan daftar kata yang sering digunakan untuk meningkatkan akurasi fitur dalam menangani keslahan penulisan ejaan kata. Penyempurnaan pada algoritma Poerter Stemmer dengan memperhatikan pola kata dasar yang terdapat di database agar kata dasar yang terbentuk dari proses *stemming* lebih tepat**.** Rencana penelitian selanjutnya adalah mencoba menguji metode yang lain pada sistem *autocorrect* dan *spelling suggestion*, serta mengembangkan sistem menjadi *plugin* yang dapat digunakan pada produk office dan browser.

DAFTAR PUSTAKA

AGARWAL, M., 2013. Text Steganographic Approaches: A Comparison. International Journal of Network Security & Its Applications (IJNSA), 5(1), pp.91–106.

ALIWY, A.H., 2012. Tokenization as Preprocessing for Arabic Tagging System. International Journal of Information and Education Technology, [online] 2(4), pp.348–353. Tersedia di: <http://www.ijiet.org/show-32-118-1.html> [Diakses 10 Juli 2018]

BANERJEE, I., BHATTACHARYYA, S. dan SANYAL, G., 2011. Novel Text Steganography through Special Code Generation. International conference on Systemics, Cybernetics and Informatics, pp.298–303.

BIRD, S., KLEIN, E. dan LOPER, E., 2009. Natural Language Processing with Python. Edition, F ed. [online] Sebastopol: O'Reilly Media. Tersedia di: <http://www.amazon.com/dp/0596516495> [Diakses 25 januari 2018]

CHANOD, J.-P. dan TAPANAINEN, P., 1996. A Non-deterministic Tokeniser for Finite-State Parsing. In: 12th European Conference on Artificial Intelligence. John Wiley & Sons, Ltd., pp.1–3.

GUEDDAH, H., YOUSFI, A. dan BELKASMI, M., 2016. The filtered combination of the weighted edit distance and the Jaro-Winkler distance to improve spellchecking Arabic texts. Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA, 2016–July, pp.1–6.

HANCOX, P. dan POLATIDIS, N., 2012. Query Matching Evaluation in an Infobot for University Admissions Processing. Symposium on Languages, Applications and Technologies, [online] 21, pp.149–161. Tersedia di: <http://drops.dagstuhl.de/opus/volltexte/2012/3520/> [Diakses 27 Juli 2018]

MADISON, J., 2011. Damn You, Auyocorrect! [online] Ebury Publishing. Tersedia di: <https://books.google.co.id/books?id=ZXdD4hu3tuEC>.

MUTAMMIMAH; SUJAINI, HERRY; NYOTO, R.D., 2017. Analisis Perbandingan Metode Spelling Corrector Peter Norvig dan Spelling Checker BK-Trees pada Kata Berbahasa Indonesia. Jurnal Sistem dan Teknologi Informasi, 5(1), pp.12–16.

NOAMAN, H.M., SARHAN, S.S. dan RASHWAN, M.A.A., 2018. Enhancing recurrent neural network - based language models by word tokenization. Human-centric Computing and Information Sciences, [online] pp.1–13. Tersedia di: <https://doi.org/10.1186/s13673-018-0133-x> [Diakses 10 Juli 2018]

OMAR, N., 2018. Arabic Nested Noun Compound Extraction Based on Linguistic Features and Statistical Measures. Journal of Language Studies, 18(May), pp.93–107.

PANDYA, S. dan VIRPARIA, P., 2011. Testing Various Similarity Metrics and their Permutations with Clustering Approach in Context Free Data Cleaning. International Journal of Computer Science and …, [online] 3(5), pp.344–350. Tersedia di: <http://www.cscjournals.org/csc/manuscript/Journals/IJCSS/volume3/Issue5/IJCSS-155.pdf> [Diakses 27 Juli 2018]

PRAYOGO, A.H., MUBAROK, M.S. dan ADIWIJAYA, 2018. On the structure of Bayesian network for Indonesian text document paraphrase identification. Journal of Physics: Conference Series, 971(1), pp.1–15.

PRESSMAN, R.S., 2010. Software Engineering A Practitioner's Approach 7th Ed - Roger S. Pressman. Seventh Ed ed. New York: McGraw-Hill.

PRESSMAN, R.S. dan MAXIM, B.R., 2015. Software Engineering: A Practitioner's Approach. Eighth Edi ed. Mc Graw-Hill Higher Education. New York: Mc Graw-Hill Higher Education.

RAHARDIAN, B.A., KURNIANINGTYAS, D., MAHARDIKA, D.P., MAGHFIRA, T.N. dan CHOLISSODIN, I., 2017. Analisis Judul Majalah Kawanku Menggunakan Clustering K-Means Dengan Konsep Simulasi Big Data Pada Hadoop Multi Node Cluster. Jurnal Teknologi Informasi dan Ilmu Komputer, [online] 4(2), p.75. Tersedia di: <http://jtiik.ub.ac.id/index.php/jtiik/article/view/239> [Diakses 24 Maret 2018]

ROCHMAWATI, Y. dan KUSUMANINGRUM, R., 2016. Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching untuk Identifikasi Kesalahan Pengetikan Teks. Jurnal Buana Informatika, [online] 7(2), pp.125–134. Tersedia di: <http://ojs.uajy.ac.id/index.php/jbi/article/view/491> [Diakses 24 maret 2018]

SURYANINGRUM, K.M. dan T, A., 2016. Pengkoreksian dan Suggestion Word pada Keyword Menggunakan Algoritma Jaro-Winkler. Jurnal Teknologi Informasi-AITI, 13(2), pp.169–181.

WIDIANINGSIH, R.K., 2014. PADA BUKU TEKS MATA PELAJARAN BAHASA INDONESIA. Universitas Negeri Yogyakarta.

WINKLER, W.E., 1999. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Bureau of the Census.