

# *Sistem Koreksi Kesalahan Pengetikan Kata Kunci dalam Pencarian Artikel Menggunakan Algoritma Jaro-Winkler*

Ariadi Retno Tri Hayati Ririd<sup>1</sup>, Pramana Yoga Saputra<sup>2</sup>, Adita Mulya Sastr<sup>3</sup>

<sup>1,2,3</sup> Teknik Informatika, Teknologi Informasi, Politeknik Negeri Malang

<sup>1</sup> [ariadi.retno@polinema.ac.id](mailto:ariadi.retno@polinema.ac.id), <sup>2</sup> [pramanay@gmail.com](mailto:pramanay@gmail.com), <sup>3</sup> [aditamulyas97@gmail.com](mailto:aditamulyas97@gmail.com)

**Abstrak**— Kesalahan pengetikan atau yang kerap disebut dengan *typo* sangat lazim bagi pengguna teknologi. Kesalahan pengetikan meliputi pergeseran urutan karakter dalam satu kata serta perubahan karakter atau huruf dari satu kata yang dapat terjadi karena kesalahan menekan huruf pada *keyboard*. Sehingga dalam sebuah mesin pencari atau *search engine*, hal tersebut akan sangat berpengaruh pada hasil pencarian artikel. Untuk mengatasi masalah tersebut, diperlukan sebuah aplikasi yang dapat memperbaiki kata-kata yang salah urutan dan memberikan saran kata dengan menggunakan algoritma *Jaro-Winkler*. Algoritma *Jaro-Winkler* merupakan suatu metode untuk menghitung kesamaan antara dua buah *string* sehingga mampu memperbaiki kata yang salah urutannya sekaligus memberikan usulan kata. Data yang diambil adalah judul artikel dari Jurnal Informatika Polinema kemudian dijadikan sebagai sampel yang digunakan pada pencarian kata kunci. Hasil pengujian yang diperoleh yaitu dengan nilai *precision* untuk pergeseran urutan karakter sebesar 82%, pengurangan karakter sebesar 65%, dan *precision* untuk penambahan karakter sebesar 61%, perubahan karakter sebesar 68%, untuk nilai *recall* rata-rata adalah 100%.

**Kata kunci**— koreksi, kata kunci, search engine, algoritma *Jaro-Winkler*

## I. PENDAHULUAN

Kesalahan pengetikan atau *typo* adalah praktek yang biasa terjadi kala membuat suatu tulisan, misalnya penginputan kata kunci. Kesalahan pengetikan meliputi pergeseran urutan karakter dalam satu kata serta perubahan karakter atau huruf dari satu kata yang dapat terjadi karena kesalahan menekan huruf pada *keyboard*.

Pencarian biasanya melalui proses *search engine* atau mesin pencari. *Search engine* merupakan aktivitas yang biasa dilakukan oleh *user*. Pada dasarnya, mesin pencari bertujuan untuk menemukan bahan yang dicari pada suatu teks untuk menemukan informasi yang relevan berdasarkan kata kunci.

Sistem pencarian kata kunci dalam sebuah jurnal sangatlah penting. Namun terkadang dalam pengetikan kata kunci, *user* bisa saja salah mengetikkan kata yang dimaksud.

Hal ini dikarenakan salah pengetikan atau kata yang dimaksud tidak sesuai dengan yang ada di *database* sehingga sistem tidak menampilkan hasil yang diinginkan sehingga pengguna harus mengetikkan kembali kata kunci pada kotak pencarian. Dengan demikian, dibuatlah sistem pengkoreksian kata kunci dengan algoritma *Jaro-Winkler*.

Algoritma *Jaro-Winkler* adalah salah algoritma untuk mengukur kesamaan antara dua buah *string* [1]. Penambahan fitur koreksi kata kunci serta pemberian usulan kata menggunakan algoritma *Jaro-Winkler* sangat diperlukan untuk memberikan hasil yang akurat. Biasanya algoritma ini digunakan di dalam pendeteksian duplikat. Semakin tinggi nilai *Jaro-Winkler* untuk dua buah *string*, maka semakin tinggi presentase kemiripan kedua buah *string* tersebut [1].

Dalam penggunaan algoritma tersebut, dibutuhkan proses pengestrakan pola (informasi dan pengetahuan yang bermanfaat) dari sebagian besar sumber data tidak terstruktur yang dikenal dengan *Text Mining*.

Penerapan fitur pencarian judul artikel pada website Jurnal Informatika Polinema (JIP) memerlukan pengembangan pada proses pencariannya dengan menambahkan pengkoreksian kata kunci serta pemberian usulan kata sehingga dapat mempermudah *user* agar mengarahkan dalam pencarian judul artikel melalui kata kunci utama.

