

# **SKRIPSI**

## **KOREKSI KESALAHAN EJAAN TERHADAP *QUERY* PENCARIAN ARTIKEL PARIWISATA BERBAHASA INDONESIA MENGGUNAKAN ALGORITMA *LEVENSHTTEIN DISTANCE* DAN *PART-OF-SPEECH* (POS) *TAGGING***



**Disusun Oleh:**

Aisyatur Radiah

NIM 200411100116

**Dosen Pembimbing 1 : Ika Oktavia Suzanti S.Kom., M.Cs**

**Dosen Pembimbing 2 : Husni S.Kom., M.T.**

**PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNIK INFORMATIKA  
FAKULTAS TEKNIK  
UNIVERSITAS TRUBARIS KE JOYO MADURA  
BANGKALAN**

**2024**

## ABSTRAK

*Spelling correction* merupakan sebuah fitur otomatis yang digunakan untuk melakukan proses pengoreksian kesalahan ejaan kata atau *query*. Ketika kata kunci (*query*) yang diinputkan terjadi kesalahan ejaan pada kata-kata didalam *query* mesin pencarian (*search engine*), salah satunya *search engine* bahasa Indonesia (SEBI). Hal ini mengakibatkan pemrosesan *query* tersebut mengembalikan hasil yang tidak sesuai dengan kebutuhan informasi pengguna atau disebut dengan *error spelling correction*. Adanya kesalahan ejaan kata pada *query* ini dapat diatasi menggunakan algoritma yaitu *Levenshtein Distance*. Algoritma *Levenshtein Distance* merupakan sebuah matriks untuk menghitung jarak dari jumlah perbedaan dua *string* yaitu *string* sumber dan *string* target, dengan menghitung perubahan koreksi ejaan *query* yang diinputkan dengan melihat kamus bahasa Indonesia yang berkaitan dengan artikel pariwisata. Algoritma *Levenshtein Distance* mempunyai kekurangan yaitu tidak mampu mengoreksi kata yang bermakna banyak (ambiguitas kata), adanya *Part-of-Speech* (POS) Terdapat mengetahui adanya ambiguitas kata dengan pelabelan kelas kata dalam *error spelling corection*. Penerapan metode *Levenshtein Distance* dan *Part-of-Speech* (POS) *Tagging* diperoleh nilai presisi 97.59% dan 97.59% tanpa *Part-of-Speech* (POS) *Tagging*.

**Kata Kunci :** *spelling correction, levenshtein distance, part-of-speech* (POS) *tagging, artikel pariwisata*.

## DAFTAR ISI

<b>ABSTRAK</b> .....	ii
<b>DAFTAR ISI</b> .....	iii
<b>DAFTAR GAMBAR</b> .....	vii
<b>DAFTAR TABEL</b> .....	viii
<b>DAFTAR PROGRAM</b> .....	ix
<b>BAB I</b> .....	10
<b>PENDAHULUAN</b> .....	10
1.1 Latar Belakang.....	10
1.2 Rumusan Masalah.....	14
1.2.1 Permasalahan .....	14
1.2.2 Metode Usulan .....	14
1.2.3 Pertanyaan Penelitian .....	14
1.3 Tujuan dan Manfaat Penelitian.....	14
1.3.1 Tujuan Penelitian.....	14
1.3.2 Manfaat Penelitian.....	15
1.4 Batasan Masalah .....	15
1.5 Sistematika Penulisan .....	15
<b>BAB II</b> .....	17
<b>KAJIAN PUSTAKA</b> .....	17
2.1 <i>Spelling Correction</i> (Koreksi Ejaan).....	17
2.1.1 Ejaan .....	18
2.1.2 <i>Typographical Error</i> .....	19
2.2 <i>Search Engine</i> .....	19
2.2.1 <i>Information Retrieval</i> (IR) .....	20
2.2.2 <i>Query</i> .....	21
2.3 Artikel Berita Pariwisata .....	22

2.3.1	Data artikel berita pariwisata .....	23
2.3.2	<i>Inverted Index</i> .....	23
2.4	<i>Text Preprocessing</i> .....	24
2.5	<i>Part-of-Speech (POS) Tagging</i> Bahasa Indonesia .....	24
2.6	Metode <i>Part-of-Speech (POS) Tagging</i> dengan Metode <i>Rule Based</i> .....	26
2.7	Pembobotan <i>Term Frequency Inverse Document Frequency (TF IDF)</i> .....	26
2.8	<i>Cosine Similarity</i> .....	27
2.9	Algoritma <i>Levenshtein Distance</i> .....	28
2.10	Evaluasi Sistem.....	30
2.10.1	Perhitungan <i>Precision</i> .....	30
2.10.2	Perhitungan <i>Recall</i> .....	30
2.11	Penelitian Terkait .....	31
<b>BAB III</b>	.....	35
<b>METODE USULAN</b>	.....	35
3.1	Tahapan Penelitian.....	35
3.1.1	Studi Literatur .....	35
3.1.2	Analisis dan Perancangan Arsitektur Sistem .....	35
3.1.3	Implementasi Sistem .....	37
3.1.4	Uji Coba Sistem .....	37
3.1.5	Analisa dan Evaluasi.....	38
3.2	Dataset.....	38
3.2.1	Data <i>Corpus</i> .....	38
3.2.2	Data Kamus.....	38
3.2.3	<i>Inverted Index</i> .....	39
3.2.4	Data Uji.....	36
3.3	<i>Text Preprocessing</i> .....	37
3.4	Arsitektur Sistem .....	39

3.5	<i>Flowchart</i> Algoritma .....	42
3.5.1	<i>Flowchart Levenshtein Distance</i> .....	42
3.5.2	Perhitungan Metode <i>Levenshtein Distance</i> .....	44
3.5.3	Penerapan <i>Part-of-Speech (POS) Tagging</i> .....	46
3.5.4	<i>TF-IDF</i> dan <i>Cosine Similarity</i> .....	51
3.6	Skenario Pengujian .....	53
3.7	Implementasi Dataset .....	54
3.7.1	Dataset .....	54
3.7.2	Pelabelan Dataset .....	54
3.7.3	Visualisasi Data dengan <i>Word Cloud</i> .....	55
3.7.4	Hasil Visualisasi Data .....	56
3.7.5	Tahapan <i>Preprocessing</i> Data .....	56
3.8	Perkiraan Jadwal .....	58
<b>BAB IV</b> .....		59
<b>HASIL DAN PEMBAHASAN</b> .....		59
4.1	Lingkungan Uji Coba .....	59
4.2	Tahapan Visualisasi Kata .....	61
4.2.1	Visualisasi Data dengan <i>WordCloud</i> .....	61
4.2.2	Tahapan <i>Preprocessing</i> Data .....	63
4.3	Implementasi <i>Function</i> Program .....	65
4.3.1	<i>Levenshtein Distance</i> dan <i>Part-Of-Speech (POS) Tagging</i> .....	65
4.3.2	Fungsi Koreksi Ejaan .....	66
4.3.3	Fungsi Modul dalam <i>Part-Of-Speech (POS) Tagging</i> .....	68
4.3.4	Fungsi Menghitung Nilai Presisi .....	68
4.3.5	Fungsi <i>TF-IDF</i> .....	70
4.3.6	Fungsi <i>Inverted Index</i> .....	71
4.3.7	Fungsi Pencarian Artikel Berita Berdasarkan <i>Cosine Similarity</i> .....	72

4.4 Implementasi Program Koreksi Ejaan.....	73
4.4.1 Implementasi Koreksi Ejaan terhadap <i>Query</i> Menggunakan <i>Levenshtein Distance</i>	73
4.4.2 Implementasi Koreksi Ejaan terhadap <i>Query</i> Menggunakan <i>Levenshtein Distance</i> dan <i>Part-Of-Speech (POS) Tagging</i> .....	74
4.4.3 Implementasi Koreksi Ejaan terhadap <i>Query</i> Pencarian Artikel Pariwisata Menggunakan <i>Levenshtein Distance</i> dan <i>Part-Of-Speech (POS) Tagging</i> .....	75
4.5 Analisa Hasil dan Uji Coba .....	79
4.5.1 Pengujian dan Analisis .....	80
4.5.2 Hasil Skenario Pengujian Koreksi Ejaan dari <i>Query</i> Berita Pariwisata Berbahasa Indonesia Menggunakan <i>Levenshtein Distance</i> dan <i>Part-of-Speech (POS) Tagging</i> .....	86
4.5.3 Analisa Hasil Skenario Uji Coba Sistem.....	103
4.6 Implementasi Sistem .....	104
4.7 Evaluasi Sistem.....	108
<b>BAB V</b> .....	96
<b>KESIMPULAN</b> .....	96
5.1 Kesimpulan.....	96
5.2 Saran.....	96
<b>REFERENSI</b> .....	97

## DAFTAR GAMBAR

Gambar 3. 1 Diagram IPO.....	36
Gambar 3. 2 <i>Inverted index</i> .....	39
Gambar 3. 3 Diagram Alur <i>Preprocessing</i> .....	37
Gambar 3. 4 Arsitektur Sistem .....	39
Gambar 3. 5 <i>Flowchart Levenshtein Distance</i> .....	43
Gambar 3. 6 Percobaan metode <i>Levenshtein Distance</i> di <i>Excel</i> .....	45
Gambar 3. 7 <i>Flowchart TF-IDF</i> .....	52
Gambar 3. 8 <i>Part-Of-Speech</i> (POS) <i>Tagging</i> Data Artikel Pariwisata .....	55
Gambar 3. 9 <i>WordCloud</i> Data Artikel Pariwisata .....	56
Gambar 4. 1 Grafik Hasil Presisi 100 data <i>Query</i> tanpa <i>Part-Of_Speech</i> (POS) <i>Tagging</i>	102
Gambar 4. 2 Grafik Hasil Presisi 100 data <i>Query</i> dan <i>Part-Of-Speech</i> (POS) <i>Tagging</i> .....	103
Gambar 4. 3 Tampilan inputan <i>query</i> .....	105
Gambar 4. 4 inputan <i>query</i> menggunakan metode <i>Levenshtein Distance</i> tanpa <i>Part-Of-Speech</i> (POS) <i>Tagging</i> .....	106
Gambar 4. 5 Tampilan skenario <i>Lavenshtein Distance</i> dan <i>Part-Of-Speech</i> (POS) <i>Tagging</i>	106
Gambar 4. 6 Menampilkan hasil pencarian Artikel .....	107
Gambar 4. 7 Hasil koreksi ejaan.....	107
Gambar 4. 8 Menampilkan Artikel tanpa <i>Part-Of-Speech</i> (POS) <i>Tagging</i> .....	108

## DAFTAR TABEL

Tabel 2. 1 Penelitian Terkait .....	32
Tabel 3. 1 Data Kamus .....	39
Tabel 3. 2 Dokumen artikel pariwisata.....	40
Tabel 3. 3 Proses Hitung Term dalam setiap dokumen .....	34
Tabel 3. 4 <i>Inverted index</i> .....	36
Tabel 3. 5 Data Uji .....	36
Tabel 3. 6 <i>Case Folding</i> .....	38
Tabel 3. 7 <i>Cleaning</i> .....	38
Tabel 3. 8 <i>Tokenizing</i> .....	38
Tabel 3. 9 <i>Tagset POS Tagging</i> .....	46
Tabel 3. 10 Contoh <i>Part-of-Speech (POS) Tagging</i> .....	47
Tabel 3. 11 Contoh artikel ambigu.....	48
Tabel 3. 12 Skenario Uji Coba.....	53
Tabel 3. 13 Jadwal Penelitian.....	58
Tabel 4. 1 Spesifikasi Perangkat Keras .....	59
Tabel 4. 2 Spesifikasi Perangkat Lunak .....	60
Tabel 4. 3 Detail Operasi Pada <i>Query</i> Salah .....	80
Tabel 4. 4 Hasil Nilai Presisi 100 <i>Query</i> .....	86
Tabel 4. 5 Hasil 100 <i>Query</i> Nilai Presisi <i>Search Engine</i> .....	94
Tabel 4. 6 Hasil Skenario Uji Coba .....	103



## DAFTAR PROGRAM

Kode Program 4. 1 Inisiasi Variabel Visualisasi Data dengan <i>WordCloud</i> .....	62
Kode Program 4. 2 <i>Cleaning</i> .....	64
Kode Program 4. 3 Inisiasi Fungsi Metode <i>Levenshtein Distance</i> .....	65
Kode Program 4. 4 Koreksi Ejaan .....	67
Kode Program 4. 5 Modul <i>Part-Of-Speech (POS) Tagging</i> .....	68
Kode Program 4. 6 Inisiasi Fungsi Nilai Presisi .....	69
Kode Program 4. 7 Inisiasi Fungsi <i>TF-IDF</i> .....	70
Kode Program 4. 8 Inisiasi Fungsi <i>Inverted Index</i> .....	71
Kode Program 4. 9 Inisiasi Fungsi Pencarian Artikel Menggunakan <i>Cosine Similarity</i> .....	72
Kode Program 4. 10 Implementasi Sistem Koreksi Ejaan Menggunakan Algoritma <i>Levenshtein Distance</i> .....	74
Kode Program 4. 11 Koreksi Ejaan Terhadap <i>query</i> dan <i>Part-Of-Speech (POS) Tagging</i> .....	75
Kode Program 4. 12 <i>query</i> Pencarian Artikel Pariwisata <i>Levenshtein Distance</i> dan <i>Part-Of-Speech (POS) Tagging</i> .....	75
Kode Program 4. 13 Inisiasi Judul dan Konten Artikel menggunakan <i>Part-Of-Speech (POS) Tagging</i> .....	78
Kode Program 4. 14 Menampilkan <i>Part-Of-Speech (POS) Tagging</i> pada Judul, tanggal dan Konten Artikel .....	79

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Bahasa Indonesia memiliki aturan kata dengan kompleks, sehingga kesalahan kata menjadi hal umum terjadi ketika *user* atau pengguna mengetik dan membuat *query* pencarian. Bahasa Indonesia berfungsi sebagai alat komunikasi dalam menyalurkan informasi, perasaan, sikap, gagasan, emosi[1]. Bahasa juga dapat digunakan sebagai sumber perspektif yang dicatat dalam bentuk laporan dokumen, dan menyampaikan serta mencari data informasi[2]. Ketika mencari informasi, baik informasi yang didapatkan melalui radio, televisi, koran, maupun media informasi lainnya. Hal ini dengan mudah semuanya tersedia di dalam internet. Ketika mencari informasi dan mengetik *query* (kata kunci yang terdiri dari satu atau beberapa kata kunci) di internet, nantinya akan muncul sebuah informasi pembetulan kata dan informasi sesuai kata kunci (*query*) yang dicari. *Spelling correction* merupakan salah satu poin utama yang akan dibahas pada topik penelitian kali ini, dimana informasi yang dicari berupa informasi mengenai pariwisata, yang berkaitan dengan artikel berita pariwisata. Berita merupakan informasi terkini dalam suatu peristiwa maupun fakta yang terjadi. Pemanfaatan media massa seperti berita ini, digunakan untuk menyampaikan sebuah pesan[3]. Berita *online* terbagi dalam beberapa bagian salah satunya artikel yang dimuat untuk menyampaikan informasi, diantaranya berita pariwisata[3].

Menurut [4] pariwisata dapat diartikan sebagai perjalanan dari suatu tempat ke tempat lain yang dilakukan oleh perorangan maupun kelompok yang dijadikan untuk usaha mencari sebuah keseimbangan maupun keserasian baik dalam kebahagiaan dengan lingkungan sekitar yang berdimensi sosial, budaya, maupun alam. Terdapat banyak kegiatan yang berkaitan dengan pariwisata, diantaranya perhotelan, kerajinan atau cinderamata, destinasi, dan *event*. Saat ini artikel berita sudah dapat diakses secara *online*, melalui *website*. Salah satu *website* yang berada di laman detik.com, dengan tampilannya dan memberikan informasi berita yang menarik terkait berita pariwisata[3]. Dalam hal ini, hanya dengan menginputkan suatu kata kunci (*query*) yang ada dalam mesin pencarian (*search engine*) bahasa

Indonesia (SEBI) nantinya sistem akan memberikan informasi berdasarkan *query* yang diberikan[5]. Pada saat ini mulai banyak bermunculan pencarian yang baru dengan berbagai kebutuhan sendiri, diantaranya *Search Engine* Bahasa Indonesia (SEBI). SEBI dilakukan pada tahun 2017 yang berfungsi untuk mengumpulkan halaman *website* dari internet, terdapat *preprocessing*, untuk membangun indeks dan menangani *query* serta mengkategorikan dokumen [6]. Namun seringkali hasil yang diperoleh *user* (pengguna) tidak sesuai dengan *output* yang diharapkan, karena adanya kesalahan pengetikan dalam memberikan sebuah kata kunci (*query*) yang tidak termasuk kata dalam kamus atau terdapat kesalahan dalam mengetik (*typo*)[7]. Ketika pengetikan yang dilakukan pada saat menulis dokumen terdapat tipografi, yang nantinya dapat membuat arti ataupun maksud yang berbeda[2]. Ketika melakukan koreksi ejaan kata pada *query*, kemudian menemukan kata yang salah sehingga perlu dilakukan pencarian kemungkinan kata yang tepat atau sesuai[8]. Pemeriksaan kesalahan yang bukan kata, dimana proses pemeriksaan ejaan kata pada *query* berfokus dalam penanganan kata yang mengalami kesalahan ejaan yang diakibatkan adanya kesalahan tipografi[9]. Adanya kesalahan kata dapat menghambat pencarian informasi yang akurat, terutama ketika mesin pencari tidak dapat memahami kata-kata yang salah ejaan. Maka dari itu terdapat satu topik untuk mengatasi permasalahan kesalahan ejaan kata dalam pencarian informasi dengan menggunakan *query* yaitu *spelling correction* (koreksi ejaan).

*Spelling correction* atau koreksi kesalahan ejaan merupakan sebuah fitur koreksi otomatis yang digunakan untuk melakukan sebuah proses pendeteksian dalam kesalahan ejaan kata dan juga pemberian sebuah saran kata dalam kesalahan ejaan pada teks[10]. Pada bagian ini, terdapat dua jenis dalam proses pemeriksaan ejaan diantaranya: pemeriksaan kesalahan yang bukan kata dan pemeriksaan kesalahan kata yang sebenarnya. Dalam pemeriksaan kesalahan bukan sebuah kata berfokus dalam penanganan kata yang salah ejaan, disebabkan oleh kesalahan tipografi. Sedangkan proses pemeriksaan kesalahan kata yang sebenarnya ditekankan dalam penanganan kesalahan pada kata yang ada pada kalimat[9]. Berdasarkan permasalahan diatas maka diperlukan sebuah metode untuk koreksi kata pada *query*. Metode *Levenshtein Distance* adalah metode pencarian solusi dalam bidang komputasi bahasa alami, dengan menentukan suatu matriks untuk

mengukur jumlah perbedaan antara dua *string*[8]. *Levenshtein Distance* merupakan metode dimana terdapat matriks untuk menghitung jumlah perbedaan yang terdapat pada *string input* (s) dan *string target* (t). Contohnya, apabila terdapat *string* sumber (s) yaitu “tihin” dan *string target* (t) “tahun”, maka akan dilakukan proses *Levenshtein Distance* dengan menghasilkan jarak antara dua *string* yaitu 1. Maka proses tersebut merupakan sebuah operasi pergantian atau disebut dengan substitusi[11].

Algoritma *Levenshtein Distance* memiliki salah satu keunggulan untuk bekerja dengan cara mengubah *source string* menjadi *target string*, dengan melakukan operasi yang meliputi penyisipan, penghapusan, dan penggantian dari suatu karakter. Dalam menghitung jarak dari perbedaan antara dua *string* yaitu dengan menentukan minimum jumlah operasi perubahan yang mana dari *string A* menjadi *string B*. Sehingga algoritma ini sesuai untuk mengoreksi perbedaan antara 2 *string* [12]. Pada tahun 2023, terdapat aplikasi yaitu pengarsipan dan pencarian dari sebuah data anggota yang terdapat di megapro club Indonesia, dimana peneliti menunjukkan *Levenshtein Distance* lebih baik dari metode *String Matching Knuth Morris Pratt*, hasil kecepatan dan ketepatannya yaitu terdapat data anggota 0,02 untuk rata-rata pencarian algoritma *Levenshtein Distance* dan 0,022 untuk rata-rata pencarian algoritma *knuth morris pratt* [13]. Dalam interaksi antara komputer dengan manusia, terdapat teori yang membahas terkait pemrosesan bahasa alami atau biasa disebut dengan *Natural Language Processing* (NLP), dimana NLP berperan untuk permasalahan terkait ambiguitas kata dengan teknik pelabelan kelas kata menggunakan *Part-of-Speech* (POS) *Tagging* metode *Rule based*. *Part-of-Speech* (POS) *Tagging* salah satu tahapan yang terdapat dalam teori NLP untuk mengetahui ambiguitas kata dan melakukan kelas kata[1]. Data *corpus* yang digunakan untuk melakukan proses *Part-of-Speech* (POS) *Tagging* pada penelitian ini yaitu *corpus* bahasa Indonesia. Oleh sebab itu adanya proses penandaan dengan menggunakan *Part-of-Speech* (POS) *Tagging* diterapkan secara otomatis dengan menggunakan berbagai metode di dalamnya. Dalam penelitian [14] yang dilakukan tahun 2017 mengenai *Part-of-Speech* (POS) *Tagging* menggunakan metode HMM dan *Rule Based*, dimana dalam penelitian ini berfokus pada pelabelan kelas kata dan penggunaan *corpus*

Pemilihan metode *Rule Based*, dalam penelitian ini, memiliki kelebihan yaitu metode ini salah satu metode yang mempunyai algoritma *Neuro Linguistic Programming* (NLP) yang mempunyai pendekatan berbasis aturan yang lebih baik dalam pemberian *tag* kata yang digunakan dalam morfologi bahasa[15]. Pada penelitian sebelumnya yang sama-sama menggunakan metode *Rule Based* menghasilkan performa yang cukup baik dengan hasil yaitu 87,4%, tetapi perlu melakukan pemeriksaan kata morfologi dan kata *slang*, yang nantinya menghasilkan istilah khusus pada *corpus*[16]. Beberapa penelitian yang membahas mengenai pemeriksaan ejaan kata pada *query* lebih banyak membahas mengenai kesalahan kata yang dimana disebabkan oleh kesalahan tipografi. Penelitian [17] pada tahun 2017, dengan menggunakan metode pendekatan kamus berbasis *Levenshtein Distance*, dapat disimpulkan bahwa aplikasi hanya dapat melakukan koreksi dokumen sebanyak 6 halaman, dimana pengujian kesalahan ejaan kata dilakukan dengan memasukkan kata yang salah. Hasil yang diperoleh menunjukkan saran kata yang diprioritaskan yaitu kata yang benar, dan hasil pengujian kesalahan kata mempunyai akurasi sebesar 86%. Kemudian hasil pengujian terhadap struktur bahasa Indonesia dilakukan dengan memberikan kalimat dan *output* sistem pada penguji sebanyak 30 kalimat dan hasil nilai akurasi yang diperoleh 76,66%. Dan penelitian[18] untuk pencarian kamus obat (*drugs e-dictionary*) menggunakan Algoritma *Levenshtein Distance*, hasil akurasi yang diperoleh dari pengujian implementasi algoritma pada modul *drugs e-dictionary* dihasilkan akurasi, *recall* dan *precision* sebesar 90%.

Dengan beberapa alasan yang telah disebutkan, maka akan dibuat sebuah sistem untuk *spelling correction* atau koreksi kesalahan ejaan terhadap *query* pada pencarian artikel berita pariwisata berbahasa Indonesia dengan menggunakan *Part-of-Speech* (POS) *Tagging* metode *Rule Based* untuk menganalisis ambiguitas kata sehingga dapat melakukan pemberian label pada kelas kata menggunakan metode *Rule Based*. Untuk mengatasi permasalahan yang ada pada penelitian *spelling correction* dalam kinerja *search engine* bahasa Indonesia dengan ambiguitas kata dan pelabelan kelas kata, diusulkan penggunaan Algoritma *Levenshtein Distance* dengan *Part-of-Speech* (POS) *Tagging* metode *Rule Based* pada artikel berita pariwisata.

## **1.2 Rumusan Masalah**

Rumusan masalah berisi permasalahan, solusi dari permasalahan, dan pertanyaan pada penelitian.

### **1.2.1 Permasalahan**

Terjadinya *error spelling correction* atau kesalahan ejaan pada kata-kata didalam kata kunci (*query*) pada suatu *search engine* bahasa Indonesia (SEBI) yang mengakibatkan hasil koreksi ejaan kata pada *query* tidak sesuai dengan kebutuhan informasi pengguna, dan kekurangan dari metode ini yaitu tidak mampu mengatasi kata yang bermakna banyak (ambiguitas kata) sehingga diperlukan metode untuk menyelesaikan permasalahan tersebut dengan menggunakan metode *Levenshtein Distance* dalam menyelesaikan permasalahan *error spelling correction* dan menerapkan *Part-of-Speech* (POS) *Tagging* untuk ambiguitas kata.

### **1.2.2 Metode Usulan**

Metode usulan yang akan digunakan dalam penelitian ini untuk mengatasi masalah koreksi ejaan pada *query* adalah menggunakan algoritma *Levenshtein Distance* dan *Part-of-Speech* (POS) *Tagging* dalam artikel berita pariwisata berbahasa Indonesia.

### **1.2.3 Pertanyaan Penelitian**

Pada uraian yang dijelaskan sebelumnya didapatkan pertanyaan penelitian sebagai berikut:

Berapa nilai *presisi* yang dihasilkan dalam penerapan koreksi kesalahan ejaan menggunakan algoritma *Levenshtein Distance* dan *Part-of-Speech* (POS) *Tagging*?

## **1.3 Tujuan dan Manfaat Penelitian**

Tujuan dan Manfaat dari Penelitian ini adalah sebagai berikut :

### **1.3.1 Tujuan Penelitian**

Tujuan perancangan identifikasi salah ketik dan perbaikan kata pada mesin pencarian berita pariwisata adalah untuk mengetahui tingkat akurasi yang meliputi *presisi* dari koreksi kesalahan ejaan dengan menggunakan metode *Levenshtein Distance* dengan *Part-of-Speech* (POS) *Tagging*.

### 1.3.2 Manfaat Penelitian

Manfaat dari penelitian yang dilakukan yaitu :

1. Mengetahui koreksi kesalahan ejaan dari *query* (yang terdiri dari satu atau beberapa kata kunci) yang diberikan menggunakan algoritma *Levenshtein Distance*, dan mengetahui adanya kata ambigu atau ambiguitas kata dengan melakukan pemberian kelas kata menggunakan *Part-of-Speech (POS) Tagging*
2. Hasil dari penelitian dapat dijadikan untuk rujukan pada penelitian selanjutnya dalam topik yang berkaitan dengan koreksi ejaan (*spelling correction*).

### 1.4 Batasan Masalah

Pada batasan masalah yang ada dalam batasan ini diperlukan agar dalam pengerjaannya tidak melebar dari sasaran yang diteliti. Adapun batasan permasalahan agar tidak meluas pada topik pembahasan penelitian ini, yaitu antara lain sebagai berikut::

1. Penelitian ini menggunakan data berita pariwisata berbahasa Indonesia.
2. Menggunakan data korpus bahasa Indonesia yang diperoleh dari Fakultas Ilmu Komputer Universitas Indonesia.
3. *Output* berupa membenarkan kata dari kata kunci (*query*) menggunakan algoritma *Levenshtein Distance* dan informasi mengenai berita pariwisata dalam data *input* pada mesin pencarian (SEBI) dan saran perbaikan kata yang telah disesuaikan pada kamus, dengan memperhatikan apabila terdapat ambiguitas kata yang nantinya dilakukan pelabelan kelas kata (pemberian tag pada *query*) menggunakan *Part-of-Speech (POS) Tagging* metode *Rule Based*.

### 1.5 Sistematika Penulisan

Sistematika yang diterapkan dalam penyusunan laporan tugas akhir ini digunakan untuk memberikan gambaran umum mengenai apa saja yang akan dibahas pada penelitian ini, sehingga secara sekilas pembaca akan mengetahui garis

besar dan poin-poin penting dalam penelitian ini. Sistematika terbagi dalam beberapa pokok bahasan sebagai berikut:

## **BAB I PENDAHULUAN**

Bab 1 ini menguraikan latar belakang, rumusan masalah, tujuan dan manfaat dari penelitian, batasan masalah, dan sistematika penulisan laporan penelitian dan jadwal penelitian.

## **BAB II KAJIAN PUSTAKA**

Bab 2 ini menjelaskan teori yang sudah ada atau dari penelitian-penelitian yang telah dilakukan dengan tujuan untuk mendukung serta mendasari dari sebuah sistem yang akan dibangun. Dasar teori sangat dibutuhkan agar tercapainya suatu tujuan sistem yang akan dibuat dan belajar dari kekurangan penelitian sebelumnya. Dasar teori dan konsep yang digunakan dalam penelitian ini adalah *spelling correction*, algoritma *Levenshtein Distance*, *Part-of-Speech (POS) Tagging* dengan metode *Rule Based*, *text preprocessing*, *search engine*, *kinerja search engine*, *precision* dan penelitian terkait.

## **BAB III METODE USULAN**

Bab 3 ini menyajikan garis besar mengenai metode yang digunakan dalam penelitian, dan analisa kebutuhan serta perancangan. Dimulai dari *dataset*, arsitektur sistem. Skenario pengujian, perhitungan metode, perhitungan, dan tahapan penelitian.

## **BAB IV METODE HASIL DAN PEMBAHASAN**

Bab 4 ini membahas sekaligus menjelaskan mengenai lingkungan uji coba, bahasan dan hasil dari implementasi sistem berdasarkan rancangan yang telah dibahas di bab 3 sebelumnya.

## **BAB V PENUTUP**

Bab 5 ini berisi terkait kesimpulan akhir dari penelitian yang telah dilakukan dan sara dari metode yang digunakan.



## BAB II

### KAJIAN PUSTAKA

#### 2.1 *Spelling Correction* (Koreksi Ejaan)

*Spelling correction* adalah cara paling umum untuk mengidentifikasi dan memberikan saran atau ide pada kata-kata yang salah ejaannya dalam sebuah teks. Koreksi ejaan adalah sebuah sistem yang digunakan untuk dapat melakukan pengoreksian kesalahan dalam ejaan. Kesalahan yang sering terjadi yaitu adanya karakter yang tidak sesuai, misalnya terdapat karakter yang berubah, atau kurangnya karakter dan terdapat penggantian karakter [8]. Misalnya dalam sebuah kata yang salah yaitu “Keuar”, kemungkinan kata yang benar adalah “Keluar”, penyebab kesalahan katanya adalah kurang huruf “l”.

Sedangkan *spelling corrector* merupakan fitur atau aplikasi yang akan melakukan proses tersebut. Fitur ini mencari kata-kata yang salah berdasarkan data korpus yang digunakan aplikasi[9]. Selain itu, saran kata diberikan dengan perhitungan algoritma yang digunakan oleh aplikasi. Pemeriksa ejaan terbagi menjadi dua jenis, yaitu: pemeriksa kesalahan yang bukan kata dan pemeriksa kesalahan kata yang sebenarnya. Pemeriksa kesalahan yang bukan kata berfokus pada penanganan kata yang salah ejaan yang disebabkan oleh kesalahan tipografi. Sedangkan pemeriksa kesalahan kata yang sebenarnya ditekankan pada penanganan kesalahan penempatan kata dalam sebuah kalimat.

Sistem koreksi kesalahan ejaan otomatis merupakan sebuah sistem yang mempunyai tujuan untuk memverifikasikan serta memperbaiki sebuah kesalahan kata dengan menggunakan kumpulan kata-kata yang disarankan[8]. Selain itu penulis di Kukich, Toutabaris ke va dan Moore, serta Pirinen dan linden membagi kesalahan ejaan menjadi dua kategori menurut penyebabnya[19]:

1. Kesalahan *kognitif* (disebut juga *otografik* atau konsisten): Kesalahan ini disebabkan oleh kecacatan dalam menulis teks. Cara atau proses penulisannya yang benar tidak diketahui oleh penulisnya. Hal ini penulis bisa saja menderita *disleksia*, *disgrafia*, ataupun masalah *kognitif* lainnya. Hal ini, orang yang menulis teks hanya sedang mempelajari bahasanya, namun tidak dengan mengetahui bagaimana ejaan yang benar.

2. Kesalahan ketik (disebut kesalahan konvensional), biasanya berkaitan dengan keterbatasan teknik perangkat *input* yang dimana *keyboard* atau sistem OCR yang bergantung terhadap lingkungan, karena mengetik dengan tergesa-gesa sering kali menyebabkan penggantian dua tombol tutup. Kesalahan ketik yang disebabkan dengan pengetikan yang tergesa-gesa hal ini bersifat *agnostic* bahasa atau tidak berhubungan dengan bahasa penulis.

### 2.1.1 Ejaan

Ejaan merupakan keseluruhan dari peraturan tentang bagaimana menggambarkan berbagai penulisan dan bagaimana interaksinya dalam sebuah bahasa. Adanya penggunaan ejaan mempunyai tujuan agar bahasa dapat dengan baik di pahami sehingga tidak terjadi kesalahpahaman makna bahasa yang diungkapkan. Terdapat Buku Pedoman Umum Ejaan Bahasa Indonesia, yang memuat empat hal utama yang menjadi aspek dalam kajian penulisan yang dibuat atau dibahas meliputi: pemakaian huruf, penulisan kata, pemakaian tanda baca dan penulisan unsur serapan. Dalam aspek - aspek ini mempunyai beberapa pokok pembahasan dalam kajian pada penelitian ini, diantaranya sebagai berikut[20] :

#### a) Pemakaian Huruf

Pemakaian huruf yang terdapat dalam sebuah buku Ejaan Bahasa Indonesia meliputi : huruf abjad, huruf vokal, huruf konsonan, gabungan huruf konsonan dan huruf kapital.

#### b) Penulisan Kata

Penulisan kata dalam sebuah ejaan Bahasa Indonesia diantaranya, meliputi: tanda titik, koma, titik koma, titik dua, tanda hubung, tanda tanya, tanda seru, dan tanda petik.

#### c) Pemakaian Tanda Baca

Pemakaian tanda baca dalam sebuah buku Ejaan Bahasa Indonesia, diantaranya mencakup sebuah tanda titik, koma, titik koma, titik dua, tanda hubung, tanda kurung siku, garis miring, serta tanda penyingkat.

#### d) Penulisan Unsur Kata Serapan

Penulisan unsur kata serapan dalam bahasa Indonesia meliputi sebuah bahasa yang baik, bahasa daerah yang dimana terdapat bahasa asing. Dalam buku ejaan Bahasa Indonesia, sebuah penulisan unsur kata dalam serapan dibagi menjadi

dua, diantaranya penulisan kata serapan yang berfungsi untuk melakukan unsur kata asing.

### **2.1.2 *Typographical Error***

*Typographical error* adalah suatu kesalahan dalam mengetik teks yang mengakibatkan perubahan arti dalam suatu kata maupun kalimat. Hal ini terjadi karena adanya ketidaktahuan penulis atau pengguna, dan kegagalan dalam mekanisme yaitu jari yang salah menekan tombol, dimana ketika menyebabkan kesalahan dalam proses pengetikan atau adanya *typographical error*. Pada bagian ini terdapat dua jenis kesalahan atau adanya *typographical error* yaitu sebagai berikut.

1. *Non-word error* : yaitu terjadinya kesalahan (*error*) dimana, tidak memiliki makna didalamnya. Misalnya kesalahan dari kalimat “Dia memberikan suatu commet yang sangat berguna”, setelah dilakukan koreksi maka hasilnya “Dia memberikan komentar yang sangat berguna”
2. *Realword error* : adalah suatu kata yang tertulis dan bernilai benar serta mempunyai sebuah arti di dalam kamus, akan tetapi tidak dimasukkan dalam kalimat dan memiliki arti yang berbeda, bahkan terdapat tata bahasa yang keliru[11].

## **2.2 *Search Engine***

*Search engine* adalah alat yang dikembangkan untuk sistem komputer, khususnya internet, untuk menemukan contoh kata atau frasa yang dapat ditemukan dalam dokumen yang mencakup dalam ruang lingkup alat tersebut[21]. Pengguna dapat melakukan pencarian yang ada di halaman web yang diperlukan melalui *search engine*. Mesin pencarian (*search engine*) merupakan salah satu mesin yang ulet dan teliti, dengan adanya eksplorasi dalam memberikan informasi yang sesuai dengan permintaan pengguna tanpa memandang kapan dan dimana waktu itu dilakukan. *Search engine* adalah sebuah *website* yang digunakan untuk mencari informasi yang ada di dalam sebuah layanan *World Wide Web* (www), file transfer *protocol* (ftp), *mailing list*. Dimana hasil dari pencarian ini nantinya akan menampilkan banyak data informasi yang berasal dari *website* sebagai penyedia informasi. Terdapat beberapa contoh dari mesin pencarian (*search engine*)

diantaranya: Google (<http://www.google.com/>), Yahoo (<http://www.yahoo.com/>), Amazone (<http://www.amazon.com/>)[22]. *Search engine* atau biasa disebut dengan mesin pencari merupakan salah satu teknik dari temu kembali informasi yang dimana menentukan dalam menemukan sebuah dokumen dan melakukan untuk mengeksekusi algoritma peringkat dan menemukan dokumen. [23]. Sebuah mesin pencarian (*search engine*) digunakan oleh pengguna internet, dalam mencari informasi. Hal ini menyebabkan banyaknya dokumen ketika disimpan dalam digital melonjak (penuh), dan mengakibatkan pengguna (*user*) mengalami kesulitan dalam menemukan artikel yang sesuai. Cara menggunakan mesin pencari yaitu dengan memasukkan kata kunci (*query*) yang ingin dicari, kemudian akan ditampilkan beberapa tautan yang mengarah ke situs atau informasi yang saling berkaitan dengan kata kunci yang dimasukkan[24].

### **2.2.1 Information Retrieval (IR)**

Proses untuk mendapatkan sumber informasi atau mengesktraksi sumber informasi biasanya dokumen, dari sejumlah data besar biasanya teks, untuk memenuhi kebutuhan informasi disebut dengan *information retrieval* (IR) [11]. Sistem temu kembali informasi atau biasa disebut dengan *information retrieval* berguna untuk memperoleh informasi yang sesuai atau diinginkan oleh *user* (pengguna) ketika menginputkan *query* untuk mendapatkan informasi. *query* yang telah dimasukkan pengguna disini yaitu dokumen yang dicari[24]. Terdapat beberapa fungsi *Information Retrieval* yaitu sebagai berikut :

1. Untuk mengidentifikasi sumber informasi yang sesuai dengan target pengguna.
2. Digunakan untuk melakukan analisis dari sumber dokumen.
3. Melakukan atau mempresentasikan isi dari sumber yang dianalisis sehingga nantinya akan sesuai dalam *query* yang pengguna lakukan.
4. Melakukan analisis dari *query* pengguna dalam merepresentasikan sebuah *query* yang sama dengan database
5. Melakukan pencocokan langkah-langkah dalam pencarian yang digunakan untuk menyimpan didalam *database*.
6. Melakukan proses agar memperoleh sebuah informasi yang sesuai.

7. Melakukan pembuatan dalam pengaturan yang nantinya diperoleh dalam membuat sebuah informasi yang diciptakan didalam sistem berdasarkan kebutuhan yang didapat kembali dari pengguna.

Dalam *information retrieval* ini nantinya sebuah *query*, bukan hanya dilakukan untuk memperbaiki sebuah objek unik yang ada dalam kumpulan data, akan tetapi dalam sebuah *query* akan dapat menyesuaikan dalam berbagai objek yang berbeda dengan derajat kesamaan yang berbeda pula. Hal ini nantinya suatu objek akan melakukan satu *entitas*, sehingga merepresentasikan sebuah informasi yang berada dalam kumpulan data maupun *database*. Sehingga dalam sistem yang ada di *Information Retrieval* pada umumnya diterapkan dalam bentuk peringkat[25]. Sistem ini juga melakukan komputasi untuk menghitung mengenai kecocokan dalam setiap objek yang nanti didalamnya terdapat *database* dan *query*, dan menciptakan suatu sistem pemeringkat agar menghasilkan sebuah hasil berdasarkan perhitungan dari hasil yang diperoleh. Hasil dari peringkat yang paling tertinggi nantinya akan direpresentasikan pada pengguna dan dikatakan sebagai hasil yang paling sesuai. Sehingga diperlukan sebuah mesin pencari (*search engine*) yang mempunyai tujuan untuk memperoleh dokumen yang akan dicari [24].

### 2.2.2 Query

Dalam *search engine* mempunyai beberapa komponen penting yang membahas dokumen dan juga *query* didalamnya[25].

1. *Query interface* merupakan sebuah komponen yang terdapat dalam *search engine* dan tampilan maupun format yang menyediakan sebuah fasilitas dalam mesin pencarian (*search engine*).
2. *Query engine* sebuah program yang bertugas sebagai penerjemah dalam memenuhi keinginan *user* atau kata yang diketikkan dalam sebuah bahasa yang mudah dipahami oleh mesin komputer. Proses dalam *query* ini melakukan sebuah pengarsipan atau pencarian arsip dan dokumen yang sesuai dalam basis data.
3. *Database* merupakan gabungan dokumen yang diarsip dalam sebuah *website* yang ada pada internet. Semakin besar jumlah skala internet semakin besar kapasitas dalam penyimpanannya.

4. *Spider* merupakan proses pendataan dari sebuah *website* yang ada dalam internet atau disebut *web crawlers*.
5. *Indexer* ialah program yang digunakan untuk mempercepat dalam proses pencarian yang dimana pemanfaatan *index* dalam buku. Dan juga adanya teknik untuk melakukan penerapan *index* yang berguna untuk mendapatkan kecepatan dalam proses pencarian data informasi.

### 2.3 Artikel Berita Pariwisata

Berita merupakan informasi terkini dalam suatu peristiwa maupun fakta yang terjadi. Berita dibuat atau ditulis oleh seorang jurnalistik atau wartawan yang mengumpulkan sebuah fakta di lapangan melalui proses jurnalistik. Berita menjadi salah satu peranan yang sangat penting dalam masyarakat karena dengan adanya berita, dapat memberikan sebuah informasi. Sebagian berita pada umumnya hanya dibuat dalam versi cetaknya. Dan bahkan dapat melihat berita *online* yang sampai saat ini sudah berkembang sangat luas dan mudah diakses dengan jaringan internet. Berita yang terdapat dalam sebuah media massa menjadi suatu cara dalam menciptakan suatu realitas yang diinginkan mengenai peristiwa atau orang yang dilaporkan. Berita berisi mengenai informasi-informasi terbaru yang berada disekitar dan biasanya disajikan dalam bentuk tulisan. Berita *online* banyak terbagi dalam berbagai berita, salah satunya artikel yang dibuat atau dimuat untuk menyampaikan informasi, mengenai pariwisata, tempat pariwisata yang sesuai. Biasanya informasi ini didapatkan dalam detik *news* dan trimbun *news*[3].

Industri perjalanan wisata memiliki definisi yang berbeda-beda menurut berbagai sudut pandang, khususnya sebagaimana ditunjukkan dalam Peraturan No.10 Tahun 2009, industri perjalanan wisata dicirikan sebagai semacam pergerakan kawasan lokal yang berisi komponen-komponen industri perjalanan dengan kantor dan administrasi berbeda yang diberikan oleh jaringan tertentu, *visioner* bisnis, otoritas publik, dan negara-negara terdekat. Secara khusus, industri perjalanan disebut sebagai industri dengan organisasi yang terkait dengan pengaturan tenaga kerja dan produk untuk mengatasi masalah wisatawan yang mengunjungi kawasan wisata. Kemajuan industri perjalanan mempunyai potensi kemajuan yang luar biasa. Hal ini menjadi tujuan untuk memperoleh modal dengan

adanya industri pariwisata baik dalam segi ekonomi mengenai kesadaran dan tanggung jawab atas lingkungan, persaingan bisnis dan produk *kredibel* dan daya saing yang menciptakan sebuah nilai rantai usaha dalam beragam jenis sistem yang ada[4].

### **2.3.1 Data artikel berita pariwisata**

Data artikel berita yang digunakan dalam penelitian ini adalah data artikel berita pariwisata yang diperoleh dari laman *website* detik.com dengan data yang diperoleh berjumlah 332 dokumen yang, dengan artikel berita yang ada. Selanjutnya nantinya akan diolah untuk mempermudah dalam proses selanjutnya, dalam mengolah data artikel berita pariwisatanya ini nantinya akan dilakukan tahapan *preprocessing* mulai dari *case folding*, *cleaning*, *tokenization*, *stopword removal* dan *stemming* yang nantinya mempermudah dalam mengolah *term* (kata) dalam setiap dokumen yang ada didalam artikel berita pariwisata.

Kemudian nantinya data artikel berita pariwisata ini akan dilakukan proses *Part-of-Speech* (POS) *Tagging* dimana data artikel berita pariwisata ini akan dilakukan pengecekan apakah terdapat kata ambigu atau tidak, dengan teknik pelabelan kelas kata. Tujuan dari adanya pemberian tag atau pelabelan kelas kata, untuk mengetahui setiap kata yang ada dalam dokumen terdapat kata ambigu, sehingga nantinya dari kata ambigu ini dapat dilakukan pemberian tag, yang akan menjelaskan makna dari setiap kata yang ada dalam artikel berita pariwisata yang ditampilkan dari hasil *inputan query* yang sebelumnya sudah diperbaiki dan diberikan pelabelan kelas kata dan tag.

Hasil dari artikel berita pariwisata yang sudah diolah dan dilakukan pelabelan kelas kata atau tag dari artikel berita pariwisata yang ada, akan dilakukan tahapan *inverted index*, berfungsi untuk mengurutkan artikel dokumen yang mempunyai bobot, sehingga nantinya mempercepat dalam proses pencarian dokumen dari dokumen yang mempunyai bobot paling tinggi.

### **2.3.2 Inverted Index**

*Inverted index* merupakan salah satu struktur data *index* yang digunakan untuk memudahkan dalam proses pencarian dari *term* yang berbeda untuk mendapatkan

suatu daftar *term* dalam sebuah dokumen. Tujuan adanya *inverted index* ini untuk meningkatkan kecepatan dan efisiensi dalam melakukan pencarian dari sekumpulan dokumen yang mengandung *query* (kata kunci) yang diinputkan. *Inverted index* berisi daftar dokumen yang diurutkan, dimana pada setiap dokumen yang sudah diurutkan tersebut, mempunyai bobot dari setiap kata (*term*). Sehingga untuk melakukan *inverted index* pada sistem ini menggunakan TF-IDF.

## 2.4 Text Preprocessing

*Text Preprocessing* adalah sebuah cara dalam memproses untuk melakukan pengolahan terhadap suatu data mentah menjadi data yang siap digunakan. Pada *preprocessing* ini dilakukan ketika *search engine* menerima permintaan dari *user*, contohnya ketika mengetikkan kata kunci (*query*) “Karnaval” pada mesin pencarian, maka nantinya *search engine* atau mesin pencarian SEBI akan melakukan pencarian, dan apabila kata kunci (*query*) terdapat kesalahan, nantinya *spelling correction* akan melakukan pengecekan dengan membernarkan kata yang ada. Tujuan dilakukan proses *preprocessing* agar nantinya data lebih mudah diproses oleh sistem yang nantinya akan menghasilkan sebuah data *term* (data yang melalui proses *preprocessing*)[8].

*Preprocessing* data pada teks terdiri dari beberapa proses tahapan diantaranya sebagai berikut:

1. *Case Folding*: berfungsi dalam mengubah huruf pada teks dari huruf kapital (*upper case*) menjadi huruf kecil (*lower case*).
2. *Cleaning*: digunakan untuk membersihkan sebuah karakter tertentu yang tidak dibutuhkan seperti simbol dan angka.
3. *Tokenizing*: berfungsi dalam menguraikan sebuah kalimat yang menjadi kata perkata, berdasarkan karakter ‘spasi’ sebagai tanda pemisahannya.

## 2.5 Part-of-Speech (POS) Tagging Bahasa Indonesia

*Natural Language Processing* ialah cabang ilmu komputer dan *linguistic* yang membahas mengenai bagian dalam pemrosesan bahasa alami. NLP mempunyai peran untuk mengatasi adanya ambiguitas kata pada teks berbahasa Indonesia. *Part-of-Speech (POS) Tagging Bahasa Indonesia* adalah sebuah proses penandaan kelas



pada setiap kata berdasarkan *corpus* bahasa Indonesia. Tahapan NLP yang berguna untuk menangani ambiguitas kata yaitu *Part-of-Speech (POS) Tagging*. Dimana tahapan dalam *Part-of-Speech (POS) Tagging* yaitu untuk menentukan kelas kata, dan hasilnya pada dokumen dapat digunakan sebagai dasar penelitian yaitu *Natural Languages Processing* pada dokumen, *machine translation*, *information retrieval*, *text summarization*, dan *language generator*[1]. *Part-of-Speech (POS) Tagging* adalah proses memberi label pada setiap kata dalam kalimat dengan *Part-of-Speech (POS) Tagging* atau *tag* yang sesuai dengan kelas kata seperti kata kerja, kata keterangan, kata sifat, dan lainnya[1]. Diperlukan sebuah kamus atau *corpus* penggunaannya dalam menentukan kelas kata. Berikut merupakan kelas kata bahasa Indonesia sebagai berikut[14]:

a. Kata benda (*noun*)

Kata benda yaitu kata atau gagasan kata sebagai pernyataan tentang sesuatu yaitu nama seseorang, nama tempat, binatang, sifat, ide, dan perbuatan.

b. Kata kerja (*verb*)

Kata kerja ialah kata atau gagasan kata yang menerangkan atau menggambarkan sebuah kejadian, tingkah laku, perbuatan, peristiwa dan keadaan.

c. Kata sifat (*Adjective*)

Kata sifat adalah sebuah kata atau gagasan kata yang digunakan untuk menerangkan kata benda dengan menjelaskannya.

d. Kata keterangan (*Adverb*)

Kata keterangan adalah kata atau gagasan kata sebagai pembatas atau pemberi informasi lebih banyak tentang kata kerja.

e. Kata bilangan (*Numeral*)

Kata bilangan merupakan kata atau gagasan yang menunjukkan suatu bilangan.

f. Kata penghubung (*Conjunction*)

Kata penghubung adalah kata atau gagasan kata yang memperluas satuan kata dan sebagai penghubung dengan beberapa satuan kata bilangan yang lain.

g. Kata depan atau Preposisi (*Preposition*)

Kata depan adalah kata yang mempunyai posisi katanya ada di depan sebelum kata benda, kata kerja, dan kata keterangan lainnya.

h. Kata injeksi atau kata seru (*Interjection*)

Kata seru adalah kata atau gagasan kata yang menunjukkan ungkapan rasa hati atau perasaan seseorang. Misalnya, kagum, heran, sedih, dan sebagainya.

i. Kata ganti orang (*Probaris ke un*)

Kata ganti adalah kata yang digunakan untuk mengganti nama, seperti *firt person* yaitu kata ganti orang pertama, kata ganti orang kedua, dan kata ganti orang ketiga.

## 2.6 Metode *Part-of-Speech (POS) Tagging* dengan Metode *Rule Based*

*Rule Based* adalah salah satu algoritma NLP dengan menerapkan *rule* atau aturan bahasa (*grammar*) agar memperoleh hasil kelas kata dalam sebuah kalimat, dimana algoritma ini menggunakan aturan bahasa. Algoritma *Rule Based* mempunyai 2 arsitektur, diantaranya algoritma yang pertama yaitu metode *Rule Based* dengan menggunakan kamus yaitu melakukan penandaan kata yaitu kelas kata (*leksikon*). Tahapan yang kedua menerapkan *disambiguation rule* dengan manual, kemudian diproses yang nantinya menjadi satu kelas kata dalam setiap kata[14]. Kemudian dari perubahan kelas kata ini yaitu kelas kata pertama dengan kelas kata terakhir nantinya akan dilakukan pencocokan *rule (aixan)* yang terdapat dalam kamus aturan. Dalam susunan *rule* pada kalimat yang terdapat dalam kamus aturan, jadi nantinya sistem tersebut akan menampilkan kata beserta kelas kata sebagai *output*. Apabila dalam perbedaan yang terdapat dalam kelas kata ditemukan adanya kelas kata yang ada dalam kamus, kemudian sistem nantinya akan langsung memberikan peringatan atau tanda dalam setiap kata dan dalam kelas kata yang benar dari kelas kata yang di dapat, maka nantinya akan ditampilkan oleh sistem. Dan juga *corpus* adalah sebuah kumpulan teks yang tersusun secara sistematis, dan teks yang terdapat didalam *corpus* digunakan dalam situasi dan kehidupan nyata[14].

## 2.7 Pembobotan *Term Frequency Inverse Document Frequency (TF IDF)*

Pada pembobotan *term* yaitu dengan menggunakan frekuensi kemunculan *term* (kata)/*Term Frequency* atau *TF* yang berkaitan dengan suatu dokumen, dimana hal ini merupakan metode pembobotan paling sederhana. *Term Frequency* atau *TF* merupakan sebuah kuantitas dari *term* yang sering muncul dalam suatu

dokumen[26]. Pada proses untuk menghitung jumlah dari kemunculan (frekuensi) *term*  $t_i$  dalam setiap dokumen  $d_i$ [26].

$$W_{TF}(t_i, d_i) = f(t_i, d_i) \quad (2.1)$$

Dimana:

$W_{TF}(t_i, d_i)$  = nilai TF *term* ke I pada dokumen ke j

$f(t_i, d_i)$  = jumlah kemunculan dari *term* ke I pada dokumen ke j

*Inverse Document Frequency* (IDF) merupakan perhitungan untuk mengetahui seberapa besar pengaruh *term* di dalam sebuah dokumen terhadap dokumen lainnya. Pada sebuah dokumen ini nantinya akan mengandung *term* yang sangat bernilai yang nantinya sangat jarang sekali ada atau muncul[26].

$$W_{IDF} = 1 + \log \frac{D}{d(t_i)} \quad (2.2)$$

Dimana:

$W_{IDF}(t_i, d_i)$  = nilai IDF *term* ke I pada dokumen ke j

$d_i$  = jumlah dokumen yang mengandung *term* ke i

$D$  = jumlah dokumen

Rumus untuk menyatakan bobot ( $W_{tf-idf}$ ) dari dokumen yang diproses terhadap dokumen kunci adalah:

$$W_{tf-idf}(t_i, d_i) = W_{TF}(t_i, d_i) \times W_{IDF}(t_i, d_i) \quad (2.3)$$

Dimana :

$W_{tf-idf}(t_i, d_i)$  = nilai TF-IDF *term* ke I pada dokumen ke j

$W_{IDF}(t_i, d_i)$  = nilai IDF *term* ke I pada dokumen ke j

## 2.8 Cosine Similarity

Metode *cosine similarity* adalah sebuah metode yang dimana penggunaanya dalam menghitung tingkat kesamaan antara satu objek dengan objek yang lain [27].

Pada penggunaan *cosine similarity* ini mempunyai tujuan untuk membandingkan tingkat kecocokan antara dua objek, karena *cosinus*  $0^0$  adalah 1 dan kurang dari 1 ( $<1$ ) untuk nilai sudut yang lain. Maka suatu nilai kemiripan antara dua objek dikatakan mirip ketika nilai *cosinus* adalah 1. Ukuran kemiripan antara dua buah vektor pada suatu ruang dimensi didapatkan dari nilai *cosinus* sudut disebut *Cosine Similarity*. Ketika digunakan dalam ruang positif hasil dari *cosine similarity* dibatasi antara 0 dan 1.

*Cosine similarity* juga diterapkan dalam penentuan nilai kemiripan pada dua dokumen teks. Dimana dengan menggunakan sebuah parameter dari jumlah kata-kata pada dua dokumen teks yang nantinya membandingkan (misalnya D1 “Dokumen 1” dan D2 “Dokumen 2”). Berikut ini terdapat rumus umum dalam penerapan *Cosine Similarity* [26].

$$CosSim(d_i, d_j) \frac{t_i \cdot d_i}{|t_i| \cdot |d_i|} = \frac{\sum_{j=1}^t (q_{ij} \cdot d_{ij})}{\sqrt{\sum_{j=1}^t (q_{ij})^2 \cdot \sum_j (d_{ij})^2}} \quad (2.4)$$

Dimana :

$q_{ij}$  = bobot istilah j pada dokumen i = *TF-IDF*

$d_{ij}$  = bobot istilah j pada dokumen i = *TF-IDF*

## 2.9 Algoritma *Levenshtein Distance*

Pada algoritma ini merupakan sebuah algoritma yang digunakan untuk koreksi ejaan yaitu algoritma *Levenshtein Distance* yang ditemukan oleh seorang ilmuwan dari Rusia pada tahun 1965 yang bernama Vladimir Levenshtein, dimana algoritma ini merupakan algoritma yang dimana suatu matriks dapat mengukur perbedaan suatu matriks, yang digunakan untuk mengukur perbedaan antara dua *string*. *Levenshtein Distance* dua buah *string* merupakan jumlah dari minimum operasi yang dibutuhkan untuk mengubah sebuah *string* (*source string*) menjadi *string* yang lain (*string target*). Dalam algoritma *Levenshtein Distance* ini merupakan suatu operasi yang melibatkan *insertion* (penyisipan), *deletion* (penghapusan), *substitution* (penggantian) dari suatu karakter tunggal[12]. Berikut ini merupakan penjelasan dari operasi algoritma *Levenshtein Distance* sebagai berikut:

### 1. Operasi Penyisipan Karakter (*Insertion*)

*Insertion* merupakan sebuah operasi penyisipan karakter baru kedalam *string*. Dalam proses penyisipan karakter, dapat dilakukan di awal kalimat, maupun akhir kalimat.

### 2. Operasi Penghapusan Karakter (*Deletion*)

*Deletion* adalah operasi penghapusan karakter yang berlebihan dalam suatu karakter. Contohnya kata “matematikan” nantinya akan dilakukan penghapusan sebuah kata yaitu hasilnya menjadi “matematika”.

### 3. Operasi pengubahan Karakter (*Substitution*)

*Substitution* yaitu operasi yang melakukan pergantian dalam suatu karakter dengan karakter lainnya yang bernilai benar. Misalnya sebuah kata “yamg” akan diganti menjadi kata “yang” dalam operasi pengubahan sebuah karakter yang dimana pada operasi ini nantinya huruf “m” akan diganti menjadi “n”.