Oleh karena itu, penulis mengembangkan sistem mesin pencarian kata kunci pada website JIP dengan menambahkan fitur koreksi kata dengan judul “Sistem Koreksi Kesalahan Pengetikan Kata Kunci pada Pencarian Artikel Menggunakan Algoritma *Jaro-Winkler* (Studi Kasus : Website JIP)”.

## II. TINJAUAN PUSTAKA

### 2.1 *Text Mining*

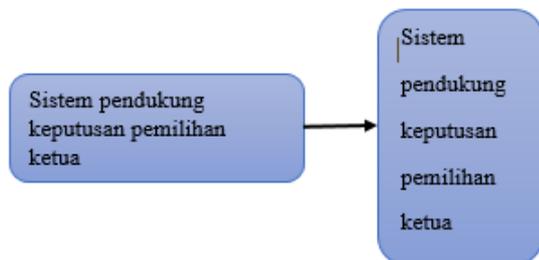
Pengertian *text mining* secara sempit hanya berupa metode yang dapat membentuk informasi baru yang spesifik atau mudah diketahui dari sebuah gugusan dokumen [3]. Secara umum mengacu pada proses penarikan intisari dari dokumen-dokumen teks tak terstruktur. Proses kerja *text mining* mencakup teknik *text-preprocessing* seperti pencarian, ekstraksi data, dan pengkategorian.

Dalam melakukan *text mining*, tahap awal yang dilakukan yaitu mempersiapkan teks dokumen yang akan digunakan terlebih dahulu, kemudian melakukan seleksi data yang akan diproses pada setiap dokumen. Proses ini berfungsi untuk mengubah bentuk data yang tidak terstruktur menjadi

data yang terstruktur. Secara umum, tahapan *preprocessing* meliputi :

a. *Tokenizing*

*Tokenizing* adalah proses dimana teks yang berupa kalimat, paragraf atau dokumen, diubah menjadi bagian-bagian/token-token tertentu [3]. Misalkan tokenisasi dari kalimat “Sistem pendukung keputusan pemilihan ketua”, menghasilkan lima token yaitu “Sistem”, “pendukung”, “keputusan”, “pemilihan”, “ketua”. Acuan sebagai pemisah antar token pada proses ini adalah spasi. Token-token tersebut yang akan diproses pada tahap berikutnya.

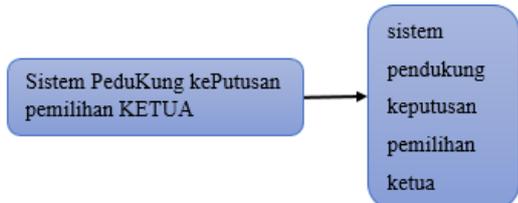


Gambar 1. Proses *Tokenizing*

b. *Case Folding*

*Case Folding* adalah proses dimana setiap string diubah menjadi huruf kecil atau non-kapital mulai dari ‘a’ sampai ‘z’. Karakter yang selain huruf dihilangkan dan dianggap delimiter (tanda koma, titik koma, titik dua, dsb.). Hal ini karena proses *Jaro-Winkler* bersifat *case sensitive* (peka terhadap huruf besar atau kecil) sehingga besar kecilnya huruf dapat mempengaruhi hasil perhitungan jarak kedekatannya [4].

Contoh pada proses *case folding* yaitu kata “Algoritma” diubah menjadi “algoritma”, sedangkan kata “algoritma” tidak berubah karena sudah dalam bentuk huruf kecil.



Gambar 2. Proses *Case Folding*

2.2 Algoritma *Jaro-Winkler*

*Jaro-Winkler* merupakan varian dari *Jaro distance* metrik yaitu sebuah algoritma untuk mengukur kesamaan antara dua *string*, biasanya algoritma ini digunakan di dalam pendeteksian duplikat. Semakin tinggi *Jaro-Winkler distance* untuk dua *string*, semakin mirip dengan *string* tersebut. *Jaro-Winkler distance* terbaik dan cocok untuk digunakan dalam perbandingan singkat seperti nama orang. Skor normalnya seperti 0 menandakan tidak ada kesamaan, dan 2 adalah sama persis [1], [5].

Setelah data yang diproses sudah selesai (*tokenizing* dan *case folding*), setiap kalimat yang sudah dipecah menjadi token diperiksa pengejaannya dengan daftar kata yang ada di *database*. Jika ditemukan adanya kecocokan maka pengejaan kata tersebut dianggap benar dan tidak perlu melalui proses perhitungan jarak. Akan tetapi jika tidak ada kata yang cocok dengan yang ada di *database* maka sistem menghitung jarak kedekatannya dengan daftar kata yang diindikasikan mempunyai

kemiripan berdasarkan jumlah karakter. Dasar dari algoritma *Jaro-Winkler* memiliki 3 (tiga) bagian, yaitu :

1. Menghitung panjang string,
2. Menemukan jumlah karakter yang “sama persis” (*m*) di dalam dua string, dan
3. Menemukan jumlah transposisi kedua buah string.