Perhitungan *Levenshtein Distance* ini didapat pada sebuah matriks yang digunakan dalam menghitung dari jumlah perbedaan yang terdapat dalam dua *string*. Pada perhitungan jarak antara dua *string* yang ditentukan dari jumlah minimum operasi pada perubahan untuk membuat sebuah *string* A menjadi *string* B[12]. Fungsi Algoritma *Levenshtein Distance* yaitu matriks 2 dimensi yang digunakan untuk perhitungan nilai jarak dalam *Levenshtein Distance*. Dengan isi di dalam nilainya yaitu matriks yang mempunyai jumlah operasi penghapusan, penyisipan, dan penggantian yang diperlukan dalam mengubah sebuah *string* target. Berikut ini rumus algoritma *Levenshtein Distance*[28].

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} lev_{a,b}(i-1,j) + 1 & \text{(penghapusan)} \\ lev_{a,b}(i,j-1) + 1 & \text{(Penyisipan)} \\ lev_{a,b}(i-1,j-1) + 1 & \text{if } a_i \neq b_j \text{ (substitusi)} \end{cases} & \text{otherwise} \end{cases} \quad (2.5)$$

$$lev_{a,b}(i,j) = \min lev_{a,b}(i-1,j) + 1 \text{ (Penghapusan)} \quad (2.6)$$

$$lev_{a,b}(i,j) = \min lev_{a,b}(i,j-1) + 1 \text{ (Penyisipan)} \quad (2.7)$$

$$lev_{a,b}(i,j) = \min lev_{a,b}(i-1,j-1) + 1, a_i \neq b_j \text{ (Substitusi)} \quad (2.8)$$

$$lev_{a,b}(i,j) = \min lev_{a,b}(i-1,j-1), a_i = b_j \text{ (Tidak ada perubahan)} \quad (2.9)$$

Keterangan:

$a$  = string Sumber

$i$  = *index* baris *string* sumber

$b$  = String Target

$j$  = *index* baris *string* target

Kondisi  $a_i \neq b_j$

$(a_i \neq b_j)$ = perlu menambahkan 1 (+1)

$(a_i = b_j)$ = tidak perlu menambahkan 1 (+1)

## 2.10 Evaluasi Sistem

### 2.10.1 Perhitungan *Precision*

Pada proses perhitungan untuk sebuah efektivitas yang diperlukan dalam temu kembali informasi (*information retrieval*), dengan melakukan sebuah perhitungan untuk nilai presisi atau biasa disebut dengan nilai ketepatan, dan juga nilai perolehan (*recall*). *Precision* adalah sebuah kesamaan dalam permintaan informasi dari kata kunci pada sebuah sistem yang nantinya digunakan untuk menampilkan banyak dokumen ketika melakukan pencocokan dokumen yang dimana dokumen tersebut tidak *relevan*. Selanjutnya *recall* merupakan sebuah percobaan dalam jumlah dokumen yang nantinya dapat ditemukan dalam proses pencarian sistem temu kembali informasi atau *information retrieval*.

Pada proses *precision* (ketepatan) adalah rasio dari dokumen yang sesuai, dan juga jumlah dalam dokumen yang ditemukan dalam sebuah pencarian nantinya. Dalam presisi ini mempunyai kemampuan dalam sistem yang dimana untuk tidak memanggil dalam sebuah dokumen yang tidak relevan. Dalam menghitung nilai sebuah presisi atau nilai presisi, maka menggunakan persamaan 2.5 berikut ini [11].

$$\text{Presisi} = \frac{\text{Jumlah jawaban relevan sistem}}{\text{Total jawaban relevan pada sistem}} \quad (2.3)$$

Perhitungan dalam kinerja sistem untuk koreksi kata atau *query* yang menerapkan dengan menggunakan metode presisi dan *recall*. Presisi adalah jumlah dalam dokumen yang mendapatkan kembali sebuah sistem yang sesuai. Sedangkan *recall* adalah jumlah dalam sebuah dokumen yang sesuai dengan yang dihasilkan dari adanya proses untuk mendapatkan kembali sebuah sistem.

### 2.10.2 Perhitungan *Recall*

Selanjutnya adalah *recall* (perolehan) merupakan sebuah rasio yang digunakan sebagai perbandingan dari sebuah dokumen yang dapat ditemukan

dengan keseluruhan dokumen yang sesuai serta berada dalam sistem. *Recall* sama juga dalam kemampuan sebuah sistem untuk mengambil kembali sebuah dokumen yang relevan. Perhitungan nilai *recall* dapat digunakan dalam persamaan 2.6 berikut ini [11]

$$Recall = \frac{Jumlah\ jawaban\ relevan\ sistem}{Total\ jawaban\ relevan\ dalam\ teks} \quad (2.4)$$

### 2.11 Penelitian Terkait

Pada tahun 2016[29], melakukan penelitian dengan melakukan koreksi kata dalam *preprocessing* analisis sentimen pengguna *twitter*, dimana hasil perbandingan dari metode yang digunakan yaitu *Levenshtein Distance* dan *Jaro-Winkler Distance* yaitu diperoleh hasil Metode *Levenshtein Distance* menghasilkan nilai tertinggi yaitu *accuracy* 72,40%, *recall* 72,07%, *f1score* 79,11% sedangkan untuk hasil dari *Jaro-Winkler Distance* mempunyai *accuracy* 70%, *recall* 69,87% dan *f1score* 79,11%. Hal ini menunjukkan bahwa metode *Levenshtein Distance* lebih optimal untuk digunakan sebagai koreksi kata dalam *preprocessing*.

Pada tahun 2017[8], melakukan penelitian mengenai koreksi ejaan dengan menggunakan bahasa Indonesia untuk metode *Levenshtein Distance*. Dimana hasil pengujian diperoleh yaitu sebanyak 90 data diantaranya terdapat 3 skenario meliputi *deletion* (penghapusan) untuk akurasi sebesar 100% dengan waktu 23 mili detik, *insertion* (penyisipan) menghasilkan 93% dengan waktu 88 mili detik, dan *substitution* (substitusi) menghasilkan waktu 96% dengan waktu 5 mili detik.

Pada tahun 2018[1]. *Part-of-Speech* (POS) *Tagging* dengan menggunakan bahasa Indonesia digunakan peneliti untuk mengidentifikasi kelas kata ambigu. Dalam pemeriksaan ini dilakukan dengan menerapkan 71 prinsip sintaksis dalam melakukan perhitungan. Hasil yang diperoleh dari pengujian ini menunjukkan bahwa perhitungan menghasilkan nilai 92 kata dari banyaknya data 100 kata ambigu akurat, dua kata salah, dan enam kata yang tidak diselesaikan perhitungan. Ada beberapa penyebab yang mengakibatkan penyajian penghitungan, antara lain kelengkapan aturan, nama kelas (pelabelan kelas kata), dan korpus yang nanti dimanfaatkan dalam proses pelabelan fitur tata Bahasa (POS) *tagging*.

Pada tahun 2020[15] penelitian tentang *Part-of-Speech (POS) Tagging* metode *Rule Based*. Penulis menganalisa mengenai penggunaan untuk *Part-of-Speech (POS) Tagging* menggunakan bahasa inggris dengan pendekatan *rule based* lebih baik dibandingkan pendeka melalui *Stokastik*.

Pada tahun 2020[18] melakukan penelitian menggunakan *Autocorrect* pada pencarian obat atau disebut dengan *Drugs e-Dictionary*. Pada penelitian ini melakukan validasi *autocorrect*, untuk pencarian modul yang ada dalam *drugs e-dictionary*, dimana modul pencarian pada *drugs e-dictionary* dengan fitur *autocorrect* dapat mendeteksi kesalahan pengetikan dalam istilah yang dimasukkan dengan menghasilkan *output* istilah obat terdekat dalam database, selanjutnya melakukan secara otomatis memberikan saran perbaikan dan menampilkan hasil dari istilah obat yang ditingkatkan kepada pengguna, hal itu mencapai 90% akurasi kueri yang dimasukkan, dengan presisi 90% dan *recall* 90%.

Tabel 2. 1 Penelitian Terkait

No.	Peneliti, Tahun	Permasalahan	Metode	Hasil
1.	M.adnan Nur 2016[29]	Melakukan koreksi kata dalam <i>preprocessing</i> analisis sentimen pengguna <i>twitter</i>	Perbandingan <i>Levenshtein Distance</i> dan <i>Jaro-Winkler Distance</i>	Metode <i>Levenshtein Distance</i> menghasilkan nilai tertinggi yaitu <i>accuracy</i> 72,40%, <i>recall</i> 72,07%, <i>fiscore</i> 79,11% sedangkan untuk hasil dari <i>Jaro-Winkler Distance</i> mempunyai <i>accuracy</i> 70%, <i>recall</i> 69,87% dan <i>fIscore</i> 79,11%. Hal ini menunjukkan bahwa metode <i>Levenshtein Distance</i> lebih optimal untuk digunakan sebagai koreksi kata dalam <i>preprocessing</i> .
2.	Muhamad Omar Braddley, dkk 2017[8]	Dengan menggunakan <i>Levenshtein Distance</i> , kata bahasa Indonesia dapat dikoreksi ejaannya.	<i>Levenshtein Distance</i>	Proses hasil pengujian diperoleh yaitu sebanyak 90 data diantaranya terdapat 3 skenario meliputi <i>deletion</i> (penghapusan) untuk akurasi sebesar 100% dengan waktu 23 mili detik, <i>insertion</i> (penyisipan) menghasilkan 93% dengan waktu 88 mili detik, dan <i>substitution</i> (substitusi) menghasilkan waktu 96% dengan waktu 5 mili detik..



No.	Peneliti, Tahun	Permasalahan	Metode	Hasil
3.	Dewi Rosmala, Zulfikar Muhammad Risyad 2017[28]	Perhitungan Jarak <i>Levenshtein</i> pada Aplikasi Pencarian Kata Isu Kota Bandung di Twitter	Algoritma <i>Levenshtein Distance</i>	Berdasarkan hasil pengujian yang dilakukan yaitu dengan menggunakan algoritma <i>Levenshtein Distance</i> mampu 100% mengubah kata dengan kesalahan ejaan pada <i>tweet</i> menjadi kata kunci pada kategori isu, isu yang diperoleh Pemerintah Kota Bandung menjadi lebih baik dan akurat.
4.	Yazid & Fatwanto 2018[1]	Peneliti melakukan proses penentuan kelas kata yang bersifat ambigu pada <i>Part-of-Speech (POS) Tagging</i> bahasa Indonesia.	<i>Part-of-Speech (POS) Tagging</i>	Pada data 100 kata ambigu dapat diatasi sebanyak 92 dengan kata ambigu (92%) dalam penggunaan <i>corpus</i> dihasilkan dimana korpus yang digunakan dalam proses <i>Part-of-Speech (POS) Tagging</i> mempengaruhi hasil dari proses pelabelan kata
5.	Pham & Student, 2020[15]	Melakukan perbandingan untuk <i>Part-of-Speech (POS) Tagging</i> pada bahasa Inggris	Pendekatan <i>Rule based</i> dan pendekatan <i>Stokastik</i>	Hasil penerapan dengan menggunakan pendekatan <i>rule Based</i> lebih efisien dan lebih cepat.
6.	Halimah Tus Sadiyah dkk 2020[18]	<i>Autocorrect</i> pada Modul Pencarian <i>Drugs e-Dictionary</i> Menggunakan Algoritma <i>Levenshtein Distance</i>	Algoritma <i>Levenshtein Distance</i>	Hasil dari istilah obat yang ditingkatkan kepada pengguna, hal itu mencapai 90% akurasi kueri yang dimasukkan, dengan presisi 90% dan <i>recall</i> 90%, dimana menerapkan Algoritma <i>Levenshtein Distance</i> dan validasi <i>autocorrect</i> , dengan modul pencarian pada <i>drugs e-dictionary</i> memanfaatkan sebuah fitur <i>autocorrect</i> yang berfungsi untuk mendeteksi kesalahan pengetikan yang terdapat dalam istilah yang dimasukkan dengan <i>output</i> yaitu istilah obat terdekat dalam <i>database</i> , selanjutnya secara otomatis dapat memberikan sebuah saran untuk perbaikan dan hasil dari istilah obat.
7.	K. Sakaguchi, T. Mizumoto, M. Amaru	Melakukan penerapan koreksi kesalahan ejaan dan <i>part-of-speech</i>	<i>Spelling correction</i> dan <i>POS tagging</i>	Hasil yang diperoleh yaitu dengan melakukan pendekatan untuk memperbaiki kesalahan ejaan dan menetapkan tag <i>part-of-speech (POS) tagging</i> secara bersamaan untuk kalimat yang ditulis oleh pelajar bahasa Inggris sebagai

No.	Peneliti, Tahun	Permasalahan	Metode	Hasil
	Komachi , and Y. M. ji atsumot 2012 [30]	(POS) <i>tagging</i> dalam tulisan bahasa inggris sebagai media pembelajaran		bahasa kedua (ESL), menunjukkan peningkatan yang signifikan secara statistik dalam penandaan POS dan koreksi ejaan, dengan hasil peningkatan nilai F sebesar 2,1% dan 3,8% untuk POS dan peningkatan nilai F sebesar 5,0% untuk koreksi kesalahan ejaan dibandingkan dengan baseline atau model <i>pipeline</i> .

## **BAB III**

### **METODE USULAN**

Pada bab ini akan menjelaskan mengenai sebuah metode usulan maupun solusi yang direncanakan. Beberapa diantaranya yaitu tahapan penelitian, metode atau algoritma, arsitektur, dataset, dan evaluasi sistem serta skenario pengujian.

#### **3.1 Tahapan Penelitian**

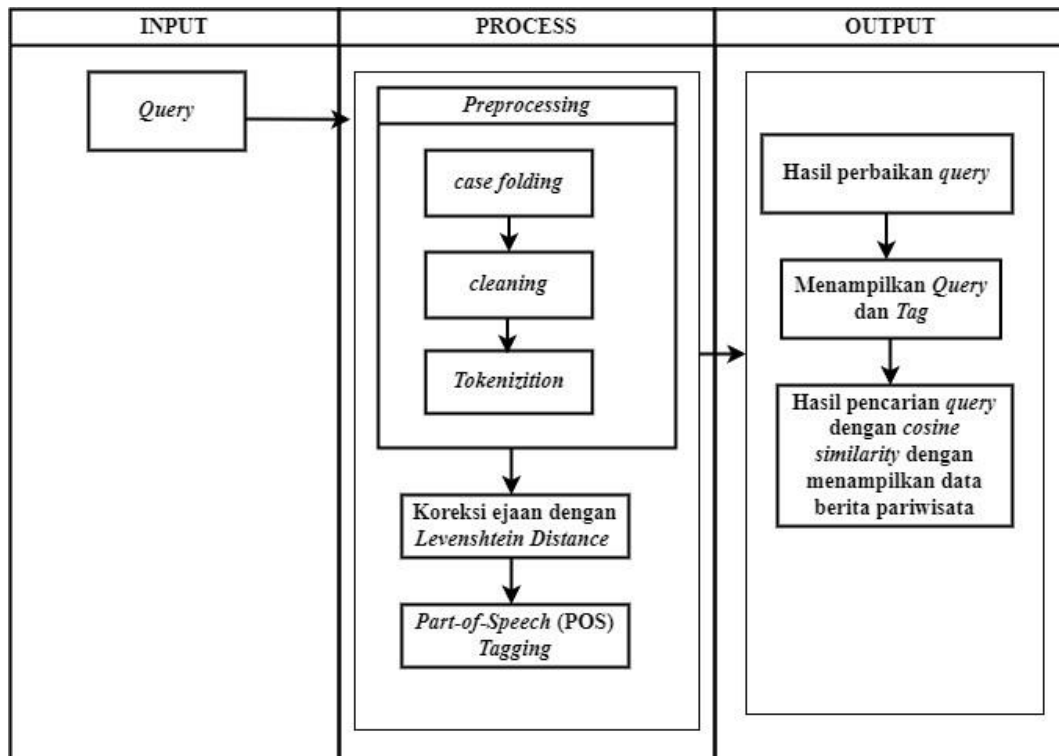
Pada tahapan penelitian ini, akan dilakukan beberapa tahapan antara lain:

##### **3.1.1 Studi Literatur**

Pada tahapan ini melakukan atau mencari referensi baik membaca agar memperoleh informasi dan melakukan pengumpulan informasi yang dibutuhkan dan dipelajari agar nantinya membantu dalam menyelesaikan penelitian. Beberapa informasi yang diperoleh yaitu dengan membaca, mempelajari literatur dari buku, jurnal, laporan penelitian, dan situs-situs *website* yang berkaitan dengan proposal penelitian skripsi ini. Data yang dikumpulkan yaitu berupa materi tentang atribut penyusunan koreksi ejaan *query* dengan menggunakan metode *Levenshtein Distance* dan *Part-of-Speech (POS) Tagging*.

##### **3.1.2 Analisis dan Perancangan Arsitektur Sistem**

Pada bagian analisis sistem ini yaitu berdasarkan hasil yang diperoleh dan dipelajari dari studi literatur yang dilakukan. Setelah melakukan analisa sebelumnya, maka selanjutnya merencanakan kerangka kerja sistem sebelum dilakukan pengimplementasian dengan menggunakan bahasa pemrograman. Pada penelitian ini, sistem yang akan dibangun adalah sebuah sistem koreksi kesalahan ejaan kata terhadap *query* pencarian artikel pariwisata berbahasa Indonesia menggunakan *Levenshtein Distance* dan *Part-of-Speech (POS) Tagging*. Berikut ini terdapat rancangan arsitektur atau diagram IPO dari sistem koreksi kesalahan ejaan kata pada artikel berita pariwisata, dapat dilihat pada Gambar 3. 1 dibawah ini.



Gambar 3. 1 Diagram IPO

Dapat dilihat pada Gambar 3. 1 diketahui perencanaan sistem pada penelitian ini adalah:

1. *Input data*

Proses awal yang dilakukan dengan menginputkan data berupa *query* (terdiri dari satu atau beberapa kata kunci) berita pariwisata pada mesin pencarian atau *search engine* bahasa Indonesia atau SEBI. Data yang diinputkan berupa kata dari kata kunci (*query*) yang diinginkan sesuai dengan kebutuhan.

2. *Process pengolahan dan pelabelan kelas kata (tag) query* berita pariwisata.

Tahap ini akan melakukan proses pengolahan data *query* berita pariwisata yang diinputkan sebelumnya, dengan melakukan *preprocessing* data. Selanjutnya kamus yang sudah tersedia didalam *database* yaitu gabungan dari data kamus berita pariwisata dan kamus bahasa Indonesia yang diperoleh dari *github* dengan link <https://github.com/keyreply/Bahasa-Indo-NLP-Dataset>. Melakukan koreksi ejaan *query* dengan metode *Levenshtein Distance* dengan

menghitung sebuah matriks yang digunakan untuk menghitung jarak perbedaan dari dua *string* diantaranya *string* sumber dan *string* target.

Kemudian melakukan pelabelan kelas kata (tag) *query* menggunakan *Part-of-Speech* (POS) *Tagging* dengan *Rule Based* yaitu metode berbasis aturan, menggunakan *corpus*. Selanjutnya melakukan pembobotan setiap *term* di dalam dokumen menggunakan *TF-IDF*, dengan mengukur similaritas antara hasil koreksi ejaan *query* yang dibenarkan dengan *query* ejaan salah, yang nantinya dibandingkan dengan kamus untuk melakukan pemeriksaan *query* yang diinputkan menggunakan *cosine similarity*, serta menghitung nilai *cosine similarity* untuk mengecek kesamaan dari kamus yang ada.

### 3. *Output* perbaikan *query* dari data yang diinputkan.

*Output* perbaikan *query* dari koreksi ejaan menggunakan metode *Levenshtein Distance* dan memberikan tag pada *query* menggunakan *Part-of-Speech* (POS) *Tagging* untuk kata ambigu pada *query* pencarian (SEBI) berita pariwisata. Maka *output* yang diperoleh yaitu perbaikan *query* berita pariwisata pada mesin pencarian (SEBI) dan mengecek ambiguitas dengan memberikan tag dari hasil perbaikan *query* yang diinputkan. Kemudian menampilkan data artikel berita pariwisata yang sudah dilakukan pemberian tag dengan menggunakan *Part-of-Speech* (POS) *Tagging* dengan penerapan *cosine similarity*.

### 3.1.3 Implementasi Sistem

Rancangan sistem yang telah dibuat ini nantinya akan diimplementasikan, dimana tahapan dari implementasi ini dimulai dengan menyiapkan data yang digunakan untuk melakukan koreksi ejaan *query* berita pariwisata dengan menggunakan metode *Levenshtein Distance* dan *Part-of-Speech* (POS) *Tagging*. Selanjutnya membuat program untuk proses data, dan melakukan proses uji coba dengan data *testing* yang digunakan. Pada proses implementasi ini nantinya data akan diolah dengan menggunakan Python dan hasil akhir menggunakan *framework* yaitu *streamlit* App.

### 3.1.4 Uji Coba Sistem

Pada tahapan uji coba sistem, nantinya akan melakukan proses uji coba, dimana data diproses dengan menggunakan metode *Levenshtein Distance* sehingga

dapat melakukan pengoreksian ejaan *query* berita pariwisata dan mengatasi ambiguitas kata dengan teknik melakukan pemberian tag pada *query*.

### 3.1.5 Analisa dan Evaluasi

Pada tahapan ini yaitu dilakukan untuk mengetahui nilai *presisi* dari koreksi ejaan *query* dengan menggunakan metode *Levenshtein Distance* dan *Part-of-Speech (POS) Tagging*. Apabila nilai *presisi* dari proses pengujian tidak sesuai dengan harapan, maka nantinya akan dilakukan evaluasi terhadap rancangan arsitektur sistem yang dibuat.

## 3.2 Dataset

Penelitian ini menggunakan dataset berita pariwisata bahasa Indonesia yang diperoleh dari penelitian terdahulu dengan melakukan *crawling* data berita pariwisata yang dimana data yang digunakan adalah data pada *website* yaitu laman *Detik.com* dengan data yang diperoleh sebanyak 332 dokumen dengan 133.403 kata. Hasil *crawling* data berita pariwisata yang diperoleh, terdapat judul, tanggal, link dan juga konten.

### 3.2.1 Data Corpus

*Corpus* adalah sebuah kumpulan teks yang tersusun dalam secara sistematis, dan teks yang terdapat didalam *corpus* digunakan dalam situasi dan kehidupan nyata[14]. Data *corpus* untuk melakukan proses *Part-of-Speech (POS) Tagging* yang digunakan dalam penelitian ini merupakan sebuah *corpus* bahasa Indonesia yang dibuat oleh ahli bahasa yang berasal dari Fakultas Ilmu Komputer Universitas Indonesia yang sudah diberi label dengan data *corpus* yang dapat di ambil di *github* <https://github.com/famrashel/idn-tagged-corpus#readmemd-versi-bahasa>, dan juga diambil dari *link* berikut <http://bahasa.cs.ui.ac.id/postag/corpus> .

### 3.2.2 Data Kamus

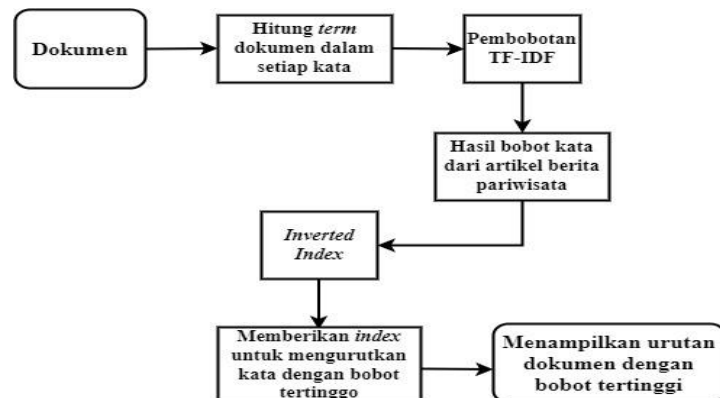
Data kamus yang didapat dari gabungan data bahasa Indonesia dengan kategori berita pariwisata sebanyak 332 dokumen dengan 133.403 kata dan juga data kamus bahasa Indonesia dari *github* dengan *link* <https://github.com/keyreply/Bahasa-Indo-NLP-Dataset> sebanyak 14.555 kata yaitu dengan data kamus baru sebanyak 23.888 kata yang disimpan di *file csv*. Berikut ini adalah tabel 3.1 dihalaman selanjutnya.

Tabel 3. 1 Data Kamus

Baris ke	Kata
1	aba
2	abad
3	abadi
4	abah
...	...
...	...
23884	zum
23885	zumba
23886	zuna
23887	zuroh
23888	zusnali

### 3.2.3 *Inverted Index*

Pada proses *inverted index* dengan TF-IDF, yaitu pada *term* (kata) yang dibuat untuk mempresentasikan *term* (kata) yang unik dalam koleksi dokumen, kemudian frekuensi kemunculan setiap *term* pada dokumen disimpan, dan dibuat *inverted index* atau daftar dokumen dari setiap *term* yang dibuat. Selanjutnya diurutkan katanya (*term*) dimana setiap *term*nya mempunyai bobot kata, selanjutnya diurutkan dan diperoleh *inverted index*. Hasil dari *inverted index* berupa *term* (kata) yang sudah punya bobot atau artikel berita pariwisata yang sudah di *inverted index*, digunakan untuk mengurutkan kata yang memiliki bobot paling tinggi. Sehingga untuk menampilkan artikel berita pariwisata yang sudah di *inverted index* dengan *query* yang sudah dilakukan perbaikan, akan dilakukan atau di proses didalam *cosine similarity*. Berikut ini adalah *flowchart* dari *inverted index* pada gambar 3.3 yang berada dibawah ini.



Gambar 3. 2 *Inverted index*

Pada bagian ini adalah contoh proses *inverted index* dengan data yang sederhana yaitu sebagai berikut :

Misalkan terdapat 6 (enam) dokumen teks yang terdapat dalam tabel 3.2 dibawah ini yaitu sebagai berikut :

Tabel 3. 2 Dokumen artikel pariwisata

Dokumen	<i>Tokenization</i>
<b>Q</b>	<b>Karnaval budaya di klaten</b>
1	"['reog', 'hingga', 'pentas', 'tari', 'meriahkan', 'karnaval', 'budaya', 'di', 'klaten']"
2	"['karnaval', 'budaya', 'klaten', 'usung', 'berbagai', 'potensi', 'seni', 'budaya']"
3	"['siang', 'ini', 'ada', 'karnaval', 'budaya', 'di', 'pusat', 'kota', 'klaten', 'ini', 'pengalihan', 'arusnya']"
4	"['kesenian', 'kuda', 'kosong', 'meriahkan', 'helaran', 'budaya', 'di', 'cianjur']"
5	"['perhatian', 'asn', 'pemprov', 'dki', 'ini', 'aturan', 'uji', 'coba', 'wfh']"
6	"['lomba', 'perahu', 'bidar', 'kembali', 'digelar', 'setelah', 'vakum', 'tahun']"

Pertama yang perlu dilakukan yaitu membuat *inverted index* melalui membagi setiap dokumen menjadi token-token atau kata-kata individu. Hasil dari ketiga token tersebut nantinya seperti pada Tabel 3.2 yaitu Dokumen artikel pariwisata. Kedua membuat daftar unik yang dimana dari semua token yang ditemukan dalam dokumen-dokumen tersebut akan menjadi daftar unik atau menjadi kunci dalam *inverted index*. Ketiga membuat sebuah posting list untuk setiap token yang terdapat dalam daftar unik, dimana dalam *posting list* ini yaitu dimana token dari *query* yang sesuai dengan kata kunci “karnaval budaya klaten” dimana dalam token ini, dokumen mana saja yang berisi informasi yang mengandung token. Berikut adalah Tabel 3. 3 contoh dari perhitungan dari *term* dalam setiap dokumen yang berada dihalaman selanjutnya.



Tabel 3. 3 Proses Hitung *Term* dalam setiap dokumen

<i>Term</i>	Q	tf						df	D/df	IDF(log D/df)	W = TF*(IDF(log D/df) )						Q
		D1	D2	D3	D4	D5	D6				D1	D2	D3	D4	D5	D6	
reog	0	1	0	0	0	0	0	1	6	0,77815125	0,77815	0	0	0	0	0	
hingga	0	1	0	0	0	0	0	1	6	0,77815125	0,77815	0	0	0	0	0	0
pentas	0	1	0	0	0	0	0	1	6	0,77815125	0,77815	0	0	0	0	0	0
tari	0	1	0	0	0	0	0	1	6	0,77815125	0,77815	0	0	0	0	0	0
meriahkan	0	1	0	0	0	0	0	1	6	0,77815125	0,77815	0	0	0	0	0	0
karnaval	1	1	1	1	0	0	0	3	2	0,30103	0,30103	0,30103	0,30103	0	0	0	0,30103
budaya	1	1	2	1	1	0	0	5	1,2	0,07918125	0,07918	0,15836	0,07918	0,07918	0	0	0,07918
di	1	1	0	1	1	0	0	3	2	0,30103	0,30103	0	0,30103	0,30103	0	0	0,30103
klaten	1	1	1	0	0	0	0	2	3	0,47712125	0,47712	0,47712	0	0	0	0	0,47712
usung	0	0	1	0	0	0	0	1	6	0,77815125	0	0,77815	0	0	0	0	0
berbagai	0	0	1	0	0	0	0	1	6	0,77815125	0	0,77815	0	0	0	0	0
potensi	0	0	1	0	0	0	0	1	6	0,77815125	0	0,77815	0	0	0	0	0
seni	0	0	1	0	0	0	0	1	6	0,77815125	0	0,77815	0	0	0	0	0
siang	0	0	0	1	0	0	0	1	6	0,77815125	0	0	0,77815	0	0	0	0
ini	0	0	0	1	0	0	0	1	6	0,77815125	0	0	0,77815	0	0	0	0
ada	0	0	0	1	0	0	0	1	6	0,77815125	0	0	0,77815	0	0	0	0
pusat	0	0	0	1	0	0	0	1	6	0,77815125	0	0	0,77815	0	0	0	0
kota	0	0	0	1	0	0	0	1	6	0,77815125	0	0	0,77815	0	0	0	0
pengalihan	0	0	0	1	0	0	0	1	6	0,77815125	0	0	0,77815	0	0	0	0
arusnya	0	0	0	0	1	0	0	1	6	0,77815125	0	0	0	0,77815	0	0	0

Term	Q	tf						df	D/df	IDF(log D/df)	W = TF*(IDF(log D/df) )						Q
		D1	D2	D3	D4	D5	D6				D1	D2	D3	D4	D5	D6	
kesenian	0	0	0	0	1	0	0	1	6	0,77815125	0	0	0	0,77815	0	0	0
kuda	0	0	0	0	1	0	0	1	6	0,77815125	0	0	0	0,77815	0	0	0
kosong	0	0	0	0	1	0	0	1	6	0,77815125	0	0	0	0,77815	0	0	0
helaran	0	0	0	0	1	0	0	1	6	0,77815125	0	0	0	0,77815	0	0	0
cianjur	0	0	0	0	1	0	0	1	6	0,77815125	0	0	0	0,77815	0	0	0
perhatian	0	0	0	0	0	1	0	1	6	0,77815125	0	0	0	0	0,77815	0	0
asn	0	0	0	0	0	1	0	1	6	0,77815125	0	0	0	0	0,77815	0	0
pemprov	0	0	0	0	0	1	0	1	6	0,77815125	0	0	0	0	0,77815	0	0
dki	0	0	0	0	0	1	0	1	6	0,77815125	0	0	0	0	0,77815	0	0
aturan	0	0	0	0	0	1	0	1	6	0,77815125	0	0	0	0	0,77815	0	0
uji	0	0	0	0	0	1	0	1	6	0,77815125	0	0	0	0	0,77815	0	0
coba	0	0	0	0	0	1	0	1	6	0,77815125	0	0	0	0	0,77815	0	0
wfh	0	0	0	0	0	1	0	1	6	0,77815125	0	0	0	0	0,77815	0	0
lomba	0	0	0	0	0	0	1	1	6	0,77815125	0	0	0	0	0	0,77815	0
perahu	0	0	0	0	0	0	1	1	6	0,77815125	0	0	0	0	0	0,77815	0
bidar	0	0	0	0	0	0	1	1	6	0,77815125	0	0	0	0	0	0,77815	0
kembali	0	0	0	0	0	0	1	1	6	0,77815125	0	0	0	0	0	0,77815	0
digelar	0	0	0	0	0	0	1	1	6	0,77815125	0	0	0	0	0	0,77815	0
setelah	0	0	0	0	0	0	1	1	6	0,77815125	0	0	0	0	0	0,77815	0
vakum	0	0	0	0	0	0	1	1	6	0,77815125	0	0	0	0	0	0,77815	0
tahun	0	0	0	0	0	0	1	1	6	0,77815125	0	0	0	0	0	0,77815	0

Hasil dari proses *inverted index*nya yaitu mengurutkan dokumen dengan bobot tertinggi dan bertujuan memungkinkan proses pencarian dokumen yang efisien dan cepat dari *query* yang cari. Berikut ini Tabel 3. 4 dibawah ini hasil dari *inverted index* secara manual.

Tabel 3. 4 *Inverted index*

Token	Dokumen
Karnaval	D1,D2,D3
budaya	D1,D2,D3,D4
di	D1,D3, D4
Klaten	D1, D2

### 3.2.4 Data Uji

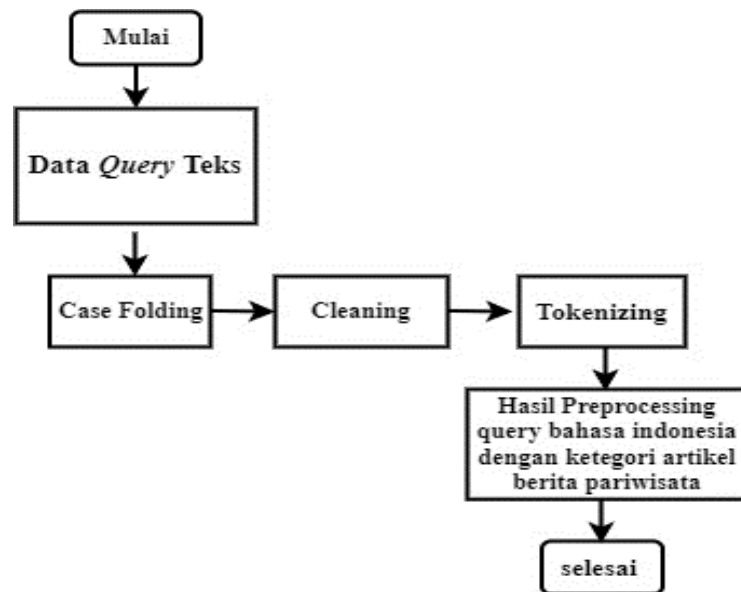
Data uji ini dilakukan dengan membuat sebuah *query* salah dan *query* benar yang nantinya akan dilakukan dalam koreksi ejaan terhadap *query* bahasa Indonesia pada pencarian artikel berita pariwisata. Dalam *query* ini mencakup mengenai pariwisata. Data uji yang digunakan dalam penelitian ini berjumlah 100 data *query* salah tentang pariwisata yang diperoleh dari artikel berita pariwisata menggunakan *microsoft excel* dengan memasukkan kata yang terdapat didalam kamus. Kesalahan kata dalam data uji meliputi dari rincian *query* berupa operasi penyisipan, penghapusan, dan substitusi. Berikut ini Tabel 3.5 Data Uji *query*.

Tabel 3. 5 Data Uji

Baris ke-	<i>Query</i> Salah	<i>Query</i> Benar
1.	Kepalla Dienas	Kepala Dinas
2.	Pntas Tari Meriahkn Karnaval Bdaya	Pentas Tari Meriahkan Karnaval Budaya
3.	Siagn Inie Adha Kanraval	Siang Ini Ada Karnaval
4.	Pemmuda lahraga daen Parwisata Pemkabb Klateen	Pemuda Olahraga dan Pariwisata Pemkab Klaten
5.	Mengikti pawwai pembagnunan	Mengikuti pawai pembangunan
6.	Parriwsta	Pariwisata
7.	Pamerran tepradu sekttor perdagnagan, pariwisatata	Pameran terpadu sektor perdagangan, pariwisataa
8.	samppah di detinasi wista	sampah di destinasi wisata
9.	Pembangnan ppariwisata brkelanjutann	Pembangunan pariwisataa berkelanjutan
10.	Menterii Parriwisata ddan Ekonmoi Krreatif	Menteri Pariwisata dan Ekobaris ke mi Kreatif

### 3.3 Text Preprocessing

Pada tahapan ini yaitu dilakukan *preprocessing*, untuk membuat koleksi data siap untuk digunakan. Data mentah di proses pada tahap *preprocessing*. Proses *preprocessing* teks ini digunakan untuk mengurangi *noise* dan merapikan data, menyamakan bentuk kata, dan mengurangi volume data. Sehingga nantinya data yang sudah di *preprocessing* akan siap di proses, sedangkan data yang di peroleh dari hasil *crawling* akan dihapus. Terdapat langkah-langkah dalam tahapan *preprocessing*. Pada Gambar 3. 3 dibawah ini adalah diagram alurnya.



Gambar 3. 3 Diagram Alur *Preprocessing*

Berikut merupakan deskripsi alur atau tahapan pada Gambar 3.3 dalam penelitian koreksi kesalahan ejaan antara lain:

#### 1. *Case Folding*.

*Case Folding* yaitu cara paling umum untuk mengubah huruf kapital menjadi huruf kecil, dimana dalam setiap karakter a-z pada suatu kata maupun kalimat akan diubah dari huruf besar menjadi huruf kecil[31]. Berikut hasil dari *case folding* pada Tabel 3. 6 di halaman berikut.

Tabel 3. 6 *Case Folding*

<b>Kata Sebelum <i>Case Folding</i></b>	<b>Kata Setelah <i>Case Folding</i></b>
Kesenian Kuda Kosong Meriahkan Helaran Budaya di Cianjur	kesenian kuda kosong meriahkan helaran budaya di cianjur
Bali Jadi Potret Pengelolaan Pariwisata Labuan Bajo	bali jadi potret pengelolaan pariwisata labuan bajo

## 2. *Cleaning*

*Cleaning* merupakan proses membersihkan sebuah karakter tertentu yang tidak dibutuhkan seperti simbol dan angka dalam kata maupun kalimat [31]. Berikut contoh hasil dari *cleaning* pada Tabel 3.7 dibawah ini.

Tabel 3. 7 *Cleaning*

<b>Kata Sebelum <i>Cleaning</i></b>	<b>Kata Setelah <i>Cleaning</i></b>
kesenian kuda kosong meriahkan helaran budaya di cianjur	kesenian kuda kosong meriahkan helaran budaya di cianjur
bali jadi potret pengelolaan pariwisata labuan bajo	bali jadi potret pengelolaan pariwisata labuan bajo

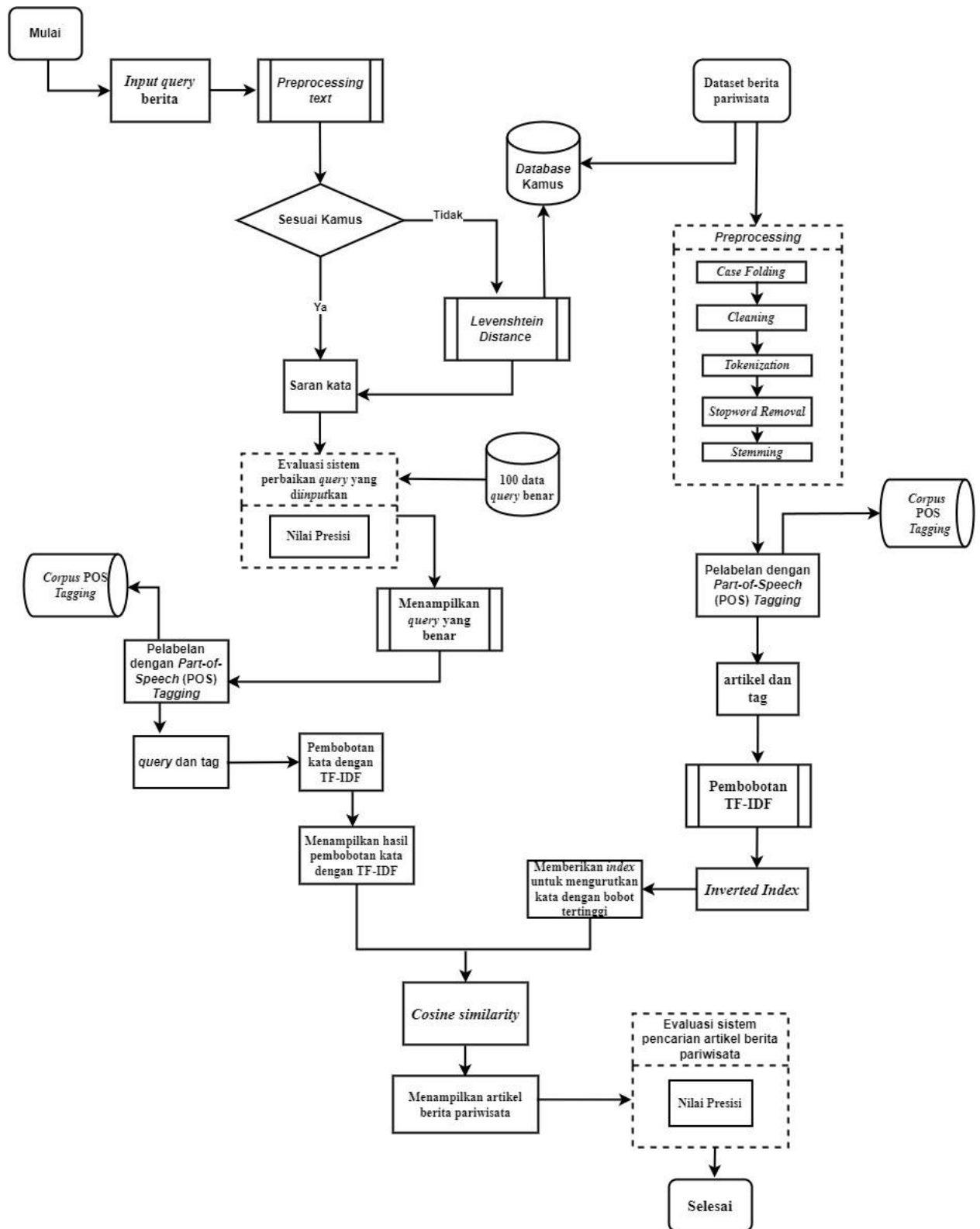
## 3. *Tokenizing*

*Tokenizing* merupakan proses pemecahan suatu kalimat menjadi satuan-satuan terkecil (kata/token)[31]. Berikut contoh hasil dari *tokenizing* pada Tabel 3. 8 dibawah ini.

Tabel 3. 8 *Tokenizing*

<b>Kata Sebelum <i>Tokenizing</i></b>	<b>Kata Setelah <i>Tokenizing</i></b>
kesenian kuda kosong meriahkan	['kesenian','kuda', 'kosong', 'meriahkan']
bali jadi potret pengelolaan pariwisata labuan bajo	['bali', 'jadi', 'potret', 'pengelolaan', 'pariwisata', 'labuan','bajo']

### 3.4 Arsitektur Sistem



Gambar 3. 4 Arsitektur Sistem

Pada Gambar 3. 4 sebelumnya merupakan arsitektur sistem yang terdiri dari beberapa tahapan berikut ini:

Pada *query*, proses yang dilakukan menginputkan data *query* teks yang berkaitan dengan berita pariwisata, dimana *query* yang diinputkan adalah *query* salah dari 100 data uji *query* yang dibuat yaitu *query* benar dan *query* salah. Setelah itu, melakukan *preprocessing* data *query*, dimana terdapat beberapa tahapan dalam *preprocessing*, diantaranya *case folding*, *cleaning*, *tokenization*. Dan menyiapkan data kamus (gabungan data hasil *crawling* yang diperoleh dari penelitian sebelumnya, dan dilakukan pada *website* atau laman *Detik.com* dengan kategori berita pariwisata dan data kamus bahasa Indonesia dari *github* dengan link <https://github.com/keyreply/Bahasa-Indo-NLP-Dataset>) yang nantinya disimpan di *database*.

Kemudian hasil dari *preprocessing* pada *query*, akan dilakukan pengecekan apakah sesuai kamus (kamus dengan kategori artikel berita pariwisata) atau tidak. Apabila *query* yang diinputkan salah, akan dilakukan pengecekan dengan menggunakan metode *Levenshtein Distance* untuk menghitung jarak antara dua *string* yang bertujuan melakukan koreksi kesalahan ejaan dan mengetahui perbaikan ejaan kata dari *query* yang diinputkan, dengan melakukan pengecekan dengan membandingkan kamus yang ada di *database*. Sebaliknya, apabila *query* tersebut sesuai dengan kamus, maka akan menampilkan saran kata berupa *query* benar.

Metode *Levenshtein Distance* dilakukan dimana suatu matriks untuk mengukur jumlah perbedaan antara dua *string* atau karakter, yang memiliki panjang karakter yang sama dengan kata *input*. *Levenshtein Distance* dua buah *string* merupakan jumlah dari minimum operasi yang dibutuhkan untuk mengubah sebuah *string* (*source string*) menjadi *string* yang lain (*string target*). Dalam algoritma *Levenshtein Distance* ini merupakan suatu operasi yang melibatkan penyisipan (*insertion*), penghapusan (*deletion*), penggantian (*substitution*) dari suatu karakter tunggal.

Selanjutnya, dalam proses evaluasi sistem perbaikan *query* yang diinputkan, yaitu menghitung nilai presisi dari *query* yang terdapat didalam 100 data uji *query* yang telah dibuat sebelumnya. Prosedur yang ada di dalam evaluasi sistem

perbaikan ini, apabila *query* yang diinputkan sesuai maka akan menampilkan *query* benar, sebaliknya jika tidak berarti *query* tersebut adalah *query* salah. Kemudian, ketika *query* yang diinputkan sudah benar, maka akan dilakukan proses *Part-of-Speech* (POS) *Tagging* untuk memberikan pelabelan kelas kata secara otomatis dalam suatu kalimat, dengan teknik melakukan pelabelan kelas kata (pemberian *tag*) dengan menggunakan *corpus* bahasa Indonesia.

*Part-of-Speech* (POS) *Tagging* dengan metode *Rule Based* digunakan sebagai dasar penelitian dari *information retrieval* yaitu melakukan pelabelan kelas kata dengan memberikan *tag* menggunakan *corpus* bahasa Indonesia. Tujuan dari adanya *Part-of-Speech* (POS) *Tagging* ini untuk mengetahui adanya ambiguitas kata atau kata ambigu, dengan memberikan informasi mengenai jenis kata atau kelas kata dalam suatu kalimat. Misalnya, kata “kosong” dapat memiliki dua makna yang berbeda, yaitu Rangkaian Helaran budaya itu dibuka dengan ikon khas Cianjur yakni Kuda Kosong. Pada konteks dengan menggunakan *Part-of-Speech* (POS) *Tagging*, dapat mengetahui apakah makna yang tepat dari kata tersebut berdasarkan konteks kalimatnya, yaitu kata “kosong” sebagai kata sifat yang berarti tidak berisi atau kata “kosong” sebagai nama suatu objek atau *entitas* nama tempat atau istilah lokal yang mungkin menjadi ikon khas Cianjur.

Kemudian, melakukan pembobotan kata dari *term* menggunakan *TF-IDF* untuk mengetahui frekuensi setiap kata didalam dokumen, dimana *term TF* diperhitungkan untuk pemberian bobot terhadap suatu kata, karena nantinya hasilnya pembobotan *TF-IDF* berupa angka. Hasil dari pembobotan kata ini nantinya dilakukan proses *inverted index* untuk melakukan pengurutan dengan memberikan *index* dari dokumen berita pariwisata yang memiliki bobot paling tinggi. Kemudian dari hasil *inverted index* ini dilakukan pengecekan dengan menggunakan *cosine similiarity* untuk mengukur seberapa sesuai suatu dokumen berita pariwisata dengan *query* yang diinputkan oleh pengguna. Kemudian untuk melakukan pengecekan dari dua dokumen tersebut dapat diketahui dokumen yang mempunyai kemiripan paling tinggi dari kesamaan dengan *query* yang inputkan oleh pengguna dengan mengecek kamus yang digunakan.

Setelah itu, *output* dari sistem ini adalah hasil koreksi ejaan *query* yang benar dimana dalam pengecekan pada *query* pencarian yang di inputkan akan mengetahui



adanya ambiguitas kata dengan teknik pelabelan kelas kata (pemberian tag). Hasil dari pengecekan tersebut menampilkan informasi mengenai artikel berita pariwisata. Kemudian dari hasil *query* yang diinputkan berupa perbaikan kata dan informasi terkait artikel berita pariwisata berbahasa Indonesia. Selanjutnya melakukan evaluasi sistem, untuk skenario uji coba dengan menghitung nilai akurasi berupa *presisi* dengan atau tanpa menggunakan *Part-of-Speech* (POS) *Tagging* dan menggunakan metode *Levenshtein Distance*.

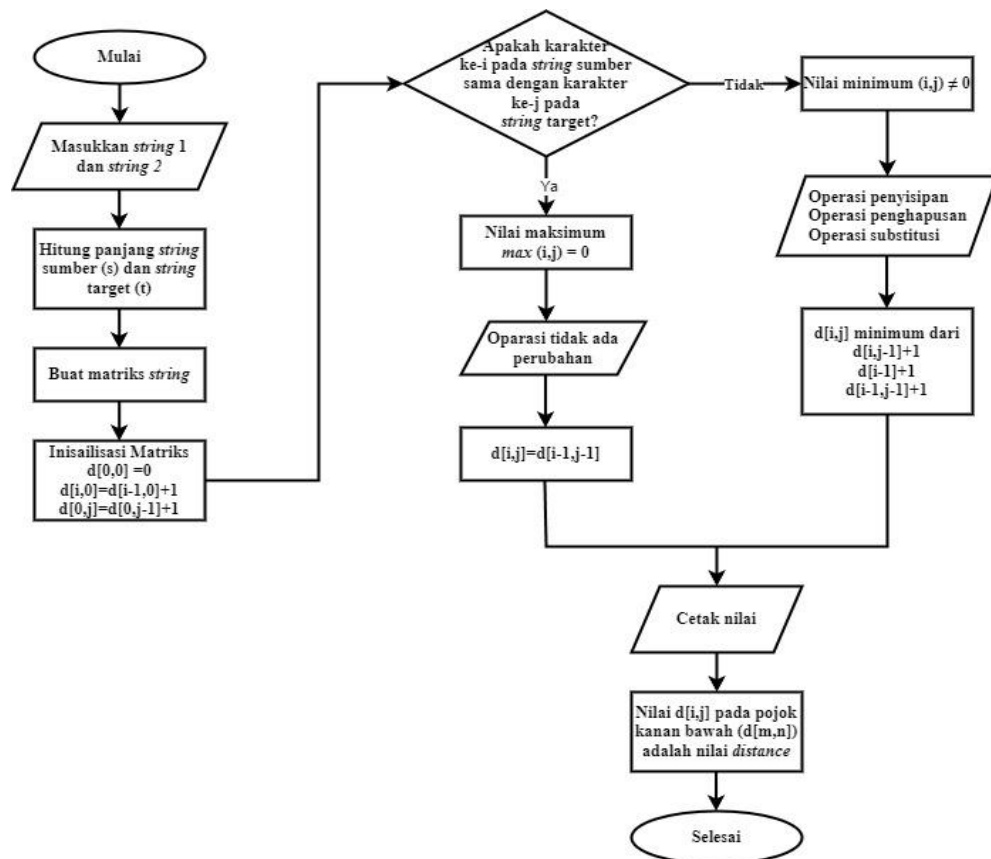
Proses dataset, dalam proses dataset ini nantinya akan menghasilkan sebuah kamus, yang mana kamus yang dibuat berkaitan dengan dataset atau koleksi data yang berjumlah 332 dokumen yang dilakukan proses pembuatan kamus dengan melakukan persamaan pada kamus *indonesianword.txt*, kemudian dari dataset ini juga dilakukan proses *preprocessing* yang mana dalam proses ini nantinya akan dilakukan *case folding*, *cleaning*, *tokenization*, *stopword removal*, dan *stemming*. Setelah itu dilakukan *Part-of-Speech* (POS) *Tagging* dalam artikel berita pariwisata didalam dokumen tersebut, untuk mengetahui adanya kata ambigu, sehingga dilakukan pelabelan kelas kata, dengan tujuan untuk dapat menentukan makna kata yang tepat dari kata tersebut berdasarkan konteks kalimatnya. Dengan demikian penggunaan *Part-of-Speech* (POS) *Tagging* dapat memahami makna kata yang ambigu. Selanjutnya akan dilakukan pembobotan *TF-IDF*, yang nantinya hasil dari pembobotan kata dari berita pariwisata, nantinya akan dilakukan *inverted index* untuk mengurutkan kata yang sudah mempunyai bobot dari dokumen berita pariwisata yang memiliki bobot paling tinggi.

### **3.5 Flowchart Algoritma**

#### **3.5.1 Flowchart Levenshtein Distance**

*Levenshtein Distance* direpresentasikan yaitu dengan memulai dari bagian sudut kiri atas yang mempunyai sebuah *array* dua dimensi (matriks), yang sebelumnya dilakukan pengisian dari sejumlah karakter untuk *string* sumber maupun *string* target. Sehingga nantinya nilai yang di masukkan didalam matriks akan melakukan perhitungan dengan mempresentasikan sebuah nilai yang terkecil dari hasil perubahan yang diperoleh dalam karakter yang terdapat di dalam *string* sumber maupun *string* target. Hasil masukan dari perhitungan yang terdapat dalam

ujung kanan bawah matriks nantinya akan merepresentasikan nilai *distance* dengan menggambarkan jumlah perbedaan yang terdapat didalam dua *string*. Terdapat beberapa langkah-langkah dalam penerapan algoritma *Levenshtein Distance* untuk memperoleh nilai *distance*. Pada Gambar 3. 5 dihalaman selanjutnya, adalah *flowchart* dari algoritma *Levenshtein Distance*.



Gambar 3. 5 *Flowchart Levenshtein Distance*

Pada tahapan yang terdapat dalam gambar dalam algoritma *Levenshtein Distance*[32]. Dalam hal ini, dapat di representasikan untuk *string* sumber (s) dan *string* target (t), adapun langkah-langkah yang dilakukan yaitu:

- Langkah pertama melakukan inisiasi yaitu sebagai berikut:
  - a. Menghitung panjang dari *string* sumber dan *string* target, dengan inisiasi yaitu m dan juga n
  - b. Membuat sebuah matriks yang berukuran 0...m baris dan 0...n kolom
  - c. Melakukan inisiasi pada baris pertama dengan 0...n
  - d. Selanjutnya membuat sebuah inisiasi berupa kolom dengan 0...m

- Langkah kedua adalah proses selanjutnya ketika sudah melakukan langkah pertama
  - a. Melakukan pengecekan atau memeriksa dari *string* sumber  $S[i]$  untuk  $1 < i < n$
  - b. Melakukan pengecekan atau memeriksa dari *string* target  $T[j]$  untuk  $1 < j < m$
  - c. Apabila terdapat *string* sumber sama dengan *string* target atau bisa dikatakan  $S[i] = T[j]$ , maka entrinya yaitu berupa nilai yang terletak tepat di diagonal yang atas posisi sebelah kiri atau bisa diinisiasi  $d[i,j] = d[i-1,j-1]$
  - d. Selanjutnya, apabila terdapat *string* sumber tidak sama dengan *string* target  $S[i] \neq T[j]$ , maka entrinya yaitu berupa  $d[i,j]$  hasil minimum dari:  
 Sebuah nilai yang ada dan terletak tepat diatasnya, kemudian ditambah satu, yaitu dengan inisiasi  $d[i,j-1]+1$   
 Setelah itu, apabila Nilai yang terletak tepat dikirinya, akan ditambah satu, yaitu  $d[i-1,j]+1$   
 Dan terletak pada bagian tepat didiagonal atas sebelah kirinya, ditambah satu, yaitu  $d[i-1,j-1]+1$
- Kemudian melakukan langkah ketiga, dimana dalam langkah ini, hasil dari entri matriks yang dilakukan pada baris ke -i dan pada kolom ke-j dapat diinisiasi, yaitu  $d[i,j]$ .
- Hasil dari proses atau langkah yang dilakukan sebelumnya, maka proses atau langkah kedua dilakukan atau diulang kembali sehingga entri dari  $d[m,n]$  ditemukan.

### 3.5.2 Perhitungan Metode *Levenshtein Distance*

Pada perhitungan dengan menggunakan metode *Levenshtein Distance* terdapat sebuah *string* yang terdiri dari dua *string* (*string* sumber dan *string* target) yang digunakan dalam contoh perhitungan dari algoritma *Levenshtein Distance*. Berikut ini adalah rumus dari *Levenshtein Distance*.

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} lev_{a,b}(i-1,j) + 1 & \text{(penghapusan)} \\ lev_{a,b}(i,j-1) + 1 & \text{(Penyisipan)} \\ lev_{a,b}(i-1,j-1) + 1 & \text{if } a_i \neq b_j \text{ (substitusi)} \end{cases} & \text{otherwise} \end{cases}$$

$$lev_{a,b}(i,j) = \min lev_{a,b}(i-1,j) + 1 \text{ (Penghapusan)} \quad (2.5)$$

$$lev_{a,b}(i,j) = \min lev_{a,b}(i,j-1) + 1 \text{ (Penyisipan)} \quad (2.6)$$

$$lev_{a,b}(i,j) = \min lev_{a,b}(i-1,j-1) + 1, a_j \neq b_i \text{ (Substitusi)} \quad (2.7)$$

$$lev_{a,b}(i,j) = \min lev_{a,b}(i-1,j-1), a_j = b_i \text{ (Tidak ada perubahan)} \quad (2.8)$$

**Keterangan:**

$a$  = string Sumber

$i$  = index baris string sumber

$b$  = string Target

$j$  = index baris string target

Kondisi  $a_i \neq b_j$

$(a_i \neq b_j)$  = perlu menambahkan 1 (+1)

$(a_i = b_j)$  = tidak perlu menambahkan 1 (+1)

Terdapat variabel  $a$  merupakan string sumber (*input*) dan  $b$  adalah string target, kedua string ini akan dilakukan perhitungan jarak  $lev_{a,b}$ . Panjang setiap string akan ditambahkan 1, kemudian dilakukan pencocokan string untuk mendapatkan jarak. Setiap karakter yang sama baik dari string sumber dan string target akan dinilai 0 dan yang berbeda akan diberi nilai 1. Pada proses ini akan dilakukan secara berurutan dengan melakukan operasi penyisipan (*insertion*), penghapusan (*deletion*), penggantian (*substitution*). Semakin kecil nilai jarak antara string, maka akan direkomendasikan sebagai perbaikan kata. Berikut ini adalah contoh dari penerapan algoritma *Levenshtein Distance*.