Pada algoritma *Jaro* digunakan rumus untuk menghitung jarak (*d<sub>j</sub>*) antara dua string yaitu *s<sub>1</sub>* dan *s<sub>2</sub>* sebagai berikut:

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

dimana:

*m* = jumlah karakter yang sama persis

|*s<sub>1</sub>*| = panjang *string* 1

|*s<sub>2</sub>*| = panjang *string* 2

*t* = jumlah transposisi

Dua karakter dari *s<sub>1</sub>* dan *s<sub>2</sub>* dikatakan cocok hanya jika tidak lebih jauh dari :

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

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 [4]. Biasanya *s<sub>1</sub>* digunakan sebagai acuan untuk urutan didalam mencari transposisi. Yang dimaksud transposisi disini adalah karakter yang sama dari *string* yang dibandingkan akan tetapi tertukar urutannya [4].

*Jaro-Winkler* menggunakan faktor penskalaan (*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 [4]. Pengukuran algoritma *Jaro-Winkler* adalah ekstensi dari algoritma *Jaro distance*.

$$d_w = d_j + (lp(1 - d_j)) \quad (3)$$

dimana:

*d<sub>j</sub>* = *Jaro distance* untuk *string s<sub>1</sub>* dan *s<sub>2</sub>*

*l* = panjang prefix umum di awal *string* nilai maksimalnya 4 karakter (panjang karakter yang sama sebelum ditemukan ketidaksamaan maksimal 4)

*p* = konstanta scaling factor. Nilai standar untuk konstanta ini menurut Winkler adalah *p* = 0,1.

Ekstensi ini dibuat berdasarkan hasil observasi Winkler bahwa kesalahan pengetikan biasanya muncul di tengah pengetikan atau di akhir kata, sangat jarang terjadi kesalahan pengetikan di awal kata [3].

Sebagai contoh, ketika *user* berniat mengetikkan kata “sistem” namun terjadi kesalahan pengetikan sehingga yang diketik adalah kata “sisrem”. Tidak ditemukan kata “sisrem” pada daftar kata di *database* sehingga kata “sisrem” akan dihitung jarak kedekatannya dengan semua kata dengan awalan sama yang berjumlah lima sampai tujuh karakter yang terdaftar di *database* menggunakan algoritma *Jaro-Winkler*. Contoh : “sistole”, “sistem”, “siswa”, “siswi”.

Perhitungan kedekatan jarak *string s<sub>1</sub>* yaitu “sisrem” dengan *string s<sub>2</sub>* yaitu “sistole”, berikut perhitungannya menggunakan algoritma *Jaro-Winkler*.

String  $s_1 = \text{sisrem}$

String  $s_2 = \text{sistole}$

Jumlah karakter string  $|s_1| = 6$

Jumlah karakter string  $|s_2| = 7$

Jumlah karakter yang sama  $s_1$  dan  $s_2$   $m = 4$

Jumlah transposisi  $t = 0$

$$dj = \frac{1}{3} x \left( \frac{4}{|6|} + \frac{4}{|7|} + \frac{4-0}{4} \right) = 0.190$$

Kemudia jika diperhatikan susunan  $s_1$  dan  $s_2$  dapat diketahui nilai kesamaan awalan atau *prefix length* ( $l$ ) adalah 3, karena karakter yang sama sebelum ditemukan adanya ketidaksamaan adalah karakter “s”, “i”, dan “s”. Nilai konstan *scaling factor* ( $p$ ) adalah 0,1. Maka nilai *Jaro-Winkler distance* yaitu :

$$dw = 0.190 + (3 \times 0.1(1 - 0.190)) = 0.433$$

Berdasarkan perhitungan tersebut, maka didapatkan hasil sebagai berikut :

TABEL I. PERHITUNGAN ALGORITMA JARO-WINKLER

No.	String 1	String 2	Nilai Jaro-Winkler
1	Sisrem	Sistole	0.433
2	Sisrem	Sistem	0.923
3	Sisrem	Siswa	0.447
4	Sisrem	Siswi	0.447

Dengan demikian, dapat diketahui bahwa *string1* “sisrem” sangat dekat jarak katanya dengan *string2* “sistem” yang memiliki nilai *Jaro-Winkler* sebesar 0,923. Semakin dekat nilai *Jaro-Winkler* dengan 1 maka semakin besar pula kedekatan jarak kata yang salah tersebut dengan kata yang ada di *database*.