String sumber (S) Tulis

String target (T) Turis

S/T		t	u	r	i	s
	0	1	2	3	4	5
t	1	0	1	2	3	4
u	2	1	0	1	2	3
l	3	2	1	1	2	3
i	4	3	2	2	1	2
s	5	4	3	3	2	1

→ Nilai jarak

Gambar 3. 6 Percobaan metode *Levenshtein Distance* di Excel

Pada perhitungan metode *Levenshtein Distance* terdapat substitusi huruf dari kata yang sebelumnya “Tulis” menjadi “Turis”, dengan hasil jarak yaitu 1 yang berada di kolom warna kuning, dan kolom warna hijau menunjukkan hasil operasi dari substitusi yaitu 1.

### 3.5.3 Penerapan *Part-of-Speech (POS) Tagging*

Penggunaan *Part-of-Speech (POS) Tagging* ini bertujuan untuk mengetahui adanya ambiguitas kata dengan teknik pelabelan kelas kata, dimana dari pelabelan kelas kata ini nantinya berpengaruh dalam mengetahui kata yang bermakna banyak. Sehingga dari data artikel berita pariwisata ini akan dilakukan *Part-of-Speech (POS) Tagging*, dan membutuhkan *corpus* yang berisi sebuah kata-kata dan kumpulan beberapa kalimat dengan memberikan pelabelan kelas katanya, dan *corpus* yang digunakan pada penelitian ini yaitu *corpus* bahasa Indonesia. Penelitian ini menggunakan data *corpus* bahasa Indonesia dari Fakultas Ilmu Komputer Universitas Indonesia atau *POS Tag-Corpus Site* (ui.ac.id). Berikut terdapat beberapa macam *tagset* beserta contohnya yang terdapat pada Tabel 3.9 dihalaman selanjutnya[33].

Tabel 3. 9 *Tagset POS Tagging*

Tag	Keterangan	Contoh
CD	<i>Cardinal number</i>	dua, juta, enam, 7916, sepertiga, 0,025, 0,525, banyak, kedua, ribuan, 2007, 25
CC	<i>Coordinating conjunction</i>	dan, tetapi, atau
DT	<i>Determiner / article</i>	Para, Sang, Si
OD	<i>Ordinal number</i>	ketiga, ke-4, pertama
FW	<i>Foreign word</i>	<i>climate change, terms and conditions</i>
JJ	<i>Adjective</i>	bersih, panjang, hitam, lama, jauh, marah, suram, nasional, bulat
IN	<i>Preposition</i>	dalam, dengan, di, ke, oleh, pada, untuk
MD	<i>Modal and auxiliary verb</i>	boleh, harus, sudah, mesti, perlu
NN	<i>Noun</i>	monyet, bawah, sekarang, rupiah
NEG	<i>Negation</i>	tidak, belum, jangan
NNP	<i>Proper noun</i>	Boediobaris ke , Laut Jawa, Indonesia, India, Malaysia, Bank Mandiri, BBKP, Januari, Senin, Idul Fitri, Piala Dunia, Liga Primer, Lord of the Rings: The Return of the King
NND	<i>Classifier, partitive, and measurement noun</i>	orang, ton, helai, lembar
PR	<i>Demonstrative noun</i>	ini, itu, sini, situ
PRP	<i>Personal noun</i>	saya, kami, kita, kamu, kalian, dia, mereka
RB	<i>Adverb</i>	sangat, hanya, justru, niscaya, segera

Tag	Keterangan	Contoh
RP	<i>Particle</i>	pun, -lah, -kah
SC	<i>Subordinating conjunction</i>	sejak, jika, seandainya, supaya, meski, seolah-olah, sebab, maka, tanpa, dengan, bahwa, yang, lebih ... daripada ..., semoga
SYM	<i>Symbol</i>	IDR, +, %, @
UH	<i>Interjection</i>	brengsek, oh, ooh, aduh, ayo, mari, hai
VB	<i>Verb</i>	merancang, mengatur, pergi, bekerja, tertidur
WH	<i>Question</i>	siapa, apa, mana, kenapa, kapan, di mana, bagaimana, berapa
X	<i>Unkbaris ke wn</i>	statemen
Z	<i>Punctuation</i>	"...", ?, .

**a. Contoh *Part-of-Speech* (POS) Tagging pada *Query***

Contoh dari *Part-of-Speech* (POS) Tagging yang dimana dengan menerapkan modul CRF *Tagger* bertujuan untuk memberikan tag secara otomatis dan berbasis aturan untuk mempermudah dalam pelabelan kelas kata, dengan data yang digunakan adalah *query* artikel berita pariwisata yaitu terdapat pada Tabel 3. 10 dibawah ini

Tabel 3. 10 Contoh *Part-of-Speech* (POS) Tagging

<i>Query benar</i>	<i>Hasil Part-of-Speech (POS) Tagging</i>
Kesenian Kuda Kosong Meriahkan Helaran Budaya Cianjur	Kesenian/NN, kuda/NN, kosong /JJ, meriahkan/VB, helaran/NN, budaya/NN, cianjur/JJ
Bali Potret Pengelolaan Pariwisata Labuan Bajo	Bali/VB, potret/NN, pengelolaan/NN, pariwisata/NN, labuan/NN, bajo/NNP

**b. Contoh *Part-of-Speech* (POS) Tagging pada artikel berita Pariwisata**

Pada bagian ini memaparkan mengenai contoh penerapan dari ambiguitas kata dengan menggunakan *Part-of-Speech* (POS) Tagging dimana dalam contoh ini terdapat kata ambigu dengan teknik memberikan tag pada setiap kata untuk memastikan kata tersebut termasuk kata sifat atau kata benda. Berikut ini adalah tabel yang mencontohkan kata ambigu dengan teknik *Part-of-Speech* (POS) Tagging.

Tabel 3. 11 Contoh artikel *ambigu*

No.	Data Artikel Berita Pariwisata	Ambiguitas Kata	<i>Part-of-Speech</i> (POS) Tagging
1.	Peserta dari 26 kecamatan dan berbagai kelompok berkumpul di depan taman lampion Jalan Veteran	Kata “berbagai kelompok” merujuk pada peserta yang berkumpul di depan taman lampion atau pada kelompok yang berbeda-beda dari 26 kecamatan.	Apakah “berbagai” berfungsi sebagai kata sifat ( <i>adjective</i> ) yang menggambarkan “kelompok” atau sebagai kata ganti ( <i>pronoun</i> ) yang mengacu pada peserta
2.	Dari Solo kita arahkan dari simpang tiga Ngigas	Dalam konteks menggunakan <i>Part-of-Speech</i> (POS) Tagging, kata "Solo" dapat menjadi kata benda (nama kota) atau kata sifat (sendiri)?, sehingga menimbulkan ambiguitas.	Kata "Solo" dapat diinterpretasikan nama kota atau kata sifat yang berarti "sendiri".
3.	Jalan protokol yang menjadi lintasan pawai dipadati warga di kanan kirinya sejak Sabtu subuh	Kata "kanan" dan "kirinya" dapat diinterpretasikan sebagai arah atau sisi jalan (kata benda) atau sebagai kata sifat yang menggambarkan posisi (kata sifat), sehingga	Dalam konteks <i>Part-of-Speech</i> (POS) Tagging, apakah kata "kanan" dan "kirinya" dapat menjadi kata benda atau kata sifat.

No.	Data Artikel Berita Pariwisata	Ambiguitas Kata	<i>Part-of-Speech</i> (POS) Tagging
		menimbulkan ambiguitas.	
4.	Namun tak hanya budaya lokal, kebudayaan Indonesia juga turut ditampilkan seperti tarian dari Sumatera dan barongsai.	Mengenai konteks <i>Part-of-Speech</i> (POS) Tagging, apakah kata "turut" dapat menjadi kata kerja atau kata sifat?, sehingga menimbulkan ambiguitas.	Kata "turut" bisa dikatakan kata kerja ( <i>verb</i> ) yang berarti ikut serta atau menjadi bagian dari, namun juga bisa dianggap sebagai kata sifat ( <i>adjective</i> ) yang berarti sama atau sejenis.
5.	Selain itu, lanjut dia, Helaran budaya juga menjadi hiburan rakyat di Kota Santri.	Dengan menggunakan <i>Part-of-Speech</i> (POS) Tagging, apakah kata "lanjut" dapat menjadi kata kerja atau kata benda?, sehingga menimbulkan ambiguitas.	Kata "lanjut" bisa dianggap sebagai kata kerja ( <i>verb</i> ) yang berarti melanjutkan atau kata benda ( <i>pronoun</i> ) yang merujuk pada sesuatu yang berlanjut.
6.	Bupati Cianjur Herman Suherman, mengatakan helaran budaya tersebut memang rutin digelar setiap	Dalam konteks ini, kata "rutin" dapat menjadi kata benda atau kata kerja, sehingga	Menggunakan <i>Part-of-Speech</i> (POS) Tagging, jika kita Kata "rutin" dapat diinterpretasikan sebagai kata benda ( <i>pronoun</i> ) yang berarti rutinitas atau kata

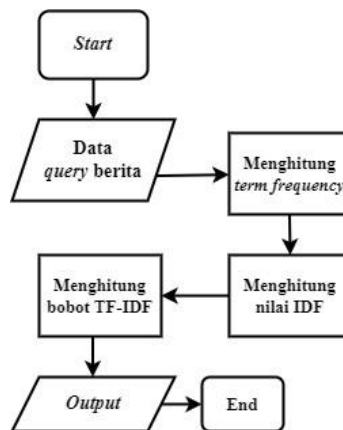


No.	Data Artikel Berita Pariwisata	Ambiguitas Kata	<i>Part-of-Speech (POS) Tagging</i>
	tahunnya dengan menampilkan beragam seni dan kebudayaan dari Kota Santri.	menimbulkan ambiguitas.	kerja ( <i>verb</i> ) yang berarti menjalankan sesuatu secara teratur.
7.	Selain itu, lanjut dia, Helaran budaya juga menjadi hiburan rakyat di Kota Santri.	Apakah kata "hiburan" dapat menjadi kata benda atau kata sifat sehingga menimbulkan ambiguitas.	Jika dalam konteks menggunakan <i>Part-of-Speech (POS) Tagging</i> , kata "hiburan" dapat diinterpretasikan sebagai kata benda ( <i>pronoun</i> ) yang merujuk pada kegiatan menghibur atau kata sifat ( <i>adjective</i> ) yang menggambarkan sesuatu yang bersifat menghibur.
8.	Demikian halnya Kepala Desa Pujon Kidul Kecamatan Pujon Kabupaten Malang, Jawa Timur yang dinilai berhasil mengembangkan sektor pariwisata sehingga dapat menyejahterahkan warganya.	Dalam konteks ini, apabila menggunakan <i>Part-of-Speech (POS) Tagging</i> , dimana kata "menyejahterahkan" dapat menjadi kata kerja atau kata benda?, sehingga menimbulkan ambiguitas.	Kata menyejahterahkan" dapat diinterpretasikan sebagai kata kerja ( <i>verb</i> ) yang berarti meningkatkan kesejahteraan atau kata benda ( <i>pronoun</i> ) yang merujuk pada keadaan sejahtera

No.	Data Artikel Berita Pariwisata	Ambiguitas Kata	<i>Part-of-Speech</i> (POS) Tagging
9.	Mungkin sekitar 3 bulan (waduk terisi) dan Desember mulai diresmikan oleh Presiden, kata Bupati Lebak Iti Octavia Jayabaya kepada wartawan di Rangkasbitung.	Kata mungkin" dapat diinterpretasikan sebagai kata keterangan ( <i>adverb</i> ) yang merujuk pada kemungkinan atau kata sifat ( <i>adjective</i> ) yang berarti tidak pasti.	Dalam konteks menggunakan <i>Part-of-Speech</i> (POS) Tagging, "mungkin" dapat menjadi kata keterangan atau kata sifat?, sehingga menimbulkan ambiguitas.
10.	Waduk ini akan menjadi suplai air baku untuk Tangerang Raya dan Jakarta.	Apakah kata "suplai" dapat menjadi kata benda atau kata kerja, sehingga menimbulkan ambiguitas	Pada konteks dengan menggunakan <i>Part-of-Speech</i> (POS) Tagging, .Kata "suplai" dapat diinterpretasikan sebagai kata benda ( <i>pronoun</i> ) yang merujuk pada pasokan atau kata kerja ( <i>verb</i> ) yang berarti menyediakan

### 3.5.4 TF-IDF dan Cosine Similarity

Pada tahapan ini, *dataset* yang sebelumnya dilakukan *preprocessing*, akan dilakukan proses TF-IDF dimana kumpulan dari dari setiap kata (*term*) akan diubah bentuk menjadi *numerical* (angka) dengan menghasilkan sebuah *matriks* vector. Hasil dari *preprocessing* diinputkan, selanjutnya dicari kemunculan kata (*term*) pada setiap dokumen. Kemudian melakukan IDF untuk mencari nilai dari IDF dengan menggunakan rumus yang ada dan dilanjutkan dengan menghitung nilai dari tf-idf untuk mencari nilai dari kumpulan *term*, yang nantinya menghasilkan bobot dari setiap kata (*term*) dalam bentuk *vector* atau angka. Berikut ini adalah Gambar 3. 7 yaitu *flowchart* TF-IDF yang terdapat dibawah ini:



Gambar 3. 7 *Flowchart* TF-IDF

Mencari *IDF* setiap kata pada dokumen dengan menghitung  $W_{IDF}$  dari kata , sehingga nantinya diperoleh nilai  $W_{IDF}$  dari kata dalam *term matriks* menggunakan rumus

$$W_{TF} = 1 + \log \frac{D}{d(t_i)}$$

1. Selanjutnya menghitung *TF-IDF* pada setiap kata dalam setiap dokumen dengan menghitung

$$W_{tf-idf}(t_i, d_i) = W_{TF}(t_i, d_i) \times W_{IDF}(t_i, d_i)$$

2. Kemudian menghitung *Cosine Similarity* setiap dokumen yang sudah didapatkan.

Penerapan metode *cosine similarity* agar dapat di proses untuk melakukan pencarian artikel berita pariwisata pada saat *query* yang diinputkan sudah benar dan menampilkan artikel berita pariwisata, sebelum dihitung tingkat kesamaannya antara artikel berita pariwisata yang satu dengan yang lainnya sesuai dengan *query* yang dicari, maka dilakukan pembobotan kata pada masing-masing dokumen artikel berita pariwisatanya, dengan menggunakan TF-IDF. Ketika *query* diproses atau diperoleh, nantinya akan dicari di dalam *inverted index*, untuk mengetahui kata apa yang sama antara *query* yang diinputkan dengan *inverted index* (dokumen yang sudah diurutkan). Sehingga untuk menemukan ataupun mendapatkan dokumen artikel dari variabel yang sudah diurutkan di *inverted index* bersama dengan *query* yang diinputkan atau dicari oleh *user*. Hasil *inverted index* yaitu dokumen artikel yang diurutkan dengan *query*, maka dilakukan perhitungan kemiripan dokumen

menggunakan *cosine similarity* yang bekerja untuk menyamakan dua buah artikel untuk mencari kesamaannya antara bobot *keyword* dokumen pada artikel berita pariwisata dengan bobot *keyword query*, maka dilakukan perhitungan kemiripan dokumen menggunakan *cosine similarity* yang bekerja untuk menyamakan dua buah artikel untuk mencari kesamaannya antara bobot *keyword* dokumen pada artikel berita pariwisata dengan bobot *keyword query*, kemudian dilakukan perangkingan. Kemudian terakhir adalah mengukur kinerja sistem yaitu evaluasi sistem dengan menghitung nilai presisi.

### 3.6 Skenario Pengujian

Skenario pengujian dilakukan untuk mengetahui hasil nilai presisi dari koreksi ejaan terhadap *query* pencarian artikel pariwisata berbahasa Indonesia dengan *Part-of-Speech (POS) Tagging* menggunakan algoritma *Levenshtein Distance*. Pada penelitian ini memerlukan uji coba sistem. Terdapat proses uji coba dalam sistem ini, yaitu uji coba pertama adalah mengetahui hasil koreksi kesalahan ejaan dari *query* artikel berita pariwisata berbahasa Indonesia menggunakan *Levenshtein Distance* dan *Part-of-Speech (POS) Tagging*. Uji coba kedua adalah mengetahui pengaruh dari *Part-of-Speech (POS) Tagging* dalam mengetahui nilai akurasi berupa *presisi* dengan algoritma *Levenshtein Distance*. Berikut ini Tabel 3. 12 untuk skenario uji coba dibawah ini.

Tabel 3. 12 Skenario Uji Coba

Skenario	Tujuan
1.	Mengetahui hasil akurasi berupa <i>presisi</i> dari koreksi ejaan dari <i>query</i> berita pariwisata berbahasa Indonesia menggunakan <i>Levenshtein Distance</i> dan <i>Part-of-Speech (POS) Tagging</i>
2.	Mengetahui hasil akurasi berupa <i>presisi</i> dari <i>query</i> berita pariwisata berbahasa Indonesia menggunakan <i>Levenshtein Distance</i> tanpa <i>Part-of-Speech (POS) Tagging</i>

### 3.7 Implementasi Dataset

Pada implementasi dataset ini akan membahas mengenai bagaimana dataset diperoleh dan digunakan untuk menghasilkan skenario uji coba dan evaluasi sistem.

#### 3.7.1 Dataset

Dataset yang digunakan dalam penelitian ini adalah dataset berita pariwisata bahasa Indonesia yang diperoleh dari *crawling* pada halaman detik.com, dengan data yang diperoleh sebanyak 332 dokumen dengan 132.689 kata. Selanjutnya dataset ini akan dihasilkan sebuah kamus yang diambil dari gabungan artikel berita pariwisata yang berjumlah 332 dokumen dengan data kamus bahasa Indonesia dari *github* dengan *link* <https://github.com/keyreply/Bahasa-Indo-NLP-Dataset> sebanyak 14.555 kata yaitu dengan data kamus baru sebanyak 23.888 kata yang disimpan di *file csv*.

Kemudian dari dataset tersebut dibuat sebuah *query* sebanyak 100 data sebagai uji coba untuk proses koreksi ejaan terhadap *query* pencarian, sehingga nantinya dari data *query* salah yang dibuat, dan disiapkan *query* benar, akan dilakukan pengecekan baik dari *query* yang diinputkan, selanjutnya dilakukan *Part-of-Speech* (POS) *Tagging*. Hasilnya nantinya akan dilakukan evaluasi berupa hasil nilai presisinya.

#### 3.7.2 Pelabelan Dataset

Pelabelan *dataset* dilakukan secara manual (dilakukan manusia) yang dibantu dengan sistem komputer, dan juga menggunakan modul *crf tagger* yang nantinya terdapat sebuah code program dengan mencakup modul *crf tagger* dari data berita pariwisata yang diperoleh. Terdapat dua jenis pelabelan dataset diantaranya sebagai berikut :

a. Pelabelan dataset *query*

Pelabelan dataset terhadap *query* ini dilakukan ketika *query* yang diinputkan sudah benar dan sudah dikoreksi kesalahan ejaannya. Sehingga hasil dari *query* benar tersebut akan dilakukan pelabelan dengan menggunakan modul *crf tagger* yaitu tag idn dari *Part-of-Speech* (POS) *Tagging*.

b. Pelabelan dataset artikel berita

Tahapan selanjutnya adalah pelabelan dataset pada artikel berita pariwisata yang dilakukan secara manual (dilakukan oleh manusia) dengan bantuan sistem komputer dan juga menggunakan modul crf tagger, dan kamus bahasa Indonesia. Dalam proses pelabelan dengan memberikan tag pada setiap kata dalam artikel berita pariwisata, dengan bantuan salah satu guru pakar Bahasa Indonesia yaitu Ibu Sinarsih, S.Pd, status pengajar di SMA Negeri 1 Torjun, Jl. Raya Torjun, Kecamatan Torjun, Kabupaten Sampang, Provinsi Jawa Timur. Pada tanggal 23 Februari 2024 telah terverifikasi atau sudah tervalidasi oleh Ibu Sinarsih, S.Pd selaku koordinator dan sebagai validator Tugas Akhir yang dipercaya sebagai pakar untuk pelabelan dataset. Hasil dataset yang dilakukan pelabelan dengan menggunakan *Part-of-Speech (POS) Tagging* yaitu dari 332 dokumen dengan 157.692 kata. Berikut ini Gambar 3. 8 adalah hasil tag kata atau pelabelan kelas kata dengan menggunakan *Part-of-Speech (POS) Tagging* yang terdapat dibawah ini.

Konten
[[('kabupaten', 'NN'), ('klaten', 'NN'), ('menyelenggarakan', 'VB'), ('karnaval', 'NN'), ('budaya', 'NN'), ('digelar', 'NN'), ('siang', 'NN'), ('jalan', 'NN'), ('utama', 'JJ'), ('kota', 'NN'), ('jenis', 'NN'), ('kese
[[('karnaval', 'NN'), ('budaya', 'NN'), ('kabupaten', 'NN'), ('klaten', 'NN'), ('digelar', 'NN'), ('siang', 'NN'), ('jalan', 'NN'), ('utama', 'JJ'), ('kota', 'NN'), ('jenis', 'NN'), ('kesenian', 'NN'), ('potensi', 'NN')]
[[('siang', 'NN'), ('karnaval', 'NN'), ('budaya', 'NN'), ('rangka', 'NN'), ('hut', 'VB'), ('ri', 'NN'), ('kabupaten', 'NN'), ('klaten', 'NN'), ('digelar', 'NN'), ('pusat', 'NN'), ('kota', 'NN'), ('klaten', 'NN'), ('dinas
[[('ribuan', 'CD'), ('warga', 'NN'), ('cianjur', 'NN'), ('padati', 'VB'), ('jalan', 'NN'), ('protokol', 'NN'), ('menyaksikan', 'VB'), ('helaran', 'NN'), ('budaya', 'NN'), ('sabtu', 'NN'), ('puluhan', 'NN'), ('kesenia
[[('uji', 'NN'), ('coba', 'VB'), ('sistem', 'NN'), ('kerja', 'NN'), ('rumah', 'NN'), ('work', 'FW'), ('from', 'FW'), ('home', 'FW'), ('with', 'FW'), ('aparatur', 'NN'), ('sipil', 'JJ'), ('negara', 'NN'), ('asn', 'NN'), ('per
[[('dinas', 'NN'), ('pariwisata', 'NN'), ('dispar', 'VB'), ('kota', 'NN'), ('palembang', 'NN'), ('menggelar', 'NN'), ('tomba', 'NN'), ('bidar', 'NN'), ('tradisional', 'JJ'), ('perahu', 'VB'), ('hias', 'NN'), ('mengu
[[('agustus', 'VB'), ('indonesia', 'NN'), ('merayakan', 'VB'), ('kemerdekaannya', 'NN'), ('detikers', 'NN'), ('daerah', 'NN'), ('mengadakan', 'VB'), ('tradisi', 'NN'), ('unik', 'JJ'), ('memperingati', 'VB'), ('k
[[('khusus', 'JJ'), ('foto', 'FW'), ('spot', 'FW'), ('maliboro', 'FW'), ('instagramable', 'FW'), ('picture', 'FW'), ('of', 'FW'), ('the', 'FW'), ('day', 'FW'), ('potd', 'FW'), ('sih', 'FW'), ('gedung', 'NN'), ('tua', 'JJ')]
[[('pemprov', 'NN'), ('sumatera', 'NN'), ('barat', 'NN'), ('sumbar', 'NN'), ('mengelar', 'NN'), ('pawai', 'NN'), ('bertajuk', 'VB'), ('merah', 'NN'), ('putih', 'VB'), ('light', 'NN'), ('carnival', 'NN'), ('kantor',
[[('monumen', 'NN'), ('ex', 'FW'), ('tentara', 'NN'), ('pelajar', 'NN'), ('pusat', 'NN'), ('kota', 'NN'), ('klaten', 'NN'), ('terawat', 'NN'), ('dikenal', 'VB'), ('diperbaiki', 'VB'), ('petugas', 'NN'), ('dinas', 'NN')]
[[('pemerintah', 'NN'), ('menggelar', 'FW'), ('the', 'FW'), ('th', 'FW'), ('ministerial', 'FW'), ('meeting', 'FW'), ('of', 'FW'), ('the', 'FW'), ('indonesia', 'FW'), ('singapore', 'FW'), ('six', 'FW'), ('bilateral', 'F
[[('dewan', 'NN'), ('pimpinan', 'NN'), ('pusat', 'NN'), ('perkumpulan', 'NN'), ('aparatur', 'NN'), ('pemerintah', 'NN'), ('desa', 'NN'), ('indonesia', 'NN'), ('dpp', 'NN'), ('paddesi', 'NN'), ('menyayangkan
[[('wali', 'NN'), ('kota', 'NN'), ('solo', 'NN'), ('gibran', 'NN'), ('rakabuming', 'NN'), ('raka', 'SC'), ('perdana', 'NN'), ('mengikuti', 'VB'), ('pawai', 'NN'), ('pembangunan', 'NN'), ('kota', 'NN'), ('solo', 'NN')]
[[('pt', 'NN'), ('waskita', 'NN'), ('karya', 'NN'), ('persero', 'NN'), ('tbk', 'NN'), ('anak', 'NN'), ('usahanya', 'RB'), ('pt', 'VB'), ('waskita', 'NN'), ('toll', 'FW'), ('road', 'FW'), ('vtr', 'FW'), ('pengoperasian', 't
[[('waduk', 'NN'), ('karian', 'NN'), ('seluas', 'JJ'), ('hektare', 'FW'), ('kabupaten', 'FW'), ('lebak', 'VB'), ('banten', 'NN'), ('diain', 'NN'), ('september', 'NN'), ('waduk', 'NN'), ('suplai', 'NN'), ('air', 'NN'),
[[('karnaval', 'NN'), ('rangka', 'NN'), ('hut', 'VB'), ('ri', 'NN'), ('kabupaten', 'NN'), ('klaten', 'NN'), ('digelar', 'NN'), ('karnaval', 'NN'), ('digelar', 'NN'), ('besok', 'NN'), ('karnaval', 'NN'), ('karnaval', 'NN
[[('indonesia', 'NN'), ('mengelar', 'NN'), ('asean', 'NN'), ('indo', 'FW'), ('pasific', 'FW'), ('forum', 'FW'), ('aipf', 'FW'), ('september', 'FW'), ('forum', 'FW'), ('flagship', 'FW'), ('event', 'FW'), ('asean', 'F
[[('pencemaran', 'NN'), ('sungai', 'NN'), ('sagea', 'NN'), ('kabupaten', 'NN'), ('halmahera', 'NN'), ('halteng', 'NN'), ('maluku', 'NN'), ('utara', 'NN'), ('malut', 'VB'), ('diduga', 'VB'), ('aktivitas', 'NN'), ('t
[[('pencemaran', 'NN'), ('sungai', 'NN'), ('sagea', 'NN'), ('kabupaten', 'NN'), ('halmahera', 'NN'), ('halteng', 'NN'), ('maluku', 'NN'), ('utara', 'NN'), ('malut', 'VB'), ('diduga', 'VB'), ('aktivitas', 'NN'), ('t
[[('provinsi', 'NN'), ('jawa', 'VB'), ('barat', 'NN'), ('jabar', 'NN'), ('keberadaannya', 'NN'), ('provinsi', 'NN'), ('tertua', 'NN'), ('jabar', 'NN'), ('berganti', 'NN'), ('gubernur', 'NN'), ('kali', 'NN'), ('tanggai
[[('menteri', 'NN'), ('perdagangan', 'NN'), ('zulkifli', 'VB'), ('hasan', 'NN'), ('alias', 'NN'), ('zulhas', 'VB'), ('menerima', 'VB'), ('keluhan', 'NN'), ('pelaku', 'NN'), ('usaha', 'NN'), ('negeri', 'NN'), ('terkait
[[('penurunan', 'NN'), ('bendera', 'NN'), ('alun', 'FW'), ('alun', 'FW'), ('jember', 'FW'), ('memperingati', 'VB'), ('hut', 'NN'), ('kemerdekaan', 'NN'), ('ri', 'NN'), ('ditutup', 'VB'), ('pagelaran', 'NN'), ('dr

Gambar 3. 8 *Part-Of-Speech (POS) Tagging* Data Artikel Pariwisata

### 3.7.3 Visualisasi Data dengan *Word Cloud*

Pada bagian ini yaitu melakukan visualisasi data dimana tujuan adanya visualisasi untuk mengekstrak sebuah informasi dalam bentuk topik seperti: data artikel berita pariwisata, dengan banyaknya teks konten artikel yang tersedia sehingga mendapatkan informasi yang menurut penting. Penelitian ini menggunakan sebuah *word cloud* yang digunakan untuk memvisualisasikan hasil data konten berita pariwisata. Adanya *word cloud* menjadi representasi sebuah data yang menampilkan kumpulan kata yang sering muncul dan penting sehingga



### *3. Tokenization*

Tahapan ini berfungsi dalam menguraikan sebuah kalimat yang menjadi kata perkata, berdasarkan karakter ‘spasi’ sebagai tanda pemisahannya.



### 3.8 Perkiraan Jadwal

Adapun jadwal yang dibuat dalam melakukan penelitian ini yaitu terdapat pada Tabel 3. 13

Tabel 3. 13 Jadwal Penelitian

Baris ke	Kegiatan	BULAN																							
		Bulan-1				Bulan-2				Bulan-3				Bulan-4				Bulan-5				Bulan-6			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Pengumpulan data dan Studi literatur																								
2	Analisis permasalahan																								
3	Perancangan Sistem																								
4	Implementasi Sistem																								
5	Ujicoba Sistem																								
6	Analisa dan Evaluasi																								
7	Penyusunan Laporan																								
8	Dokumentasi																								

## BAB IV

### HASIL DAN PEMBAHASAN

Pada bab ini membahas mengenai pengimplementasian sistem dan proses pengujian sistem yang telah diimplementasikan sistem berdasarkan perancangan dan metodologi yang telah dijelaskan pada bab sebelumnya serta pembahasan dari hasil pengujian sistem. Terdapat beberapa tahapan yang akan dilakukan dalam proses ini, yaitu lingkungan uji coba, pelabelan data, *text preprocessing*, pengujian sistem, mulai dari pengujian sistem terhadap koreksi ejaan maupun terhadap temu kembali informasi (*information retrieval*), beserta analisis dan evaluasi sistem yang dihasilkan

#### 4.1 Lingkungan Uji Coba

Pada bagian ini, menjelaskan mengenai ruang lingkup dalam implementasi uji coba sebuah program yang telah dibuat. Tahapan ini membahas mengenai sebuah sistem yang diuraikan mengenai perangkat lunak maupun perangkat keras yang digunakan dalam mengimplementasikan penelitian dari awal pengumpulan data sampai pengujian sistem. Terdapat Tabel 4. 1 dibawah ini adalah sebuah tabel lingkungan uji coba sistem atau program yang akan diterapkan.

Tabel 4. 1 Spesifikasi Perangkat Keras

No.	Kebutuhan	Spesifikasi
1.	CPU	AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz
2.	Sistem Operasi	Windows 11 Home Single Language
2	CPU	AMD Ryzen 5
4	RAM	8,00 GB (7,33 GB usable)

Tabel 4. 2 Spesifikasi Perangkat Lunak

No.	Kebutuhan	Versi	Fungsi
1	Visual Studio Code	3.9.1	Berfungsi untuk <i>text editor</i> yang digunakan untuk menulis bahasa program, maupun meng-edit <i>file</i> pemrograman serta mengeksekusi <i>code</i> atau menjalankan aplikasi.
2	Xampp	v3.3.0	Sebagai <i>Web Server</i> lintas <i>platform open-source</i> yang terdiri dari <i>apache</i> HTTP server, database MySQL Mengolah dan membaca dataset kamus dalam bentuk tabel.
3.	Google Colabority	Reguler	Berfungsi untuk menulis, menjalankan, dan berbagi kode python melalui <i>web browser</i> .
4.	Python	3.10.12	Berfungsi sebagai salah satu bahasa pemrograman yang digunakan untuk implementasi pada penelitian ini.
3	pandas	1.4.2	<i>Library</i> yang digunakan untuk menganalisis, memuat, dan menampilkan data.
4	numpy	1.22.4	Berfungsi untuk melakukan manipulasi dalam sebuah <i>matriks</i> dan operasi matematika.
5.	Sastrawi	1.0.1	<i>Library</i> yang berfungsi sebagai pustaka python sederhana yang nantinya memungkinkan untuk mereduksi kata-kata infleksi dalam Bahasa Indonesia ke bentuk dasarnya.
6.	NLTK	3.7	<i>Library</i> yang berfungsi dan digunakan dalam pemrosesan data, salah satunya yaitu dalam proses tokenisasi data.

No.	Kebutuhan	Versi	Fungsi
8.	<i>Streamlite</i>	1.15.1	Sebuah <i>framework</i> berbasis python dan bersifat <i>open-source</i> yang dibuat untuk memudahkan dalam membangun sebuah aplikasi web.
9.	<i>streamlit-option-menu</i>	0.3.2	Berfungsi untuk memungkinkan <i>user</i> memilih satu item dari daftar opsi dalam menu.

## 4.2 Tahapan Visualisasi Kata

Pada tahapan ini dilakukan proses visualisasi yang bertujuan untuk mengekstrak informasi dalam bentuk topik. Penelitian ini menggunakan *word cloud* yaitu memvisualisasikan hasil data artikel pariwisata pada bagian konten. Data artikel pariwisata dilakukan indentifikasi berdasarkan banyaknya frekuensi kata dalam konten artikel pariwisata. Berikut ini adalah fungsi dan potongan dari penerapan *word cloud* untuk melihat sekaligus merepresentasikan data dengan menampilkan kumpulan kata yang paling banyak muncul dan penting yaitu sebagai berikut

### 4.2.1 Visualisasi Data dengan *WordCloud*

```

482 st.write("""## Visualisasi""") #menampilkan judul
483 halaman dataframe
    uploaded_files = st.file_uploader("Please choose a
484 CSV file", type=['csv'])
485     if uploaded_files is not None:
486         data = pd.read_csv(uploaded_files,
error_bad_lines=False)
487         all_text = ' '.join(data['Konten'])
488
489         # Try using default font or specify a font
family
490         font_path =
r"C:\xampp\htdocs\RISSET_DIA\AGENCYB.TTF" # Use default
font or specify a font file path
491         # Tetapkan family font sebagai 'AGENCYB'
492         #font_family = 'AGENCYB'
493
494         # Buat objek WordCloud dengan menyertakan
parameter font_path dan font_family

```

```

495     wordcloud = WordCloud(width=800, height=400,
background_color='white',
font_path=font_path).generate(all_text)
496
497     # Display WordCloud
498     plt.figure(figsize=(10, 5))
499     plt.imshow(wordcloud, interpolation='bilinear')
500     plt.axis('off')
501     plt.show()
502     # Tampilkan WordCloud menggunakan streamlit
503     st.image(wordcloud.to_array(),
use_column_width=True)

```

Kode Program 4. 1 Inisiasi Variabel Visualisasi Data dengan *WordCloud*

### Penjelasan Program:

1. Menampilkan Judul Halaman yaitu dengan menggunakan kode `st.write("""## Visualisasi""")`
2. Pada kode program baris 484 yaitu untuk mengunggah file CSV dan menampilkan sebuah komponen yang di unggah dengan pesan “Please choose a CSV file” dan hanya menerima file dengan format CSV.
3. Pada kode program baris 485 digunakan unruk memeriksa file yang telah di upload. Apabila ada akan dilakukan blok untuk kode yang akan dieksekusi.
4. Pada kode program baris ke 486 untuk membaca file CSV dengan menggabungkan teksnya dalam DataFrame `data`. Parameter berguna untuk mengabaikan baris yang bermasalah. Kemudian pada baris ke 487 menggabungkan semua teks yang tersimpan dalam kolom “Konten” dari dataframe `data` menjadi satu string yang besar yang dipisahkan oleh spasi, nantinya akan disimpan dalam `all_text`
5. Selanjutnya baris ke 490 menentukan *Path Font* yang dimana untuk menetapkan sebuah path ke file *font* yang akan digunakan dakam *word cloud*
6. Kemudian, baris ke 495 kode `wordcloud` yang digunakan untuk membuat sebuah objek *word cloud* yang lebar = 800 piksel, tinggi = 400 piksel, dengan latar belakang berwarna putih, dan membuat sebuah font oleh `font_path`. Pada kode `generate(all_text)` yang nantinya menghasilkan *word cloud* berdasarkan sebuah teks yang digabungkan dalam sebuah `all_text`

7. Pada kode program baris ke 498 sampai baris ke 501 menampilkan sebuah *word cloud* menggunakan Matplotlib dengan beberapa fungsi dibawah ini
  - `plt.figure(figsize=(10, 5))` digunakan untuk membuat *figure* (plot) ukuran 10 inci x 5 inci
  - `plt.imshow(wordcloud, interpolation='bilinear')` berfungsi untuk menampilkan gambar *WordCloud* dengan interpolasi bilinear dalam membuat tampilan lebih halus
  - `plt.axis('off')` `plt.show()` digunakan untuk mebaris ke naktifkan sumbu plot yang sudah dibuat
  - `plt.show()` digunakan untuk menampilkan plot yang sudah dibuat.
8. Pada kode program baris ke 503, berfungsi untuk menampilkan hasil *Word Cloud* dengan menggunakan *streamlit*.
9. Sehingga proses yang digunakan untuk menampilkan hasil visualisasi data terdapat pada gambar 3.9 yaitu *Word Cloud* visualisasi artikel berita pariwisata.

#### 4.2.2 Tahapan *Preprocessing* Data

Pada tahapan *preprocessing* data ini, sebelum melakukan proses koreksi kesalahan ejaan terhadap *query* pencarian *Search Engine* Bahasa Indonesia (SEBI), perlu dilakukan tahapan *preprocessing* terlebih dahulu. Tujuannya, untuk membersihkan data menjadi terstruktur dan berbobot, sehingga data akan dengan mudah diproses pada tahapan selanjutnya. Pada *preprocessing* ini akan dilakukan dengan bantuan *library* python, berikut ini adalah tahapan dari *preprocessing* yang digunakan.

##### 1. *Case Folding*

Tahapan pertama yang dilakukan dalam *text preprocessing* yaitu proses *case folding*, mengubah huruf kapital menjadi huruf kecil.

```
524 user_input = user_input.lower()
```

Pada code program baris 524 tersebut teks *input* diubah menjadi huruf kecil menggunakan `lower()`.

##### 2. *Cleaning*

Pada tahapan ini digunakan untuk membersihkan sebuah karakter tertentu yang tidak dibutuhkan seperti simbol dan angka. Berikut ini *code* programnya

```

525 clean =[]
526         for i in range (len(user_input)):
527             clean_result = re.sub("@[A-Za-z0-
528 9_]+","", user_input[i]) #cleansing mention
528             clean_result1 = re.sub("#[A-Za-z0-
529 9_]+","", clean_result) #cleansing hashtag
529             clean_result2 = re.sub(r'http\S+', '',
clean_result1) #cleansing url link
530             clean_result3 = re.sub("[^a-zA-Z ]+","",
", clean_result2) #cleansing character
                    clean.append(clean_result3)

```

Kode Program 4. 2 Cleaning

1. Pada baris ke 525 sampai baris ke 527 yaitu `for i in range (len(user_input)):` untuk melakukan *looping* (perulangan) yang akan berjalan sebanyak karakter yang ada dalam *string* `user_input`.  
`clean_result = re.sub("@[A-Za-z0-9_]+","", user_input[i])` *#cleansing mention* berguna untuk menghapus mention mulai dari @ yang diikuti dengan huruf, angka, maupun garis bawah, sehingga nantinya karakter yang terdapat dalam `user_input` yang hasilnya disimpan dalam `clean_result`.

2. Kemudian baris ke 528 `clean_result1 = re.sub("#[A-Za-z0-9_]+","", clean_result)` *#cleansing hashtag* yang digunakan untuk membersihkan teks yang dimana untuk menghapus semua *hashtag* yang dimulai dari tanda # yang diikuti dengan huruf, angka atau garis bawah. Selanjutnya baris ke 529 `clean_result2 = re.sub(r'http\S+', '', clean_result1)` *#cleansing url link* berguna untuk memberishkan teks yang berfungsi untuk menghapus semua url yang dimulai dengan 'http' yang disimpan dalam `clean_result2`. Dan baris ke 530 `clean_result3 = re.sub("[^a-zA-Z ]+","", clean_result2)` *#cleansing character* dan baris ke 531 `clean.append(clean_result3)`, setelah membersihkan semua elemen yang terdapat dalam teks dalam daftar *clean* yang nantinya mempunyai tujuan untuk menyimpan semua karakter yang telah dibersihkan dan dilakukan analisis selanjutnya.

### 3. Tokenization

Tahapan ini berfungsi dalam menguraikan sebuah kalimat yang menjadi kata perkata, berdasarkan karakter 'spasi' sebagai tanda pemisahannya, berikut ini *code* programnya.

```
533 tokenize = user_input.split()
```

#### 4.3 Implementasi *Function* Program

Pada bagian ini terdapat beberapa *function* yang akan dibuat dalam implementasi program koreksi ejaan terhadap *query* pencarian artikel pariwisata berbahasa Indonesia menggunakan metode *Levenshtein Distance* dan *Part-Of-Speech* (POS) *Tagging* yang dapat dilihat pada kode program dibawah ini, diantaranya sebagai berikut:

##### 4.3.1 *Levenshtein Distance* dan *Part-Of-Speech* (POS) *Tagging*

```
#Fungsi Algoritma Levenshtein Distance
26 def levenshtein_distance(s1, s2):
27     m, n = len(s1), len(s2)
28     dp = [[0] * (n + 1) for _ in range(m + 1)]
29
30     for i in range(m + 1):
31         dp[i][0] = i
32     for j in range(n + 1):
33         dp[0][j] = j
34
35     for i in range(1, m + 1):
36         for j in range(1, n + 1):
37             cost = 0 if s1[i - 1] == s2[j - 1] else 1
38             dp[i][j] = min(dp[i - 1][j] + 1, dp[i][j - 1]
39 + 1, dp[i - 1][j - 1] + cost)
40     return dp[m][n]
```

Kode Program 4. 3 Inisiasi Fungsi Metode *Levenshtein Distance*

##### Penjelasan Program :

1. Pada kode program baris ke 26 sampai baris ke 28 adalah fungsi `levenshtein_distance` dengan definisi fungsinya sebagai berikut :
  - Fungsi `levenshtein_distance` digunakan untuk menerima dua *string* yaitu `s1`, `s2`
  - Variabel `m`, `n` digunakan untuk panjang dari *string* `s1`, `s2`
  - Variabel `dp` digunakan untuk menyimpan hasil perhitungan jarak edit *distance* dengan tabel 2D yang berukuran  $(n + 1) \times (m + 1)$  dengan inisiasi 0.
2. Pada kode program baris ke 30 sampai baris ke 33 adalah inisiasi dari tabel DP dengan inisiasi sebagai berikut :
  - Pada dua *loop* pertama digunakan untuk mengisi baris pertama dan kolom pertama pada tabel `dp`



- Pada `dp[i][0] = i` digunakan untuk membuat jarak edit dari *string* `s1` ke *string* kosong untuk menghapus semua karakter dari `s1`
  - Pada `dp[i][0] = j` digunakan untuk membuat sebuah jarak edit dari *string* `s2` untuk melakukan penyisipan dalam semua karakter dari `s2`
3. Pada kode program baris ke 35 sampai baris ke 38 melakukan proses inisiasi untuk mengisi tabel DP yaitu sebagai berikut:
- Pada proses baris ke 35 dan baris ke 36 melakukan proses mengisi tabel `dp` yaitu `i = 1` dan `j = 1`, `i = m` dan `j = m`
  - Pada proses baris ke 37 terdapat variabel `cost = 0` atau `i` jika berbeda, dimana jika ada karakter `s1[i - 1]` sama dengan `s1[i - 1]`. Proses ini adalah operasi substitusi apabila dibutuhkan
  - Pada proses baris ke 38 yaitu mengisi nilai minimum dari 3 operasi yang terdapat dalam variabel `dp[i][j]`. Diantaranya terdapat operasi penghapusan (`(dp[i - 1][j] + 1)`), penyisipan (`(dp[i][j - 1] + 1)`), dan substitusi (`(dp[i - 1][j - 1])`) atau tidak ada operasi apapun jika sama.
4. Pada kode program baris ke 40 digunakan untuk mengembalikan nilai `dp[m][n]` yang dimana adalah jarak edit antara `s1`, `s2`.

#### 4.3.2 Fungsi Koreksi Ejaan

```

78 def correct_spelling(input_word, word_list):
79     min_distance = float('inf')
80     corrected_word = input_word
81
82     for word in word_list:
83         if input_word == word:
84             return input_word
85
86         for i in range(len(input_word)):
87             edited_word = input_word[:i] +
88 input_word[i+1:]
89             distance = levenshtein_distance(edited_word,
90 word)
91             if distance < min_distance:
92                 min_distance = distance
93                 corrected_word = word
94
95         for i in range(len(input_word) + 1):
96             for char in 'abcdefghijklmnopqrstuvwxyz':

```

```

96         edited_word = input_word[:i] + char +
input_word[i:]
97         distance =
levenshtein_distance(edited_word, word)
98         if distance < min_distance:
99             min_distance = distance
100             corrected_word = word
101
102     for i in range(len(input_word)):
103         for char in 'abcdefghijklmbaris ke
pqrstuvwxyz':
104             edited_word = input_word[:i] + char +
input_word[i+1:]
105             distance =
levenshtein_distance(edited_word, word)
106             if distance < min_distance:
107                 min_distance = distance
108                 corrected_word = word
109
110     return corrected_word

```

Kode Program 4. 4 Koreksi Ejaan

#### Penjelasan Program:

1. Pada kode program baris ke 79 membuat fungsi `correct_spelling` digunakan untuk menerima dua parameter diantaranya `input_word`, kata yang diperbaiki dan `word_list`) kata yang nantinya akan dibandingkan untuk menemukan kata yang benar.
2. Pada kode program baris ke 80 terdapat variabel `min_distance` dengan inisiasi nilai tak hingga `= float('inf')`, digunakan untuk menyimpan jarak edit minimum yang ditemukan selama perbandingan.
3. Pada kode program baris ke 81 terdapat variabel `corrected_word` digunakan untuk menyimpan kata paling mirip berdasarkan jarak edit minimum.
4. Pada kode program baris ke 83 sampai baris ke 85 adalah proses memeriksa kata yang terdapat dalam `word_list`. Apabila `input_word` sama dengan kata dalam `word_list` nantinya fungsi ini akan mengembalikan `input_word` dimana kata-kata tersebut sudah benar.
5. Pada kode program baris ke 87 sampai baris ke 89 digunakan untuk operasi penghapusan yaitu melakukan loop pertama iterasi untuk setiap indeks pada `input_word`, selanjutnya terdapat variabel `edited_word`

berguna untuk menghapus karakter pada indeks  $i$  dalam `input_word`, kemudian nantinya akan dihitung didalam variabel `distance`.

6. Pada kode program baris ke 98 sampai baris ke 100 berfungsi jika `distance` lebih kecil dari `min_distance`, nantinya `min_distance` diperbarui dengan `distance`, dan `corrected_word` akan diperbarui dengan `word`
7. Pada kode program baris ke 102 sampai baris ke 105 digunakan untuk operasi penyisipan, dimana dalam `input_word` akan melakukan loop iterasi melalui setiap indeks yang nantinya iterasi tersebut juga akan melalui setiap karakter alfabet, sehingga variabel `edited_word` dibuat dengan menyisipkan `char` pada indeks `i` dari `input_word`.
8. Pada kode program baris ke 106 sampai baris ke 108 digunakan untuk operasi penggantian atau substitusi yaitu dengan melakukan loop melalui indeks dalam `input_word` yang nantinya setelah menggantikan karakter akan dihitung variabel `distance` antara `edited_word` dan `word`.
9. Pada kode program baris ke 110 digunakan untuk mengembalikan kata yang diperbaiki yaitu kata yang paling mirip dari `input_word` berdasarkan jarak edit minimum.

#### 4.3.3 Fungsi Modul dalam *Part-Of-Speech (POS) Tagging*

541	<code>data = CRFTagger()</code>
542	<code>model_path =</code> <code>'/xampp/htdocs/SKRIPSI_DIA/all_indo_man_tag_corpus_model.crf.tagger'</code>
544	<code>data.set_model_file(model_path)</code>

Kode Program 4. 5 Modul *Part-Of-Speech (POS) Tagging*

##### Penjelasan Program:

Pada kode program baris ke 541 sampai baris ke 544 digunakan untuk mengimplementasikan modul dengan menggunakan `CRFTagger` dan corpus `all_indo_man_tag_corpus_model.crf.tagger` yang disimpan dalam `model_path`.

#### 4.3.4 Fungsi Menghitung Nilai Presisi

433	<code>def calculate_precision(test_data, dictionary):</code>
-----	--

```

434     true_positives = 0
435     false_positives = 0
436
437     for test_word, true_word in test_data:
438         corrected_word = correct_spelling(test_word,
439 dictionary)
440
441         if test_word == true_word: # Kata asli sudah
benar
442             if corrected_word != test_word:
443                 false_positives += 1
444             else: # Kata asli salah
445                 if corrected_word == true_word:
446                     true_positives += 1
447                 else:
448                     false_positives += 1
449
450         precision = true_positives / (true_positives +
false_positives) if (true_positives + false_positives) >
0 else 0
451
452     return precision

```

Kode Program 4. 6 Inisiasi Fungsi Nilai Presisi

#### Penjelasan Program:

1. Pada kode program baris ke 433 melakukan fungsi yang dinamakan dengan `calculate_precision` dan menerima dua parameter yaitu
  - Parameter `test_data` adalah pasangan kata yaitu dari `test_word`, `true_word`, Dimana `test_word` yaitu kata yang di uji dan `true_word` kata yang benar.
  - Parameter `dictionary` digunakan sebagai kamus untuk koreksi ejaan.
2. Pada kode program baris ke 434 sampai ke 435 melakukan inisiasi melalui data uji dimana terdapat dua variabel `true_positives` dan `false_positif` untuk menghitung jumlah kasus positif benar dan positif salah.
3. Pada kode program baris ke 437 sampai ke 438 melakukan looping dari `test_word`, `true_word`, yang terdapat dalam `test_data`. Sedangkan `corrected_word` yaitu hasil fungsi dari `correct_spelling` yang mencoba memperbaiki `test_word` dengan menggunakan `dictionary`.
4. Pada kode program baris ke 440 sampai baris ke 442, yaitu apabila `test_word` sudah benar atau sama dengan `true_word`, tetapi sistem

koreksi mengubahnya, nantinya akan dihitung sebagai `false_positives`.

5. Pada kode program baris ke 433 sampai baris ke 447 yaitu melakukan penentuan positif benar dan positif salah, apabila `test_word` salah maka:
  - Apabila koreksi sistem sama dengan `true_word`, maka hal ini dihitung sebagai `true_positives`
  - Apabila salah maka akan dihitung sebagai `false_positives`.
6. Pada kode program baris ke 449 yaitu melakukan perhitungan nilai presisi dimana presisi dihitung sebagai rasio `true_positives` terhadap total positif (`true_positives + false_positives`). Jika tidak ada positif atau denominator nol dimana nilai presisi diatur menjadi 0.
7. Pada kode program baris 451 melakukan proses pengembalian nilai presisi

#### 4.3.5 Fungsi TF-IDF

```
201 def calculate_tfidf_from_database():
202     corpus = fetch_text_from_database()
203     vectorizer = TfidfVectorizer()
204     tfidf_matrix = vectorizer.fit_transform(corpus)
205     return tfidf_matrix, vectorizer
```

Kode Program 4. 7 Inisiasi Fungsi TF-IDF

##### Penjelasan Program:

1. Pada kode program baris ke 201 membuat fungsi `calculate_tfidf_from_database()` yang digunakan untuk menghitung nilai TF-IDF dari kumpulan teks yang diambil dari database.
2. Pada kode program baris ke 202 berfungsi untuk mengambil teks dari database yang dimana terdapat variabel `corpus` untuk menyimpan kumpulan teks yang disimpan dari database, dimana terdapat fungsi `fetch_text_from_database()` untuk mengambil teks dari database dan mengembalikan dalam bentuk list atau array.
3. Pada kode baris ke 203 digunakan untuk membuat objek TF-IDF *Vectorizer* untuk mengubah kumpulan dokumen teks menjadi representasi matriks TF-IDF dan memiliki bobot pada setiap kata dalam dokumen.

4. Pada kode baris ke 204 digunakan untuk menghitung matriks TF-IDF dengan menggunakan `fit_transform(corpus)`, dimana `fit` berfungsi untuk memahami kosakata dan `idf` dari `corpus`. Dan `transform` untuk mengubah `corpus` menjadi matriks TF-IDF.
5. Pada kode program baris ke 205 berfungsi untuk mengembalikan hasil dari dua objek yaitu `tfidf_matrix`, `vectorizer`.

#### 4.3.6 Fungsi *Inverted Index*

```

208 def create_inverted_index_from_database():
209     corpus = fetch_text_from_database()
210     inverted_index = {}
211     for i, doc in enumerate(corpus):
212         words = set(doc.split())
213         for word in words:
214             if word not in inverted_index:
215                 inverted_index[word] = [i]
216             else:
217                 inverted_index[word].append(i)
218     return inverted_index

```

Kode Program 4. 8 Inisiasi Fungsi *Inverted Index*

##### Penjelasan Program:

1. Pada kode program baris ke 208 digunakan untuk membuat fungsi *inverted index* dari kumpulan teks yang diambil dalam database.
2. Pada kode program baris ke 209 digunakan untuk mengembalikan teks data database, yang diambil dari variabel `corpus`. Kemudian mengembalikan dalam bentuk list atau array.
3. Pada kode program baris ke 210 berfungsi untuk menginisiasi *inverted index* yang digunakan untuk menyimpan kata kunci dan daftar indeks sebuah dokumen sebagai nilai.
4. Pada kode program baris ke 211 sampai baris ke 212 untuk membuat *inverted index* dimana melakukan loop pertama pada `for i, doc in enumerate(corpus)`, kemudian mengambil kata-kata unik dari sebuah dokumen yang dipecah menjadi kata-kata, dan `set` disini untuk mendapatkan kata unik dan menghindari duplikasi dalam *inverted index*.
5. Pada kode program baris ke 215 sampai baris ke 217 untuk membuat *inverted index*, dimana loop ini digunakan untuk mengiterasi setiap kata unik `words` dalam dokumen. Apabila kata tidak ada dalam

`inverted_index`, maka nantinya akan dibuat sebuah entri baru dimana dengan kata sebagai kata kunci dan terdapat nilai berupa daftar yang berisi indeks dokumen ke `[i]`. Jika kata sudah terdapat dalam `inverted_index`, indeks dokumen saat ini nantinya akan ditambahkan ke dalam daftar dokumen yang mengandung kata tersebut.

6. Pada kode program, baris ke 218 digunakan untuk mengembalikan *inverted index* dimana kunci adalah kata-kata dan nilai adalah daftar indeks dokumen

#### 4.3.7 Fungsi Pencarian Artikel Berita Berdasarkan *Cosine Similarity*

```
224 def search_news(query, tfidf_matrix, vectorizer,
225                 documents):
226     query_vec = vectorizer.transform([query])
227     cosine_similarities = cosine_similarity(query_vec,
228                                             tfidf_matrix).flatten()
229     related_docs_indices =
230     cosine_similarities.argsort()[::-1]
231     results = []
232
233     for i in related_docs_indices:
234         results.append((documents[i],
235                        cosine_similarities[i]))
236     return results
```

Kode Program 4. 9 Inisiasi Fungsi Pencarian Artikel Menggunakan *Cosine Similarity*

##### Penjelasan Program:

1. Pada kode baris ke 224 digunakan untuk membuat fungsi `search_news` untuk menerima 4 parameter diantaranya `query`, `tfidf_matrix`, `vectorizer`, `documents`.
2. Pada kode program baris ke 225 digunakan untuk melakukan transformasi `query` ke vector TF-IDF, dimana terdapat variabel `query_vec` dengan parameter `vectorizer.transform([query])` mengubah kueri pengguna menjadi vector TF-IDF.
3. Pada kode program baris ke 226 berfungsi untuk menghitung kesamaan kosinus dengan menggunakan *cosine similarity*, dimana untuk menghitung kesamaan kosinus dengan vector `query` dan vector dokumen dalam tf-idf matriks menggunakan `cosine_similarity(query_vec, tfidf_matrix)` kemudian nantinya akan diubah menjadi array Id.

4. Pada kode program baris ke 227 dan baris ke 228 digunakan untuk mengurutkan indeks dokumen berdasarkan kesamaan, dimana terdapat variabel `related_docs_indices` *array* yang berisi indeks dokumen, dilanjutkan dengan mengembalikan indeks yang akan mengurutkan *array* `cosine_similarity` dan `[::-1]` untuk mengebalikan urutan yang nantinya indeks yang kesamaannya tertinggi berada di awal.
5. Pada kode program baris ke 229 sampai baris ke 231 digunakan untuk mengumpulkan hasil pencarian, dimana terdapat daftar untuk menyimpan dokumen dengan nilai kesamaannya dalam variabel `results`, dan melakukan *looping* untuk setiap indeks dokumen yang diurutkan, sehingga hasil kesamaannya akan ditambahkan ke dalam variabel `results`.