### 2.3 Penelitian Terdahulu

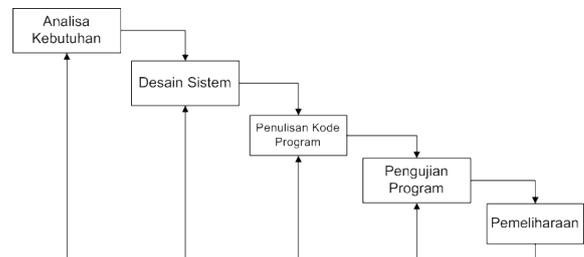
Pada penelitian kasus-kasus sebelumnya telah dilakukan penelitian mengenai pengkoreksian kesalahan pengetikan menggunakan algoritma *Jaro-Winkler* dengan judul “*Pengkoreksian dan Suggestion Word pada Keyword Menggunakan Algoritma Jaro-Winkler*” [3]. Hasil dari penelitian tersebut bahwa algoritma *Jaro-Winkler* dapat diimplementasikan untuk pengkoreksian pada sebuah *keyword* bahan pustaka. Algoritma ini dapat diterapkan dalam memberikan rekomendasi kata, jika ada salah pengetikan dalam menginputkan suatu kata saat pencarian bahan pustaka.

Selain itu pada penelitian lain tahun 2015 dengan judul “*Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching untuk Identifikasi Kesalahan Pengetikan Teks*” [1]. Menyatakan bahwa algoritma *Jaro-Winkler* memiliki nilai tertinggi dibandingkan dengan ketiga algoritma yang lain (Algoritma *Levenshtein Distance*, Algoritma *Damerau Levenshtein Distance*, dan Algoritma *Hamming Distance*) yang terbagi kedalam empat jenis kesalahan penulisan yaitu jenis penghapusan huruf 0,92, jenis kesalahan penambahan huruf 0,90, jenis kesalahan penggantian huruf 0,70, dan jenis kesalahan penukaran urutan huruf 0,95.

## III. METODE PENELITIAN

### 3.1 Tahapan Penelitian

Metode penelitian dalam pengembangan sistem koreksi kata kunci ini menggunakan model *waterfall* (*Classic Life Cycle*). Metode *waterfall* adalah suatu proses pembuatan situs *web* secara terstruktur dan berurutan dimulai dari penentuan masalah, analisa kebutuhan, perancangan implementasi, integrasi, uji coba sistem, penempatan situs *web* dan pemeliharaan [3].



Gambar 3. Model Waterfall

### 3.2 Pengolahan Data

Kegiatan yang dilakukan pada proses pengolahan data yaitu peneliti mengumpulkan data artikel-artikel yang diambil dari JIP. Tabel 2 berikut adalah data yang dikumpulkan oleh peneliti sejak tahun 2014 sampai 2018 sebanyak 168 judul.

TABEL 2. DATA JUDUL ARTIKEL

No.	Judul Artikel
1	Rancangan Bangun Sistem Informasi Lowongan Kerja Di Jpc Polinema Dengan Metode Quick Sort
2	Aplikasi Pencarian Penjualan Laptop Menggunakan Teknologi Web Scarping
3	Implementasi Explicit Semantic Analysis Berbahasa Indonesia Menggunakan Corpus Wikipedia Indonesia
4	Pengenalan Tas Ransel Pada Citra Digital Dengan Ekstraksi Fitur Tekstur Menggunakan Metode Gray Level Co-Occurrence Matrix
5	Rancang Bangun Game Maze 2d “Return To Earth”
6	Sistem Pengenalan Wajah Untuk Keamanan Folder Menggunakan Metode Triangle Face
7	Peramalan Stok Obat Di Puskesmas Gending Probolinggo Menggunakan Metode Winter’s Exponential Smoothing
8	Implementasi Metode Naive Bayes Untuk Intrusion Detection System (Ids)
9	Robot Pengetik Untuk Alat Bantu Pengoperasian Komputer Bagi Penyandang Cacat
10	Implementasi Fsm (Finite State Machine) Pada Game Perjuangan Pangeran Diponegoro
...	...
168	Sistem Pakar Diagnosa Penyakit Diabetes Melitus

Tabel 3 merupakan sekumpulan daftar kata yang telah dipecah dari judul artikel. Kata-kata inilah yang kemudian akan dibandingkan dengan kata kunci yang dimasukkan oleh *user* untuk dapat menghasilkan usulan kata kunci bagi *user* dalam melakukan proses pencarian selanjutnya secara lebih akurat.

TABEL 3. DAFTAR KATA

Id	Kata	Id	Kata
1	Pengetik	12	Peramalan
2	Android	13	Informasi
3	Analisis	14	Notifikasi

4	Algoritma	15	Otomatis
5	Aplikasi	16	Pengelolaan
6	Komputer	17	Pemetaan
7	Berbasis	18	Raspberry
8	Perangkat	19	Informatika
9	Teknologi	20	Klasifikasi
10	Penerapan	...	...
11	Implementasi	1328	Virtual

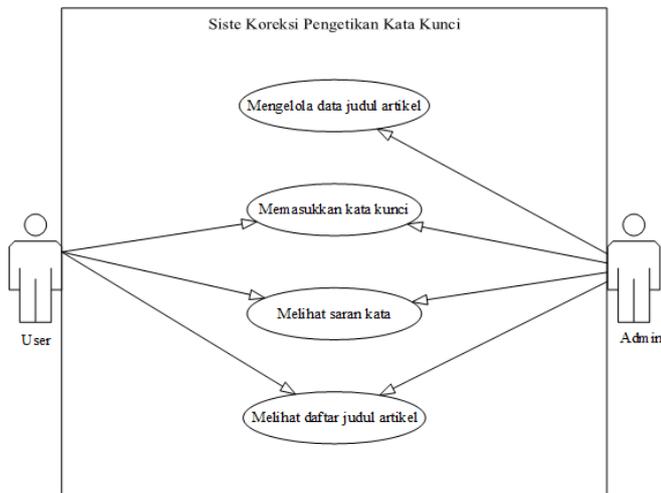
### 3.3 Metode Pengujian

Proses pengujian yaitu beberapa kata diujicoba dengan beberapa kesalahan pengetikan sehingga mampu mengoreksi sekaligus menampilkan usulan kata yang hasil perhitungannya bernilai tinggi. Dapat dilihat pada tabel 1 yakni kata dengan nilai tertinggi yang akan ditampilkan sebagai usulan kata.

## IV. PERANCANGAN

### 4.1 Use Case System

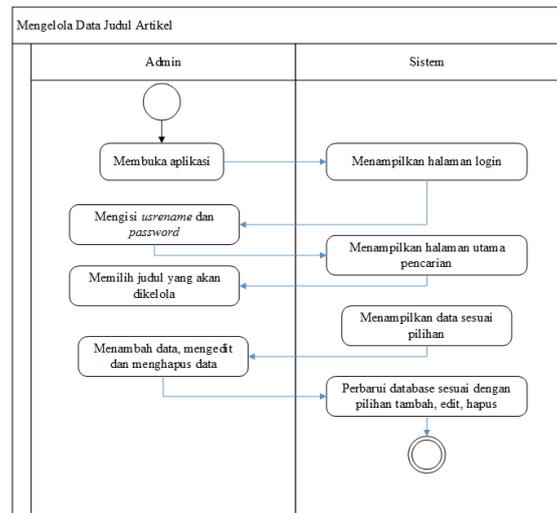
Gambar 5 merupakan usecase diagram sistem yang terdiri dari dua aktor yaitu *user* dan *admin*. *User* dapat melakukan pencarian kata kunci pada mesin pencari, juga mampu melihat usulan kata. Selanjutnya *admin*, selain dapat melakukan pencarian kata kunci, *admin* juga memiliki akses login serta dapat menambahkan artikel baru, mengubah data artikel, dan akses untuk menghapus data artikel.



Gambar 4. Use Case System

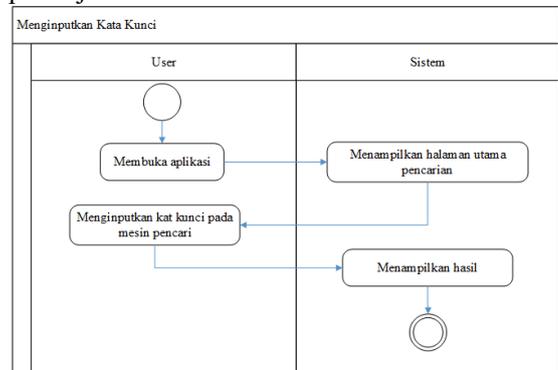
### 4.2 Activity Diagram

*Activity* diagram merupakan alur kerja yang berupa kerangka visual yang berisi aktivitas dan tindakan, yang juga dapat berisi alternatif lain berupa pilihan atau perulangan. Dalam UML (*Unified Modeling Language*), *activity* diagram digunakan sebagai bentuk alur aktivitas komputer dan juga memaparkan aliran pesan dari satu aktivitas ke aktivitas lainnya.



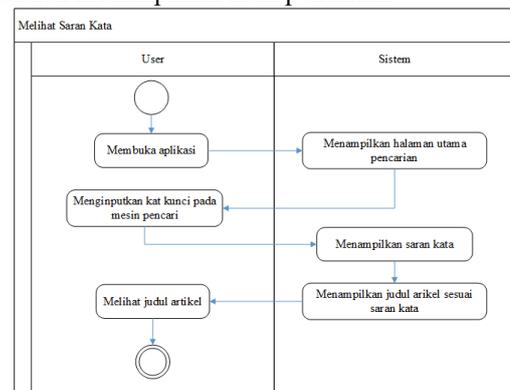
Gambar 5. Activity diagram mengelola data judul artikel

Gambar 5 merupakan *activity* diagram pengolahan data judul artikel yang hanya bisa diakses oleh *admin*. *Admin* membuka aplikasi kemudian sistem akan menampilkan halaman login yang berisi *username* dan *password*. Reaksi sistem selanjutnya yaitu menampilkan halaman utama pencarian *admin*. Selain adanya fitur koreksi kata kunci, disini *admin* mengelola data berupa perubahan, penambahan, serta penghapusan judul artikel.