#### 4.4 Implementasi Program Koreksi Ejaan

##### 4.4.1 Implementasi Koreksi Ejaan terhadap *Query* Menggunakan *Levenshtein Distance*

```

742 st.title("Skenario Levenshtein Distance Tanpa POS Tagging ")
743     col1, col2 = st.columns([3, 1]) # Adjust the column
      ratios as needed
744     # Kunci unik untuk form masukan
745     text_key = "koreksi_input"
746     user_input = col1.text_input("Masukkan Query",
key=text_key)
747     col2.write("")
748     col2.write("")
749     # Menyediakan kunci unik untuk tombol
750     button_key = "koreksi_button"
751     periksa = col2.button("Koreksi Query",
key=button_key)
752     if periksa:
753         #st.write(user_input)
754         if user_input == "":
755             st.warning("silahkan masukan")
756         else:
757             user_input = user_input.lower()
758             clean = []
759             for i in range (len(user_input)):
760                 clean_result = re.sub("@[A-Za-z0-
9_]+","", user_input[i]) #clenasing mention
761                 clean_result1 = re.sub("#[A-Za-z0-
9_]+","", clean_result) #clenasing hashtag
762                 clean_result2 = re.sub(r'http\S+', '',
clean_result1) #cleansing url link
763                 clean_result3 = re.sub("[^a-zA-Z ]+","",
clean_result2) #cleansing character

```



764	<code>clean.append(clean_result3)</code>
765	
766	<code>tokenize = user_input.split()</code>
767	
768	<code>result = ""</code>
769	<code>for input_word in tokenize:</code>
770	<code>corrected_word =</code>
771	<code>correct_spelling_deletion(input_word)</code>
772	<code>result += str(corrected_word) + " "</code>
773	<code>st.success(f"hasil correction: {result}")</code>
774	<code>hasil1 =result.split()</code>

Kode Program 4. 10 Implementasi Sistem Koreksi Ejaan Menggunakan Algoritma *Levenshtein Distance* tanpa *Part-of-Speech (POS) Tagging*

**Penjelasan Program :**

1. Pada kode program baris ke 742 berfungsi menampilkan judul aplikasi atau sistem.
2. Pada kode program baris ke 743 sampai baris ke 751 digunakan untuk membuat kolom inputan dan tombol koreksi.
3. Pada kode program baris ke 752 sampai baris ke 756 digunakan untuk melakukan validasi input pengguna, dengan memeriksa apakah inputan pengguna kosong, apabila kosong akan menampilkan peringatan, jika tidak kosong, akan dilanjutkan ke blok `else`.
4. Pada kode program baris ke 757 sampai baris ke 764 digunakan untuk melakukan *cleaning* atau pembersihan data dari *inputan* pengguna.
5. Pada kode program baris ke 766 sampai baris ke 771, digunakan untuk melakukan tokenisasi dari *input* yang dibersihkan menjadi token (kata-kata). Selanjutnya melakukan koreksi setiap kata dengan menggunakan fungsi `correct_spelling_deletion`, dan menggabungkan kata yang dikoreksi menjadi satu *string*.
6. Pada kode program baris ke 773 sampai baris ke 774 digunakan untuk menampilkan hasil koreksi pengguna

**4.4.2 Implementasi Koreksi Ejaan terhadap *Query* Menggunakan *Levenshtein Distance* dan *Part-Of-Speech (POS) Tagging***

541	<code>data = CRFTagger()</code>
	<code>model_path</code>
542	<code>= '/xampp/htdocs/RISET_DIA/all_indo_man_tag_corpus_model.crf.tagger'</code>

543	<code>hasil1 = result.split()</code>
544	<code>data.set_model_file(model_path)</code>
545	<code>judul = data.tag_sents([result.split()])</code>
546	<code>st.success(f"hasil Part-of-Speech (POS)</code> <code>Tagging:{judul}")</code>

Kode Program 4. 11 Koreksi Ejaan Terhadap *query* dan *Part-Of-Speech* (POS) *Tagging*

#### Penjelasan Program :

1. Pada kode program baris ke 541 berfungsi untuk inisiasi objek CRFTagger yang berbasis *Conditional Random Fields* (CRF) yang digunakan untuk melakukan *Part-of-Speech* (POS) *Tagging*.
2. Pada kode program baris ke 542 untuk menetapkan path atau lokasi dari file model CRF yang digunakan untuk melakukan POS *Tagging*.
3. Pada kode program baris ke 543 digunakan untuk memecah teks *input* yaitu dengan menjadi daftar kata, dan memisahkan *string* berdasarkan spasi.
4. Pada kode baris ke 544 sampai baris ke 545 memuat model POS *Tagging* pada bagian judul, sehingga nantinya ketika sudah dihasilkan *inputan* benar, akan dilakukan *Part-of-Speech* (POS) *Tagging* menggunakan `tag_sents`.
5. Pada kode program baris ke 547 digunakan untuk menampilkan hasil judul yang dilakukan *Part-of-Speech* (POS) *Tagging*.

#### 4.4.3 Implementasi Koreksi Ejaan terhadap *Query* Pencarian Artikel Pariwisata Menggunakan *Levenhstein Distance* dan *Part-Of-Speech* (POS) *Tagging*

834	<code>mycursor.execute(f"SELECT * FROM</code> <code>data_berita_pariwisata_2 WHERE judul LIKE '%{result}%'</code> <code>or konten LIKE '%{result}%'")</code>
835	<code>judul_list = mycursor.fetchall()</code>
836	<code>#st.write(f"judul_list")</code>
837	<code>mycursor.close()</code>
838	<code>if len(judul_list) == 0:</code>
839	<code>st.warning("Tidak Ada Berita")</code>
840	<code>else:</code>
841	<code># Ambil konten artikel dari judul_list</code>
842	<code>corpus = [judul[4] for judul in</code> <code>judul_list]</code>

Kode Program 4. 12 *query* Pencarian Artikel Pariwisata *Levenhstein Distance* dan *Part-Of-Speech* (POS) *Tagging*

### Penjelasan Kode Program:

1. Pada kode baris ke 834 digunakan untuk menjalankan *query* SQL untuk mengambil semua data dari tabel.
2. Pada kode baris ke 835 digunakan untuk mengambil hasil *query* yang dimana menyimpan hasil *query* dari variabel `judul_list`.
3. Pada kode baris ke 837 digunakan untuk menutup koneksi *cursor* untuk mengosongkan sumber daya.
4. Pada kode baris ke 838 sampai baris ke 840 digunakan untuk mengecek hasil *query*, apabila tidak ada data akan menampilkan pesan peringatan “Tidak Ada Berita”, sebaliknya pada kode baris ke 841 dan baris ke 842 jika data ditemukan akan mengambil konten artikel dari hasil *query* dan menyimpan dalam variabel `corpus`.

```
665 displayed_documents = []
666         judul_tersimpan = []
667         for i in
related_docs_indices[:min(5, len(judul_list))]:
668             if i <= len(judul_list):
669                 # st.header(f"Judul:
{judul_list[i][1]}") #menampilkan judul artikel
670 judul_tersimpan.append(judul_list[i][1])
671                 #st.write(f"Konten:
{judul_list[i][4]}") #menampilkan artikel
672                 st.markdown(
673                     f'<h3><a style="color:
#778899;"
href="{judul_list[i][3]}">{judul_list[i][1]}</a></h3>'
, unsafe_allow_html=True)
674 st.write(f"{judul_list[i][2]}") #menampilkan tanggal
dan waktu
675 st.markdown(judul_list[i][4][:200])
#judul_tagged =
judul_tagged[0] # Kembalikan ke bentuk yang
diharapkan
#judul_tagged = '
'.join([f'{word}/{tag}' for word, tag in
judul_tagged]) # Gabungkan kembali kata-kata dengan
tag
676 # POS Tagging pada judul
artikel
671 judul_tagged =
judul_list[i][1].split()
```

```

        judul_tagged =
data.tag_sents([judul_tagged])
        #judul_tagged = 'test'

683         st.success(f"Hasil Part-
of-Speech (POS) Tagging untuk judul artikel:
{judul_tagged}")

        #displayed_documents.append(judul_list[i][1])
        # POS Tagging pada judul
        artikel

        # tanggal_tagged =
        judul_list[i][2].split()
        # tanggal_tagged =
data.tag_sents([tanggal_tagged])
        #judul_tagged = 'test'
        # print(tanggal_tagged)
        # st.success(f"Hasil Part-
of-Speech (POS) Tagging untuk judul artikel:
{tanggal_tagged}")

        #displayed_documents.append(judul_list[i][2])
        # POS Tagging pada konten
        artikel

693         konten_tagged =
694         judul_list[i][4].split()
        konten_tagged =
data.tag_sents([konten_tagged])

        # st.success(f"Hasil Part-
of-Speech (POS) Tagging untuk konten artikel:
{konten_tagged}")

698         displayed_documents.append(judul_list[i][4])

```

```

667 for i in related_docs_indices[:min(5, len(judul_list))]:
668     if i < len(judul_list):
669         st.header(f"Judul:
{judul_list[i][1]}") #menampilkan judul artikel
        st.subheader(f"Tanggal:
605 {judul_list[i][2]}") #menampilkan tanggal dan waktu
        #st.write(f"Konten:
606 {judul_list[i][4]}") #menampilkan artikel
        st.markdown(
607             f'<h3><a
608 style="color: #FFFFFF;"
href="{judul_list[i][3]}">{judul_list[i][1]}</a></h3>',
        unsafe_allow_html=True)
609 st.markdown(judul_list[i][4][:200])

```

```

610                                     #judul_tagged =
judul_tagged[0] # Kembalikan ke bentuk yang diharapkan
611                                     #judul_tagged = '
'.join([f'{word}/{tag}' for word, tag in judul_tagged])
# Gabungkan kembali kata-kata dengan tag
612                                     # POS Tagging pada judul
artikel
613                                     judul_tagged =
judul_list[i][1].split()
614                                     judul_tagged =
data.tag_sents([judul_tagged])
615                                     #judul_tagged='test'
616                                     print(judul_tagged)
617                                     st.success(f"Hasil Part-
of-Speech (POS) Tagging untuk judul artikel:
{judul_tagged}")

```

Kode Program 4. 13 Inisiasi Judul dan Konten Artikel menggunakan *Part-of-Speech (POS) Tagging*

#### Penjelasan Program:

1. Pada kode baris ke 602 dan baris ke 603 digunakan untuk melakukan iterasi pada indeks dalam dokumen terkait yang dibatasi maksimal 5 artikel atau sebanyak jumlah artikel dalam `judul_list` dimana memastikan indeks `i` tidak melebihi panjang `i < len(judul_list)`.
2. Pada kode baris ke 604 sampai baris ke 606 digunakan untuk menampilkan informasi artikel yaitu judul dan tanggal artikel menggunakan `streamlit`.
3. Pada kode baris ke 607 sampai baris ke 609 digunakan menampilkan link dan konten artikel, dimana ketika judul artikel sebagai link yang dapat diklik, dan menampilkan konten artikel sebanyak 500 karakter.
4. Pada kode baris ke 613 sampai baris ke 614 digunakan untuk *Part-Of-Speech (POS) Tagging* pada judul artikel yaitu dengan memecah menjadi daftar kata-kata, melakukan *POS Tagging* pada daftar kata dengan `CRFTagger`
5. Pada kode baris ke 615 sampai 617 digunakan untuk menampilkan hasil *Part-Of-Speech (POS) Tagging* pada judul artikel.

```

619                                     # POS Tagging pada judul
artikel
620                                     tanggal_tagged =
judul_list[i][2].split()
621                                     tanggal_tagged =
data.tag_sents([tanggal_tagged])

```

```

622                                     #judul_tagged='test'
623                                     print(tanggal_tagged)
624                                     st.success(f"Hasil Part-
of-Speech (POS) Tagging untuk judul artikel:
{tanggal_tagged}")
625
626                                     # POS Tagging pada
konten artikel
627                                     konten_tagged =
judul_list[i][4].split()
628                                     konten_tagged =
data.tag_sents([konten_tagged])
629                                     print(konten_tagged)
630                                     st.success(f"Hasil Part-
of-Speech (POS) Tagging untuk konten artikel:
{konten_tagged}")

```

Kode Program 4. 14 Menampilkan *Part-Of-Speech* (POS) *Tagging* pada Judul, tanggal dan Konten Artikel

#### Penjelasan Program:

1. Pada kode program baris ke 619 sampai baris ke 620 digunakan untuk melakukan *Part-Of-Speech* (POS) *Tagging* pada judul artikel, yaitu memecah *string* tanggal artikel menjadi daftar kata-kata, selanjutnya melakukan POS *Tagging* daftar kata dengan *CRFTagger*, kemudian kode baris ke 623 sampai baris ke 624 menampilkan *Part-Of-Speech* (POS) *Tagging* untuk tanggal artikel.
2. Pada kode program baris ke 626 sampai baris ke 630 digunakan untuk melakukan *Part-Of-Speech* (POS) *Tagging* pada konten artikel, yaitu memecah *string* konten artikel menjadi daftar kata-kata, selanjutnya melakukan POS *Tagging* daftar kata dengan *CRFTagger*, kemudian menampilkan *Part-Of-Speech* (POS) *Tagging* untuk konten artikel.

#### 4.5 Analisa Hasil dan Uji Coba

Pada umumnya, suatu sistem akan melalui proses perancangan dan pembuatan sebuah sistem, kemudian proses selanjutnya yaitu melakukan pengujian sistem tersebut. Tujuan dari adanya pengujian sistem ini mempunyai tujuan yaitu sistem yang dibuat memiliki fungsionalitas dengan baik atau tidak, dan menguji apakah sesuai dengan hasil perhitungan manual yang dilakukan sebelumnya dengan yang diterapkan dalam sistem, serta mengetahui seberapa pengaruh adanya metode *Part-of-Speech* (POS) *Tagging* dalam proses koreksi ejaan pada *query* pencarian

artikel pariwisata yaitu dalam mengetahui kata ambigu dalam pelabelan kelas kata (*tag*) pada sistem dan nilai presisi yang dihasilkan setiap *query* yang diinputkan. Pengujian yang akan dilakukan dalam sistem terdiri dari pengujian untuk mengetahui nilai presisi dari koreksi ejaan terhadap *query* yang berjumlah 100 data menggunakan *Levenshtein Distance* dan *Part-of-Speech* (POS) *Tagging*. Selanjutnya melakukan pengujian sistem untuk mengetahui nilai presisi dari koreksi ejaan terhadap *query* berita pariwisata berbahasa Indonesia menggunakan *Levenshtein Distance* tanpa *Part-of-Speech* (POS) *Tagging*.

#### 4.5.1 Pengujian dan Analisis

Pada bagian ini yaitu melakukan analisa dan uji coba, dimana uji coba dilakukan sebanyak 3 tahap seperti skenario uji coba pada Tabel 3. 12 Data uji yang digunakan sama untuk setiap uji coba yang dilakukan yaitu berjumlah 100 *query* pada Tabel 4. 3 Data uji yang digunakan berjumlah 100 data *query* salah tentang artikel pariwisata. Data uji bervariasi mulai dari 2-5 kata pada kalimat *query* dengan total kata keseluruhan 128 kata, dan 57 kesalahan ejaan, dimana dalam pengujian ini melakukan pengecekan kesalahan ejaan secara manual yaitu dengan mengoreksi menggunakan metode *Levenshtein Distance*, dimana nilai jarak serta kesalahan yang terjadi dalam setiap *query* pada tiga (3) operasi diantaranya, operasi penyisipan, operasi penghapusan, dan operasi substitusi, sehingga hal ini dapat diketahui masing-masing kesalahan dalam setiap kata pada 100 data *query* yang dibuat, diantaranya

- 23 penyisipan
- 21 penghapusan
- 13 substitusi

Tabel 4. 3 Detail Operasi Pada *Query* Salah

NO.	QUERY BENAR	QUERY SALAH	OPERASI		
			D	I	S
1	karnaval budaya kabupaten klaten	karnval budaya kbupaten klaten		2	
2	kunjungan wisatawan	kunjungan wista	1	1	

NO.	QUERY BENAR	QUERY SALAH	OPERASI		
			D	I	S
3	objek wisata	objek wisataa	1		1
4	Jalan menuju kawasan wisata	jalan menuju kawasan wista		1	
5	Destinasi Wisata Sejarah	destinasi wista indonesia terbaru		1	1
6	kawasan Waduk Jatigede	kawsan wadk Jatigede		2	
7	Dinas Kebudayaan dan Pariwisata Banjarnegara	dinas kebudyan dan pariwisata banjarnegara		2	
8	Mendongkrak kunjungan wisatawan	mendongkrak knjungan wisatawan		1	
9	Upacara Tabuik, Tradisi Religius Masyarakat	upacara tabuik, tradisi religius masyarakat		1	
10	Lomba Hias dan Balap Perahu di Majalengka	lomba hias dan blap perahu di majalengka		1	
11	Wisatawan yang datang ke pantai	wisatawanm yang datangg ke pantai	2		
12	Bus pariwisata terbakar	Bus pariwisata terbakarrr	1	1	
13	Gencarnya promosi wisata	Gencarnya promosi wista		1	
14	Pantai Amed menawarkan wisata bahari	Pantai Amed menawrkan wisata bahari		1	
15	Pasir di pantai berwarna hitam	Pasir di pantai berwarna hitamm	1		
16	jumlah turis asing	jumlahh turis asing	1		
17	Seluk-beluk Tercetusnya Menara	Seluk-beluk Tercetusnyaa Menara	1		
18	Genjot Potensi Wisata Jabar	Genjot Potinsi Wisata jabar			1
19	mendorong lalu lintas turis	mendorong lalu lintas turiss	1		
20	kawasan pariwisata unggulan	kawasan parriwisata unggulan	1		
21	wisatawan asing di Indonesia	wisatawanm asing di Indonesia	1		






NO.	QUERY BENAR	QUERY SALAH	OPERASI		
			D	I	S
22	pariwisata di Bali	pariwisata di Bali <sup>ii</sup>	1		
23	sektor pariwisata	ssektor ppariwisata	1		
24	wisatawan asing	wistawan asing <sup>gg</sup>	1	1	1
25	pariwisata berkembang dan menarik	pariwista berkembang <sup>gg</sup> dan menarik	1	1	
26	festival budaya	festival buday <sup>aa</sup>	1		
27	Destinasi wisata Indonesia terbaru	Destinasi wista indonesia terbaru		1	1
28	Promo Liburan Indonesia	Promo linuran Indonesia			1
29	Event Pariwisata Indonesia	Event pariwisata indonesua			1
30	Kuliner halal bali	Kulinr halal bali		1	
31	Wisata Tersembunyi di Indonesia	Wista teresembunyi di Indonesia	1	1	
32	Balap Sepeda sambil Berwisata	Balp Ssepeda sambill Bewisata	2	2	
33	Wisata murah di Indonesia	Wista murah di Indonesia	1		
34	Harga tiket wisata malang	Harga tiket wista malam <sup>g</sup>		1	1
35	Bus Wisata Terbakar Hebat di Jalan Solo	Bus Wista Terbkar Hebat <sup>tt</sup> di Jaln Solo	1	3	
36	Klaten menyelenggarakan karnaval budaya	Klaten menyelnggarakan karnval budaya		2	
37	Kepala Dinas	Kepall <sup>a</sup> Diny <sup>s</sup>	1		1
38	Pentas Tari Meriahkan Karnaval Budaya	Pntas Tari Meriahkn Karnaval Bdaya	1	3	
39	Siang Ini Ada Karnaval	Siamg Inie Adha Karaval	1	1	1
40	Pemuda Olahraga dan Pariwisata Pemkab Klaten	Pemmuda lahraga daen Parwisata Pemkab <sup>bb</sup> Klateen	4	2	
41	Mengikuti pawai pembangunan	Mengikti pawwai pembanginan	1	1	1
42	Pariwisata Indonesia	Parriwsta	1	2	

NO.	QUERY BENAR	QUERY SALAH	OPERASI		
			D	I	S
43	Pameran terpadu sektor perdagangan, pariwisata	Pamerran terpadu sekttor perdagingan, pariwisata	3		1
44	sampah di destinasi wisata	samppah di detinasi wista	1	2	
45	Pembangunan pariwisata berkelanjutan	Pembangnan ppariwisata brkelanjutann	2	2	
46	Menteri Pariwisata dan Ekobaris ke mi Kreatif	Menterii Parriwisata ddan Ekonomi Kreatif	4		1
47	pengembangan wisata ramah muslim	pengembangann wysta ramah muslim	1	11	
48	ikon pariwisata dalam negeri	ikon parriwisata dalam negerii	2		
49	Seorang sopir pariwisata di Bali	Sseorang sopyr pariwisatay di Bali	2		2
50	mengimbau masyarakat atau wisatawan	mengimbau masyarakk atau wisatawan	2	1	
51	Beragam peristiwa terjadi di Jawa Barat	Beragammm peristiwa terjadi di Jawa Baratt	2		
52	Seorang sopir pariwisata	Seorangg soyir pariwisataaa	2	1	
53	harga tiket dan sejenisnya	harga tikett dan sejenisnyaa	2		
54	membangun ekosistem pariwisata	membbbangun ekosistem pariwisata	1		1
55	perjalanan wisatawan nusantara	pperjalanan wiratawan nusantara	1		1
56	parkir objek wisata Pantai	parkir objek wisataa Pantai	1		
57	daya tarik wisata	daya tarik wisataa	1		
58	restoran yang menyediakan makanan	rrestoran yang menyediakan makanan	1		
59	kunjungan wisatawan meningkat.	kunjungann wiisatawan meningkat	1		
60	Kunjungan Turis Naik Signifikan	kunjungannn turis naik signifikan	1		

NO.	QUERY BENAR	QUERY SALAH	OPERASI		
			D	I	S
61	pengunjung restoran	penggunjung restorann	1		
62	mengelola usaha pariwisata	meengelola ushaa pariwisata	2	1	
63	destinasi pariwisata berdaya saing	ddestinasi pariwisata berdaya saingg	2		
64	mengelola bisnis pariwisata	mengelola bbisnis pariwisata	1		
65	desa wisata dan destinasi pariwisata	desa wisata dan destinasi pariwisataa	1		1
66	pariwisata hijau dan energi hijau.	pariwisatah hijau dan energi hijau.	1		
67	pergelaran angklung terbesar	ppergelaran angkluny terbesar	1		1
68	Pemuda Olah dan Pariwisata	Pemuda Olah dan Pariwisata		1	
69	Pantai Teluk Penyu	pantai teluk penyuu	1		
70	pergi ke Pantai Pasir Putih	pergi ke pantai pasir pputih	1		
71	kekayaan modalitas sosial budaya	kekayaan modalitass sosial buddaya	2		
72	dekorasi lukisan batik karya Museum	deekorasi lukisan batik karya mmuseum	2		1
73	taman dan kolam khas Bali.	tamay dan kolammm khas bali		1	1
74	perumahan tradisional	perumaham tradysional		1	1
75	mengunjungi tempat-tempat bersejarah	mmengunjungi ttempat-tempat bersejarah	1		
76	kerjasama di berbagai sektor	kerjasama di beibagai setkor			2
77	Biaya untuk layanan umum	biayya untuk layanan umum	1		
78	rekreasi dan pendidikan	rekrehasi dan pendidikann	1		1
79	Harga mobil listrik di negara	Hargay mobil listrik di negaraa	1		1
80	kerja sama ekobaris ke mi dan sosial budaya	ketja sama ekonbaris ke mi dan sossial budaya	1		1
81	video pariwisata Bali	video ppariwisata bali	2		

NO.	QUERY BENAR	QUERY SALAH	OPERASI		
			D	I	S
82	Penampilan tarian Bali	panampilan tarian bbali	1		1
83	pembangunan Taman Mini Indonesia	pembangunann Taman Mini Idonesia	1	1	1
84	rekor pertunjukan angklung terbesar	rekorrr pertunjukkan angklung terbesar	1	1	
85	hadir di Festivall Perahu Rakit	hadur di Festival Perahuy Rakit	1		1
86	warisan budaya tradisional	warissan bidaya tradisional	1		1
87	peragaan busana	peragaan bussana	2		
88	pekerja bangunan arsitektur Bali.	pekerja bengunan arsitektur Bali.			1
89	Wisatawan dari luar negeri	Wisatawin dari luar negery			2
90	alunan musik kesenian	alunan musyk kesunian			2
91	objek wisata religi	objekk wisata religi	1		
92	dampak positif bagi sektor pariwisata	dampakk positif bagi sekktor pariwisata	2		
93	melestarikan budaya Indonesia	elestarikan budayya Indonesia	1	2	1
94	Pantai Berawa	antai Berawa		1	
95	Destinasi wisata Candi Borobudur	Deestinasi wisata Ccandi Borobudur	2		
96	bus pariwisata di Bali	buss parriwisata di Bali	2		
97	memanfaatkan potensi pariwisata.	memmanfaatkan potensy pariwisataaa	1	1	1
98	kualitas desa wisata	kualits desa wissata	1		1
99	Pengembangan pariwisata	pengembangn ppariwisata	1	1	
100	Menteri Pariwisata dan Ekonomi Kreatif	menterii pariwista dan ekonomi Kreatiff	2	1	

Keterangan : **D** = Jumlah operasi penghapusan (deletion) pada *Query*

- I** = Jumlah operasi penambahan (*insertion*) pada *Query*  
**S** = Jumlah operasi pergantian atau (*substitution*) pada *Query*  
 = Letak operasi penghapusan (*deletion*) pada kata  
 = Letak operasi penambahan (*insertion*) pada kata  
 = Letak operasi pergantian (*substitution*) pada kata

#### 4.5.2 Hasil Skenario Pengujian Koreksi Ejaan dari *Query* Berita Pariwisata Berbahasa Indonesia Menggunakan *Levenshtein Distance* dan *Part-of-Speech (POS) Tagging*

Pada bagian ini yaitu melakukan proses pengujian dari 100 data *query* yang dibuat. Sehingga hasil dari nilai presisi yang diperoleh dalam pengujian *Levenshtein Distance* dan tanpa menggunakan *Part-Of-Speech (POS) Tagging* diperoleh nilai presisi sebagai berikut :

- Hasil Nilai Presisi Uji *Testing Data Query* 100 *Levenshtein Distance* dan Tanpa *Part-Of-Speech (POS) Tagging*

Berikut ini hasil nilai presisi dari 100 *query* pada Tabel 4. 4 berikut ini:

Tabel 4. 4 Hasil Nilai Presisi 100 *Query*

Query Salah	<i>Levenshtein Distance</i>		<i>Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision	Hasil Koreksi	Precision
1	karnaval budaya kabupaten klaten	0,5	karnaval budaya kabupaten klaten	0,125
1	kunjungan wisata	0,5	kunjungan wisata	0,5
2	objek wisata	0,5	objek wisata	0,5
3	jalan menuju kawasan wisata	0,25	jalan menuju kawasan wisata	0,25
4	destinasi wisata indonesia terbaru	0,5	destinasi wisata indonesia terbaru	0,25
5	kawasan waduk jatigede	0,666666667	kawasan waduk jatigede	0,166666667

Query Salah	<i>Levenshtein Distance</i>		<i>Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision	Hasil Koreksi	Precision
6	dinas kebudayaan dan pariwisata banjarnegara	0,2	dinas kebudayaan dan pariwisata banjarnegara	0,2
7	mendongkrak kunjungan wisatawan	0,333333333	mendongkrak kunjungan wisatawan	0,333333333
8	upacara tabuik tradisi religius masyarakat	0,2	upacara tabuik tradisi religius masyarakat	0,2
9	lomba hias dan balap perahu di majalengka	0,142857143	lomba hias dan balap perahu di majalengka	0,142857143
10	wisatawan yang datang ke pantai	0,4	wisatawan yang datang ke pantai	0,1
11	bus pariwisata terbakar	0,333333333	bus pariwisata terbakar	0,333333333
12	gencarnya promosi wisata	0,333333333	gencarnya promosi wisata	0,333333333
13	pantai amed menawarkan wisata bahari	0,2	pantai amed menawarkan wisata bahari	0,2
14	pasir di pantai berwarna hitam	0,2	pasir di pantai berwarna hitam	0,2
15	jumlah turis asing	0,333333333	jumlah turis asing	0,166666667
16	sebelum tercetusnya menara	0,666666667	sebelum tercetusnya menara	0,166666667
17	gendit potensi wisata jabar	0,5	gendit potensi wisata jabar	0,125

Query Salah	<i>Levenshtein Distance</i>		<i>Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision	Hasil Koreksi	Precision
18	mendorong lalu lintas turis	0,25	mendorong lalu lintas turis	0,25
19	kawasan pariwisata unggulan	0,333333333	kawasan pariwisata unggulan	0,333333333
20	wisatawan asing di indonesia	0,25	wisatawan asing di indonesia	0,125
21	pariwisata di balai	0,333333333	pariwisata di balai	0,333333333
22	sektor pariwisata	1	sektor pariwisata	0,25
23	wisatawan asing	0,5	wisatawan asing	0,5
24	pariwisata berkembang dan menarik	0,25	pariwisata berkembang dan menarik	0,25
25	festival budaya	0,5	festival budaya	0,5
26	destinasi wisata indonesia terbaru	0,5	destinasi wisata indonesia terbaru	0,25
27	promo liburan indonesia	0,333333333	promo liburan indonesia	0,333333333
28	event pariwisata indonesia	0,333333333	event pariwisata indonesia	0,333333333
29	kuliner halal bali	0,333333333	kuliner halal bali	0,166666667
30	wisata tersembunyi di indonesia	0,5	wisata tersembunyi di indonesia	0,125
31	bala sepeda sambil berwisata	1	bala sepeda sambil berwisata	0,125
32	wisata murah di indonesia	0,25	wisata murah di indonesia	0,125

Query Salah	<i>Levenshtein Distance</i>		<i>Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision	Hasil Koreksi	Precision
33	harga tiket wisata malang	0,5	harga tiket wisata malang	0,25
34	bus wisata terbakar hebat di jln solo	0,571428571	bus wisata terbakar hebat di jln solo	0,142857143
35	klaten menyelenggarakan karnaval budaya	0,5	klaten menyelenggarakan karnaval budaya	0,25
36	kepala dinas	1	kepala dinas	0,25
37	pentas tari meriah karnaval budaya	0,8	pentas tari meriah karnaval budaya	0,1
38	siang ini agha karnaval	1	siang ini agha karnaval	0,125
39	pemuda olahraga daun pariwisata pemkab klaten	1	pemuda olahraga daun pariwisata pemkab klaten	0,083333333
40	mengikuti pawai pembangunan	1	mengikuti pawai pembangunan	0,166666667
41	pariwista	1	pariwista	0
42	pameran terpadu sektor perdagangan pariwisata	0,8	pameran terpadu sektor perdagangan pariwisata	0,1
43	sampah di destinasi wisata	0,75	sampah di destinasi wisata	0,125
44	pembangunan pariwisata berkelanjutan	1	pembangunan pariwisata berkelanjutan	0,166666667