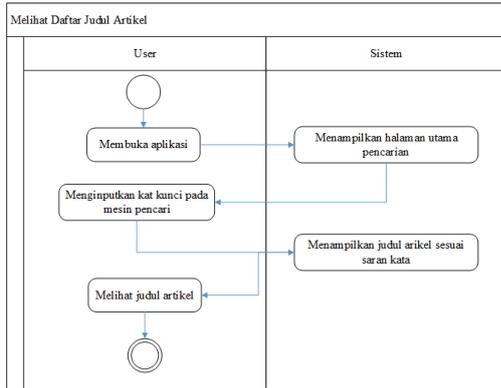
Gambar 6. Activity diagram menginputkan kata kunci

Gambar 6 merupakan *activity* dari menginputkan kata kunci dimana *user* masuk ke sistem setelah itu sistem akan menampilkan halaman utama pencarian. Kemudian *user* memasukkan kata kunci pada kotak pencarian yang sudah tersedia selanjutnya menekan tombol *search*. Reaksi akhir sistem adalah menampilkan hasil pencarian.



Gambar 7. Activity diagram melihat saran kata

Gambar 7 merupakan *activity* diagram yang menggambarkan aksi *user* melihat saran kata. *User* membuka aplikasi kemudian sistem akan menampilkan halaman utama pencarian, selanjutnya *user* menginputkan kata kunci pada kotak pencarian. Dengan begitu sistem dapat menampilkan saran kata yang sesuai atau yang mendekati kata kunci.



Gambar 8. Activity diagram melihat daftar judul artikel

Gambar 8 merupakan proses *activity* diagram dalam melihat judul artikel; setelah melakukan pencarian judul artikel melalui mesin pencari seperti yang dijelaskan pada *activity* digaram sebelum-sebelumnya, disini sistem akan menampilkan judul artikel sesuai dengan sara kata yang diberikan oleh sistem.

## V. IMPLEMENTASI

Implementasi adalah penulisan bahasa pemrograman pada komputer, agar kode program dapat berjalan sesuai dengan rancangan. Berikut langkah langkah pada tahap implementasi.

### 5.1 Implementasi Basis Data

Pada tahap ini dilakukan implementasi dari perancangan *database* yang dibentuk menggunakan MySQL.

Tabel	Tindakan
tb_hasil	★ Jelajahi Struktur Cari Tambahkan Kosongkan Hapus
tb_kata	★ Jelajahi Struktur Cari Tambahkan Kosongkan Hapus
tb_koleksiartikel	★ Jelajahi Struktur Cari Tambahkan Kosongkan Hapus
tb_relasi	★ Jelajahi Struktur Cari Tambahkan Kosongkan Hapus
user	★ Jelajahi Struktur Cari Tambahkan Kosongkan Hapus
5 tabel	Jumlah

Gambar 9. Tabel koleksi artikel

### 5.2 Implementasi Antarmuka

Implementasi antarmuka berisi tentang tampilan *interface* dari sistem yang sudah didesain pada bab sebelumnya. Terdiri dari implementasi tampilan halaman *user*, dan tampilan halaman admin

No	ID	JUDUL ARTIKEL	AUTHOR	VOLUME	NOMOR	TAHUN
1	1	RANCANGAN BANGUN SISTEM INFORMASI LOWONGAN KERJA DI JPC POLINEMA DENGAN METODE QUICK SORT	Budi Harjanto, Daddy Kusbiarto Purwoko Aji, Julia Intan Amiri	4	4	2018
2	2	APLIKASI PENCARIAN PENJUALAN LAPTOP MENGGUNAKAN TEKNOLOGI WEB SCRAPING	Yan Watequlis Syafudin, Arida Feri Syafliandini, Hafid Rizy Prisdadana	4	4	2018
3	3	IMPLEMENTASI EXPLICIT SEMANTIC ANALYSIS BERBAHASA INDONESIA MENGGUNAKAN CORPUS WIKIPEDIA INDONESIA	Faisal Rahutomo, Pramana Yoga Saputra, Carlin Febriawan Pratama Putra	4	4	2018
4	4	PENGENALAN TAS RANSEL PADA CITRA DIGITAL DENGAN EKSTRAKSI FITUR TEKSTUR MENGGUNAKAN METODE GRAY LEVEL CO-OCCURRENCE MATRIX	Cahya Rahmad, Mungki Astiningrum, Ada Putra Lesmana	4	4	2018
5	5	RANCANG BANGUN GAME MAZE 2D "RETURN TO EARTH"	Ridwan Rismanto, Dyah Ayu Irawati, Ari Mahardika Ahmad Nofis	4	4	2018

Gambar 10. Tampilan halaman *user*

No	ID	JUDUL ARTIKEL	AUTHOR	VOLUME	NOMOR	TAHUN	AKSI
1	1	RANCANGAN BANGUN SISTEM INFORMASI LOWONGAN KERJA DI JPC POLINEMA DENGAN METODE QUICK SORT	Budi Harjanto, Daddy Kusbiarto Purwoko Aji, Julia Intan Amiri	4	4	2018	[Edit] [Hapus]
2	2	APLIKASI PENCARIAN PENJUALAN LAPTOP MENGGUNAKAN TEKNOLOGI WEB SCRAPING	Yan Watequlis Syafudin, Arida Feri Syafliandini, Hafid Rizy Prisdadana	4	4	2018	[Edit] [Hapus]
3	3	IMPLEMENTASI EXPLICIT SEMANTIC ANALYSIS BERBAHASA INDONESIA MENGGUNAKAN CORPUS WIKIPEDIA INDONESIA	Faisal Rahutomo, Pramana Yoga Saputra, Carlin Febriawan Pratama Putra	4	4	2018	[Edit] [Hapus]

Gambar 11. Tampilan halaman admin

## 5.3 Implementasi Proses

ID	JUDUL ARTIKEL	AUTHOR	VOLUME	NOMOR	TAHUN
24	IMPLEMENTASI TWITTER SENTIMENT ANALYSIS UNTUK REVIEW FILM MENGGUNAKAN ALGORITMA SUPPORT VECTOR MACHINE	Faisal Rahutomo, Pramana Yoga Saputra, Mithad Agtamas Prayawan	4	2	2018
33	PENERAPAN ALGORITMA NAIVE BAYES UNTUK KLASIFIKASI RETENSI AIRSP	Budi Harjanto, Yuli Aryanita, Iustika Mitha Humalia	4	2	2018
62	RANCANG BANGUN APLIKASI PENCARIAN RITEL TERPADUK JASA KIRIMAN BARANG BERBASIS MOBILE DENGAN	Martin Nugroho Paragot, Daddy Kusbiarto, Cahya	3	3	2017

Gambar 12. Tampilan halaman pencarian

Setelah *user* selesai menginputkan kata kunci, maka sistem akan menampilkan saran kata beserta judul artikel yang sesuai dengan hasil saran seperti yang terlihat pada gambar 12. Dengan menginputkan kata "agloritma" yang dimaksudkan adalah "algoritma" ini termasuk dalam pergeseran urutan karakter dalam satu kata dengan hasil perhitungan sebesar 0.96 yang berarti mendekati nilai 1 kemiripan kata.

## VI. HASIL DAN PEMBAHASAN

Proses uji coba dilakukan dengan sub-sub uji coba fungsional dan sub-sub uji coba akurasi pada sistem. Pengujian fungsional bertujuan untuk mengetahui apakah sistem yang dibangun telah menyajikan fitur-fitur sesuai dengan yang dibutuhkan. Untuk pengujian sistem koreksi pencarian menggunakan metode perhitungan *precision scale* dan *recall*. *Precision scale* adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Sedangkan *recall* adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi.

### 6.1 Pengujian Fungsionalitas

Berdasarkan hasil pengujian yang dilakukan, pengujian secara fungsionalitas dengan menguji setiap menu sudah berjalan dengan baik. Status uji dinyatakan berhasil setelah dilihat dari proses sistem pada pengujian. Hal ini ditunjukkan dengan adanya kesesuaian terhadap analisis dan rancangan yang telah dibahas pada tahap analisis dan perancangan.

### 6.2 Pembahasan Hasil Pengujian

Analisis hasil uji coba yang didapatkan setelah melakukan uji coba terhadap sistem dengan menginputkan sebanyak 10 query, dengan kesalahan query berupa penghapusan karakter, pergeseran urutan karakter, penambahan karakter, perubahan karakter pada satu kata, serta penggabungan beberapa kata.

TABEL4. QUERY

Query	Precision	Recall
mteode froward cahining	0,89	1
pnenentuan loksai klinki	0,90	1
vsiualisasi gis	0,85	1
satr modle aegncy	0,25	1
anlaisa froecasting	0,80	1
adpative leanring vectro	0,71	1
stegnaografi ranodm btye	0,42	1
pengmembangan sisetm infomasi	0,65	1
diangosa penyaikt	0,6	1
fuzyz tuskamoto	0,81	1
<b>Rata-Rata</b>	<b>0,69</b>	<b>1</b>

Hasil pengujian pada Tabel 4 dengan query pergeseran urutan karakter mendapatkan hasil rata-rata dari setiap query yang diuji yaitu precision sebesar 69% dan recall sebesar 100%. Dengan nilai recall demikian dapat dikatakan bahwa seluruh dokumen yang dicari dengan query tersebut bisa ditemukan. Sedangkan pada precision terdapat beberapa data yang tidak relevan yakni menjadi saran kata dikarenakan nilai batas ambang dari perhitungan algoritma Jaro-Winkler adalah 0.80 hingga 0.90. Hal ini mengakibatkan nilai kata yang dibandingkan dengan hasil diatas 0.80 akan menjadi saran kata sehingga judul yang tidak termasuk dalam kata kunci pun akan terpanggil.

## VII. KESIMPULAN DAN SARAN

### 7.1 Kesimpulan

Berdasarkan hasil pengujian dan pembahasan dari sistem pengkoreksian kata kunci dalam pencarian artikel menggunakan algoritma Jaro-Winkler maka diperoleh kesimpulan sebagai berikut:

1. Pengembangan sistem ini dilakukan melalui beberapa tahapan penelitian yakni identifikasi masalah, studi literatur, pengumpulan data, perancangan, implementasi dan testing. Identifikasi masalah diperoleh berdasarkan latar belakang penelitian. Studi literatur diperoleh dari melalui artikel-artikel mengenai penelitian terdahulu guna mendapatkan dasar teori terhadap masalah yang diteliti.

Pengumpulan data didapatkan dari website JIP. Tahap selanjutnya merupakan perancangan alur sistem aplikasi. Implementasi dibuat berdasarkan perancangan yang telah dilakukan. Pengujian program atau testing dilakukan dengan dua metode pengujian yaitu pengujian yang berfokus pada fungsionalitas sistem dan pengujian menggunakan precision dan recall.

2. Algoritma Jaro-Winkler mampu memberikan hasil dengan nilai precision untuk query pergeseran urutan karakter sebesar 82 % dan recall sebesar 100%, kemudian precision untuk query pengurangan karakter sebesar 65% dan recall sebesar 100%, precision untuk query penambahan karakter sebesar 61% dan recall sebesar 100%, serta precision untuk query perubahan karakter sebesar 68% dan recall 100%.

### 7.2 Saran

Saran yang dapat diberikan dari hasil penelitian untuk pengembangan sistem ini ke depan adalah sebagai berikut:

1. Pengembangan sistem selanjutnya seharusnya hanya menampilkan saran kata yang terdapat kesalahan kata pada kata kunci yang diinputkan.
2. Sistem dapat dikembangkan memisah jenis kata yang berimbuhan dan menghilangkan kata imbuhan.
3. Data index dapat ditambahkan dengan jenis kata-kata yang banyak mengandung istilah informatika.

## DAFTAR PUSTAKA

- [1] Rochmawati, Yeny dan Kusumaningrum, "Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching untuk Identifikasi Kesalahan Pengetikan Teks", Jurnal Buana Informatika, vol. 7, no. 2, pp. 125-134, April 2016.
- [2] S.Margi, Kristian dan T.Agus, "Pengkoreksian dan Suggestion Word pada Keyword Menggunakan Algoritma Jaro-Winkler", Jurnal Teknologi Informasi, vol. 13, no. 2, pp. 169-181, 2016.
- [3] Prasetyo, Baihaqi dan Had, Agung, "Algoritma Jaro-Winkler Distance: Fitur Autocorrect dan Spelling Suggestion pada Penulisan Naskah Bahasa Indonesia Di BMS TV", JTIK, vol 5, no 4, pp 435-444, 2018.
- [4] Novantara, Panji dan Pasruli, "Impelementasi Algoritma Jaro-Winkler Distance untuk Sistem Pendeteksi Plagiarisme pada Dokumen Skripsi", Jurnal Buffer Informatika, vol 3, no 2, ISSN 2527-4856, 2017.
- [5] "Jurnal Informatika Polinema" [online]. Desember 2018 Available : <http://jip.polinema.ac.id/ojs3/index.php/jip/about>
- [6] Christina, Sherly, etc. "Mendeteksi Plagiarism pada Doukemn Proposal Skripsi Menggunakan Algoritma Jaro Winkler Distance".
- [7] Priansya, Stezar. "Normalisasi Teks Media Sosial Menggunakan Word2vec, Levenshtein Distance, dan Algoritma Jaro-Winkler Distance", 2017.
- [8] Kurniawati, Anna, etc. "Implementasi Algoritma Jaro-Winkler Distance untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia", 2010.
- [9] Anggara, Yudha, "Deteksi Plagiarisme Dokumen Bahasa Indonesia dengan Algoritma jaro-Winkler", 2016.
- [10] Friendly, "Perbaikan Metode Jaro-Winkler Distance untuk Approximate String Search Menggunakan Data Terindeks Aplikasi Multiuser", Jurnal Teknovasi, vol 4, no 2, pp. 69 – 78, ISSN : 2540-8389, 2017.