Query Salah	<i>Levenshtein Distance</i>		<i>Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision	Hasil Koreksi	Precision
45	menteri pariwisata dan ekonomi kreatif	1	menteri pariwisata dan ekonomi kreatif	0,1
46	pengembangan pesta ramah muslim	0,5	pengembangan pesta ramah muslim	0,125
47	ikon pariwisata dalam negeri	0,5	ikon pariwisata dalam negeri	0,25
48	seseorang sopir pariwisata di bali	0,6	seseorang sopir pariwisata di bali	0,1
49	mengimbau masyarakat atau wisatawan	0,25	mengimbau masyarakat atau wisatawan	0,25
50	beragam peristiwa terjadi di jawa barat	0,333333333	beragam peristiwa terjadi di jawa barat	0,083333333
51	seorang sopir pariwisata	1	seorang sopir pariwisata	0,166666667
52	harga tiket dan sejenisnya	0,5	harga tiket dan sejenisnya	0,25
53	membangun ekosistem pariwisata	0,333333333	membangun ekosistem pariwisata	0,166666667
54	perjalanan wisatawan nusantara	0,666666667	perjalanan wisatawan nusantara	0,166666667
55	parkir objek wisata pantai	0,25	parkir objek wisata pantai	0,25
56	daya tarik wisata	0,333333333	daya tarik wisata	0,333333333

Query Salah	<i>Levenshtein Distance</i>		<i>Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision	Hasil Koreksi	Precision
57	restoran yang menyediakan makanan	0,25	restoran yang menyediakan makanan	0,125
58	kunjungan wisatawan meningkat	0,333333333	kunjungan wisatawan meningkat	0,333333333
59	kunjungan turis naik signifikan	0,25	kunjungan turis naik signifikan	0,125
60	pengunjung restoran	1	pengunjung restoran	0,25
61	mengelola ustaz pariwisata	0,666666667	mengelola ustaz pariwisata	0,166666667
62	destinasi pariwisata berdaya saing	0,5	destinasi pariwisata berdaya saing	0,125
63	mengelola bisnis pariwisata	0,333333333	mengelola bisnis pariwisata	0,333333333
64	desa wisata dan destinasi pariwisata	0,4	desa wisata dan destinasi pariwisata	0,1
65	pariwisata hijau dan energi hijau	0,2	pariwisata hijau dan energi hijau	0,2
66	pergelaran angklung terbesar	0,666666667	pergelaran angklung terbesar	0,166666667
67	pemuda olah dan pariwisata	0,25	pemuda olah dan pariwisata	0,25
68	pantai teluk penyu	0,333333333	pantai teluk penyu	0,333333333
69	pergi ke pantai pasir putih	0,2	pergi ke pantai pasir putih	0,2

Query Salah	<i>Levenshtein Distance</i>		<i>Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision	Hasil Koreksi	Precision
70	kekayaan modalitas sosial budidaya	0,5	kekayaan modalitas sosial budidaya	0,25
71	dekorasi lukisan batik karya museum	0,4	dekorasi lukisan batik karya museum	0,1
72	tama dan kolam khas bali	0,4	tama dan kolam khas bali	0,1
73	perumahan tradisional	1	perumahan tradisional	0,25
74	mengunjungi tempatnya bersejarah	0,666666667	mengunjungi tempatnya bersejarah	0,166666667
75	kerjasama di berbagai setor	0,5	kerjasama di berbagai setor	0,25
76	biaya untuk layanan umum	0,25	biaya untuk layanan umum	0,125
77	rekreasi dan pendidikan	0,666666667	rekreasi dan pendidikan	0,166666667
78	harga mobil listrik di negara	0,4	harga mobil listrik di negara	0,1
79	ketua sama ekonomis ke mi dan sosial budaya	0,375	ketua sama ekonomis ke mi dan sosial budaya	0,0625
80	video pariwisata balai	0,666666667	video pariwisata balai	0,333333333
81	penampilan tarian bali	0,666666667	penampilan tarian bali	0,166666667

Query Salah	<i>Levenshtein Distance</i>		<i>Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision	Hasil Koreksi	Precision
82	pembangunan taman mini indonesia	0,5	pembangunan taman mini indonesia	0,125
83	rekor pertunjukkan angklong terbesar	0,5	rekor pertunjukkan angklong terbesar	0,125
84	hadir di festival perahu rakit	0,4	hadir di festival perahu rakit	0,1
85	warisan budaya tradisional	0,666666667	warisan budaya tradisional	0,166666667
86	peragaan busana	0,5	peragaan busana	0,5
87	pekerja bangunan arsitektur bali	0,25	pekerja bangunan arsitektur bali	0,25
88	wisatawan dari luar negeri	0,5	wisatawan dari luar negeri	0,125
89	alunan musik kesenian	0,666666667	alunan musik kesenian	0,333333333
90	objek wisata religi	0,333333333	objek wisata religi	0,166666667
91	dampak positif bagi sektor pariwisata	0,4	dampak positif bagi sektor pariwisata	0,1
92	lestarikan budaya indonesia	0,666666667	lestarikan budaya indonesia	0,166666667
93	santai berawa	0,5	santai berawa	0,25
94	destinasi wisata candi borobudur	0,5	destinasi wisata candi borobudur	0,125
95	bus pariwisata di bali	0,5	bus pariwisata di bali	0,125

Query Salah	<i>Levenshtein Distance</i>		<i>Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision	Hasil Koreksi	Precision
96	memanfaatkan potensi pariwisata	1	memanfaatkan potensi pariwisata	0,166666667
97	kualitas desa wisata	0,666666667	kualitas desa wisata	0,166666667
98	pengembangan pariwisata	1	pengembangan pariwisata	0,25
99	menteri pariwisata dan ekonomi kreatif	0,4	menteri pariwisata dan ekonomi kreatif	0,1

- Hasil Nilai Presisi Data *Query* Pencarian *Levenshtein Distance* dan Tanpa *Part-Of-Speech (POS) Tagging*.

Pada bagian ini merupakan hasil pencarian dari artikel berita pariwisata, dimana ketika *query* diinputkan, kemudian hasil *query* benar ini nantinya akan menampilkan data artikel pariwisata yaitu pada bagian judul yang sesuai dengan *query* yang diinputkan. Berikut ini tabel 4.5 dibawah ini.

Tabel 4. 5 Hasil 100 *Query* Nilai Presisi *Search Engine*

No.	<i>Search Engine Levenshtein Distance</i>		<i>Search Engine Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision <i>Search Engine</i>	Hasil Koreksi	Precision <i>Search Engine</i>
1	karnaval budaya	0,6	karnaval budaya	0,6
2	puncak ijen	1,0	Puncak ijen	1,0
3	tarian reog klaten	Tidak ada berita	tarian reog klaten	Tidak ada berita
4	perahu	0,6	perahu	0,6
5	cianjur	0,3	Cianjur	0,3

No.	<i>Search Engine Levenshtein Distance</i>		<i>Search Engine Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision <i>Search Engine</i>	Hasil Koreksi	Precision <i>Search Engine</i>
6	kawasan waduk jatigede	Tidak ada berita	kawasan waduk jatigede	Tidak ada berita
7	tariann tradisional klaten	Tidak ada berita	tarian tradisional klaten	Tidak ada berita
8	kunjungan wisatawan	0,0	kunjungan wisatawan	0,0
9	upacara tabuik tradisi religius masyarakat	Tidak ada berita	upacara tabuik tradisi religius masyarakat	Tidak ada berita
10	lomba hias dan balap perahu di majalengka	Tidak ada berita	lomba hias dan balap perahu di majalengka	Tidak ada berita
11	pantai	1,0	pantai	1,0
12	kesenian	0,2	kesenian	0,2
13	Pantai amed	1,0	Pantai amed	1,0
14	pawai	0,4	pawai	0,4
15	pasir di pantai berwarna hitam	Tidak ada berita	pasir di pantai berwarna hitam	0,2
16	Pawwaii HUT RI Padangg	Tidak ada berita	Pawwaii HUT RI Padangg	Tidak ada berita
17	Monumennt Tentara Pelajarr Klaten	Tidak ada berita	Monument Tentara Pelajar Klaten	Tidak ada berita
18	potensi wisata jabar	Tidak ada berita	gendit potensi wisata jabar	Tidak ada berita
19	mendorong lalu lintas turis	0,0	mendorong lalu lintas turis	0,0

No.	<i>Search Engine Levenshtein Distance</i>		<i>Search Engine Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision <i>Search Engine</i>	Hasil Koreksi	Precision <i>Search Engine</i>
20	kawasan pariwisata unggulan	0,0	kawasan pariwisata unggulan	0,0
21	wisatawan asing di indonesia	0,0	wisatawan asing di indonesia	0,0
22	waduk karian lebak	Tidak ada berita	waduk karian lebak	Tidak ada berita
23	Karnaval	0,8	karnaval	0,8
24	asean indo pacificc	1,0	Asean indo pacificc	1,0
25	Pariwisata Sungai	Tidak ada berita	Pariwisata sungai	Tidak ada berita
26	festival budaya	0,5	festival budaya	0,5
27	destinasi wisata indonesia terbaru	Tidak ada berita	destinasi wisata indonesia terbaru	0,25
28	Presiden Jokowi Gelar 3 Budaya	Tidak ada berita	promo liburan indonesia	Tidak ada berita
29	Pemprov Bali Objek Wisata	Tidak ada berita	Pemprov Bali Objek Wisata	Tidak ada berita
30	kuliner halal bali	Tidak ada berita	kuliner halal bali	Tidak ada berita
31	wisata tersembunyi di indonesia	Tidak ada berita	wisata tersembunyi di indonesia	Tidak ada berita
32	Pasar Pengembangan UMKM	Tidak ada berita	Pasar pengembangan umkm	Tidak ada berita

No.	<i>Search Engine Levenshtein Distance</i>		<i>Search Engine Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision <i>Search Engine</i>	Hasil Koreksi	Precision <i>Search Engine</i>
33	wisata murah di indonesia	Tidak ada berita	wisata murah di indonesia	Tidak ada berita
34	harga tiket wisata malang	Tidak ada berita	harga tiket wisata malang	Tidak ada berita
35	bus wisata	0,75	bus wisata terbakar	0,75
36	klaten menyelenggarakan karnaval budaya	0,0	klaten menyelenggarakan karnaval budaya	0,0
37	kepala dinas	0,0	kepala dinas	0,0
38	pentas tari meriah karnaval budaya	Tidak ada berita	pentas tari meriah karnaval budaya	Tidak ada berita
39	siang ini ada karnaval	1,0	siang ini ada karnaval	1,0
40	pemuda olahraga daun pariwisata pemkab klaten	Tidak ada berita	pemuda olahraga daun pariwisata pemkab klaten	Tidak ada berita
41	pawai pembangunan	1,0	pawai pembangunan	1,0
42	sektor perdagangan pariwisata	Tidak ada berita	sektor perdagangan pariwisata	Tidak ada berita
43	Produk Pameran Meksiko	Tidak ada berita	Produk Pameran Meksiko	Tidak ada berita
44	Perjanjian Dagang	1,0	Perjanjian dagang	1,0
45	pembangunan pariwisata	0,0	pembangunan pariwisata	0,0
46	turis	0,6	turis	0,6



No.	<i>Search Engine Levenshtein Distance</i>		<i>Search Engine Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision <i>Search Engine</i>	Hasil Koreksi	Precision <i>Search Engine</i>
47	Wisata Sejarah Palas	Tidak ada berita	Wisata Sejarah Palas	Tidak ada berita
48	makanan khas lombok	1,0	makanan khas lombok	1,0
49	pariwisata di bali	0,4	pariwisata di bali	0,4
50	wisatawan	0,0	wisatawan	0,0
51	beragam peristiwa terjadi di jawa barat	0,0	beragam peristiwa terjadi di jawa barat	0,0
52	seorang sopir pariwisata	0,0	seorang sopir pariwisata	0,0
53	harga tiket dan sejenisnya	Tidak ada berita	harga tiket dan sejenisnya	Tidak ada berita
54	Pariwisata Jabar	1,0	membangun ekosistem pariwisata	1,0
55	Pantai Usaha Terkikis	Tidak ada berita	perjalanan wisatawan nusantara	Tidak ada berita
56	parkir objek wisata pantai	0,0	parkir objek wisata pantai	0,0
57	wisata	1,0	wisata	1,0
58	restoran yang menyediakan makanan	0,0	restoran yang menyediakan makanan	0,0
59	kunjungan wisatawan meningkat	Tidak ada berita	kunjungan wisatawan meningkat	Tidak ada berita

No.	<i>Search Engine Levenshtein Distance</i>		<i>Search Engine Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision <i>Search Engine</i>	Hasil Koreksi	Precision <i>Search Engine</i>
60	kunjungan turis naik signifikan	Tidak ada berita	kunjungan turis naik signifikan	Tidak ada berita
61	pengunjung restoran	0,0	pengunjung restoran	0,0
62	Mengelola pariwisata	Tidak ada berita	mengelola pariwisata	Tidak ada berita
63	destinasi pariwisata berdaya saing	0,0	destinasi pariwisata berdaya saing	0,0
64	mengelola bisnis pariwisata	0,0	mengelola bisnis pariwisata	0,0
65	bisnis wisata	Tidak ada berita	bisnis wisata	Tidak ada berita
66	pariwisata hijau dan energi hijau	Tidak ada berita	pariwisata hijau dan energi hijau	Tidak ada berita
67	pergelaran angklung terbesar	1,0	pergelaran angklung terbesar	1,0
68	pemuda olah dan pariwisata	0,0	pemuda olah dan pariwisata	0,0
69	pantai teluk penyu	1,0	pantai teluk penyu	1,0
70	pergi ke pantai pasir putih	0,0	pergi ke pantai pasir putih	0,0
71	budidaya	0,0	budidaya	0,0
72	museum	1,0	museum	1,0
73	festival	0,8	festival	0,8
74	perumahan tradisional	Tidak ada berita	perumahan tradisional	Tidak ada berita

No.	<i>Search Engine Levenshtein Distance</i>		<i>Search Engine Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision <i>Search Engine</i>	Hasil Koreksi	Precision <i>Search Engine</i>
75	karnaval	0,8	karnaval	0,8
76	kerjasama di berbagai setor	Tidak ada berita	kerjasama di berbagai setor	Tidak ada berita
77	biaya untuk layanan umum	Tidak ada berita	biaya untuk layanan umum	Tidak ada berita
78	rekreasi dan pendidikan	Tidak ada berita	rekreasi dan pendidikan	Tidak ada berita
79	harga mobil listrik di negara	0,0	harga mobil listrik di negara	0,0
80	budaya	0,8	ketua sama ekonomis ke mi dan sosial budaya	0,8
81	pariwisata	0,4	video pariwisata balai	0,333333333
82	bali	1,0	bali	1,0
83	pembangunan taman mini indonesia	0,0	pembangunan taman mini indonesia	0,0
84	rekor pertunjukkan angklung terbesar	Tidak ada berita	rekor pertunjukkan angklung terbesar	Tidak ada berita
85	hadir di festival perahu rakit	Tidak ada berita	hadir di festival perahu rakit	Tidak ada berita
86	warisan budaya tradisional	0,0	warisan budaya tradisional	0,0
87	peragaan busana	0,0	peragaan busana	0,0

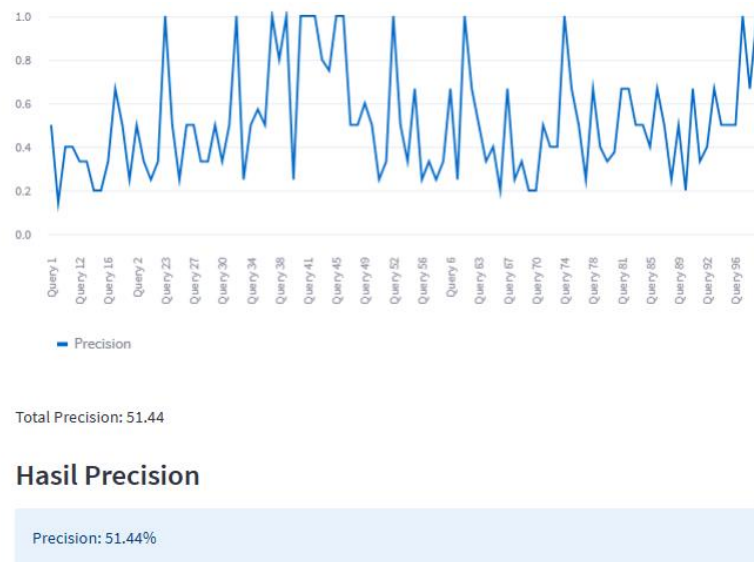
No.	<i>Search Engine Levenshtein Distance</i>		<i>Search Engine Levenshtein Distance dan Part-of-Speech (POS) Tagging</i>	
	Hasil Koreksi	Precision <i>Search Engine</i>	Hasil Koreksi	Precision <i>Search Engine</i>
88	pekerja bangunan arsitektur bali	Tidak Ada berita	pekerja bangunan arsitektur bali	Tidak ada berita
89	wisatawan dari luar negeri	0,0	wisatawan dari luar negeri	0,0
90	alunan musik kesenian	0,0	alunan musik kesenian	0,0
91	objek wisata religi	0,0	objek wisata religi	0,0
92	dampak positif bagi sektor pariwisata	Tidak ada berita	dampak positif bagi sektor pariwisata	Tidak ada berita
93	lestarikan budaya indonesia	0,0	lestarikan budaya indonesia	0,0
94	pantai berawa	0,6	pantai berawa	0,6
95	destinasi wisata candi borobudur	Tidak ada berita	destinasi wisata candi borobudur	Tidak ada berita
96	bus pariwisata di bali	0,0	bus pariwisata di bali	0,0
97	memanfaatkan potensi pariwisata	Tidak ada berita	memanfaatkan potensi pariwisata	Tidak ada Berita
98	kualitas desa wisata	0,0	kualitas desa wisata	0,0
99	pengembangan pariwisata	0,0	pengembangan pariwisata	0,0
100	menteri pariwisata dan ekonomi kreatif	0,0	menteri pariwisata dan ekonomi kreatif	0,0

- Grafik dari hasil nilai presisi uji *testing* data *query* 100 menggunakan algoritma *Levenshtein Distance* dan Tanpa *Part-Of-Speech* (POS) *Tagging*.

Berikut ini merupakan grafik dari hasil nilai presisi uji testing data *query* 100 *Levenshtein Distance* dan Tanpa *Part-Of-Speech* (POS) *Tagging* pada gambar dibawah ini:

- ❖ Grafik Nilai Presisi dari hasil 100 data *query Levenshtein Distance* tanpa *Part-Of-Speech* (POS)

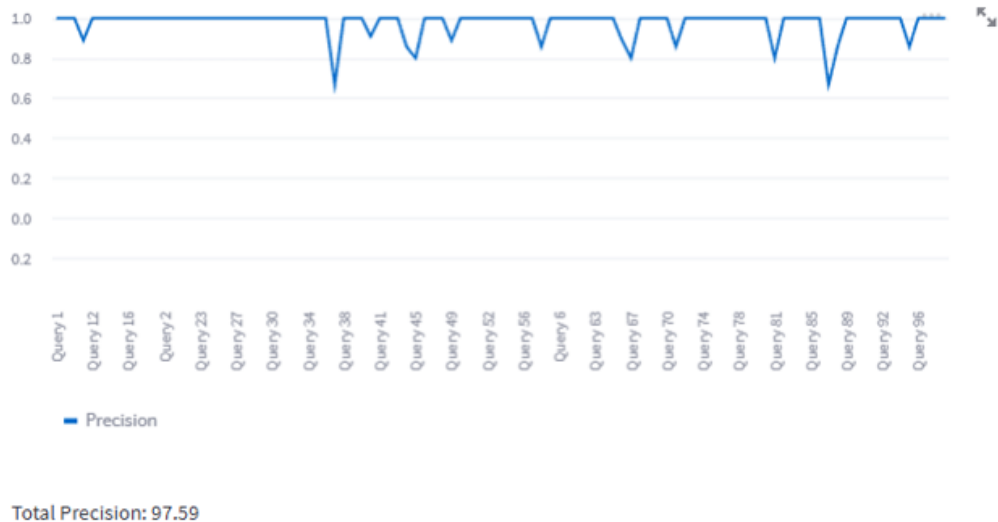
Pada bagian ini menampilkan gambar grafik dengan total nilai presisi yang dihasilkan untuk menghitung berapa jumlah kata yang berhasil dikoreksi, dengan total jumlah kesalahan kata, algoritma *Levenshtein Distance* dan *Part-of-Speech* (POS) *Tagging* dari 100 data *query*, menghasilkan grafik dengan hasil presisi. Dimana total nilai presisi diperoleh dari kata yang telah dikoreksi, selanjutnya presisi dihitung dengan membandingkannya dengan kamus atau daftar kata yang benar. Sehingga hasil presisi ini dihitung sebagai jumlah kata yang dikoreksi dengan benar dibagi dengan jumlah total kata yang telah dikoreksi. Maka hasil total nilai presisi ini, adalah jumlah kata yang berhasil diperbaiki dibagi jumlah dari semua nilai presisi yaitu seperti pada gambar 4.1 dibawah ini.



Gambar 4. 1 Grafik Hasil Presisi 100 data *Query* tanpa *Part-Of\_Speech* (POS) *Tagging*

- ❖ Gambar Grafik Nilai Presisi dengan *Part-Of-Speech* (POS) *Tagging*

Pada bagian ini menampilkan gambar grafik dengan total nilai presisi yang dihasilkan untuk menghitung berapa jumlah kata yang berhasil dikoreksi, dengan total jumlah kesalahan kata, dari 100 data *query*, menggunakan algoritma *Levenshtein Distance* dan *Part-of-Speech (POS) Tagging* menghasilkan grafik dengan hasil presisi yaitu 97.59% seperti pada gambar 4.2 dibawah ini.



Gambar 4. 2 Grafik Hasil Presisi 100 data *Query* dan *Part-Of-Speech (POS) Tagging*

#### 4.5.3 Analisa Hasil Skenario Uji Coba Sistem

Pada pengujian dari skenario uji coba sistem yang dilakukan dengan beberapa data *query* diperoleh hasil skenario uji coba sistem dengan nilai presisi yang terdapat pada tabel 4. 6 di bawah ini:

Tabel 4. 6 Hasil Skenario Uji Coba

NO	<i>Levenshtein Distance</i>			<i>Part-Of-Speech (POS) Tagging</i>			Keterangan
	<i>Query</i>	Nilai Presisi	Waktu	<i>Query</i>	Nilai presisi	Waktu	
1	100	97.59%	3 Detik	100	20.41%	5 Detik	Data dengan jumlah banyak dan banyak kesalahan ejaannya akan menghabiskan waktu yang cukup lama ketika di

							koreksi dan di uji.
2	10	51.41%	1 Detik	10	21.01%	2 Detik	Data dengan jumlah sedikit dan sedikit kesalahan ejaannya akan mempermudah dalam koreksi dan tidak menghabiskan waktu.
3	50	27.41%	2 Detik	50	22.55%	2 Detik	Data dengan jumlah cukup akan tetapi terdapat banyak kesalahan kata yang dibenarkan dan tidak ada dikamus akan menyebabkan nilai presisinya kecil.

#### 4.6 Implementasi Sistem

Pada bagian ini menjelaskan mengenai pembuatan *user interface* aplikasi berbasis *website* yang sederhana sehingga mempermudah *user* dalam menggunakan aplikasi dalam koreksi kesalahan ejaan diimplementasikan terhadap sistem sesuai dengan perancangan dalam sistem koreksi ejaan terhadap *query* pencarian artikel pariwisata. Aplikasi ini membantu *user* dalam mengoreksi kesalahan ejaan kata pada *query* (yang terdiri dari beberapa kata kunci) dalam mesin pencarian untuk menemukan artikel berita pariwisata. Berikut ini adalah tampilan dan hasil aplikasi atau sistem yang dibuat, antara lain sebagai berikut :

- Tampilan Sistem
- *Inputan query*
- *Output query* pencarian artikel pariwisata
- 100 data *query*

- *Query* artikel pariwisata

Berikut adalah hasil dari implementasi sistem koreksi ejaan terhadap *query* pencarian Artikel Pariwisata:

- Tampilan Sistem Koreksi Kesalahan Ejaan Terhadap *Query* Pencarian Artikel Pariwisata



Gambar Tampilan Sistem Koreksi Ejaan

- *Inputan query* salah data artikel pariwisata dan hasil koreksi *ejaan* menggunakan *Levenshtein Distance* dan *Part-Of-Speech (POS) Tagging* dapat dilihat pada Gambar 4. 3 dibawah ini.

## Skenario Levenshtein Distance & POS Tagging

Masukkan Query

karnaval budaya kbupaten klaten

Koreksi Query

Gambar 4. 3 Tampilan *inputan query*

- *Inputan query* salah data artikel pariwisata menggunakan *Levenshtein Distance* tanpa *Part-Of-Speech (POS) Tagging* dan hasil koreksi *ejaan* dapat dilihat pada Gambar 4. 4 dibawah ini.



## Skenario Levenshtein Distance Tanpa POS Tagging

Masukkan Query

karnaval budaya kbupaten klaten

Koreksi Query

Gambar 4. 4 inputan query menggunakan metode *Levenshtein Distance* tanpa *Part-Of-Speech (POS) Tagging*

Dari hasil Koreksi Ejaan terhadap *query* Pencarian Menggunakan Metode *Levenshtein Distance* dan *Part-Of-Speech (POS) Tagging* yang dilakukan diperoleh implementasi sistem seperti Gambar 4. 4 di bawah ini.

- Tampilan hasil koreksi ejaan menggunakan Metode *Levenshtein Distance* dan *Part-Of-Speech (POS) Tagging* seperti Gambar 4. 5 di bawah ini.

## Sistem Koreksi Ejaan Levenshtein Distance dan Part-of-Speech (POS) Tagging

### Skenario Levenshtein Distance & POS Tagging

Masukkan Query

karnaval budaya kbupaten klaten

Koreksi Query

hasil correction: karnaval budaya kabupaten klaten

hasil Part-of-Speech (POS) Tagging:[["karnaval", "NN"], ("budaya", "NN"), ("kabupaten", "NN"), ("klaten", "JJ")]]

Gambar 4. 5 Tampilan skenario *Levenshtein Distance* dan *Part-Of-Speech (POS) Tagging*

- Tampilan hasil Pencarian Artikel Pariwisata menggunakan *Levenshtein Distance* dan *Part-Of-Speech (POS) Tagging* yang menampilkan artikel, diperoleh implementasi sistem seperti Gambar 4. 5 di bawah ini.

## Judul: Karnaval Budaya Klaten 2023 Usung Berbagai Potensi Seni Budaya

Tanggal: 19 Agu 2023 14:44

### Karnaval Budaya Klaten 2023 Usung Berbagai Potensi Seni Budaya

Karnaval budaya Kabupaten Klaten tahun 2023 digelar siang ini di jalan utama kota. Berbagai jenis kesenian dan potensi kebudayaan disajikan peserta. Pantauan detikJateng, sekitar pukul 13.00 WIB para p

Hasil Part-of-Speech (POS) Tagging untuk judul artikel: [[('Karnaval', 'NNP'), ('Budaya', 'NNP'), ('Klaten', 'NNP'), ('2023', 'CD'), ('Usung', 'NNP'), ('Berbagai', 'NNP'), ('Potensi', 'NNP'), ('Seni', 'NNP'), ('Budaya', 'NNP')]]

Hasil Part-of-Speech (POS) Tagging untuk judul artikel: [[('19', 'CD'), ('Agu', 'NNP'), ('2023', 'CD'), ('14:44', 'CD')]]

Gambar 4. 6 Menampilkan hasil pencarian Artikel

- Tampilan hasil koreksi ejaan *query* menggunakan *Levenshtein Distance* Tanpa *Part-Of-Speech* (POS) *Tagging* yang menampilkan koreksi, diperoleh implementasi sistem seperti Gambar 4. 7 di bawah ini.

## Sistem Koreksi Ejaan Levenshtein Distance dan Part-of-Speech (POS) Tagging

### Skenario Levenshtein Distance Tanpa POS Tagging

Masukkan Query

karnaval budaya kabupaten klaten

Koreksi Query

hasil correction: karnaval budaya kabupaten klaten

Rata-rata Precision: 0.0

Nilai Precision: 0.75

Gambar 4. 7 Hasil koreksi ejaan

- Tampilan hasil pencarian artikel menggunakan *Levenshtein Distance* Tanpa *Part-Of-Speech* (POS) *Tagging* yang menampilkan koreksi, diperoleh implementasi sistem seperti Gambar 4. 8 di bawah ini.

## Judul: Karnaval Budaya Klaten 2023 Usung Berbagai Potensi Seni Budaya

Tanggal: 19 Agu 2023 14:44

link: <https://www.detik.com/jateng/berita/d-6884810/karnaval-budaya-klaten-2023-usung-berbagai-potensi-seni-budaya>

### Karnaval Budaya Klaten 2023 Usung Berbagai Potensi Seni Budaya

Karnaval budaya Kabupaten Klaten tahun 2023 digelar siang ini di jalan utama kota. Berbagai jenis kesenian dan potensi kebudayaan disajikan peserta. Pantauan detikJateng, sekitar pukul 13.00 WIB para p

Nilai Precision: 0.75

Gambar 4. 8 Menampilkan Artikel tanpa *Part-Of-Speech (POS) Tagging*

#### 4.7 Evaluasi Sistem

Pada bagian evaluasi sistem ini dilakukan sebagai langkah selanjutnya setelah sistem yang sebelumnya berhasil dibuat. Tujuan adanya evaluasi sistem untuk menjelaskan kembali hasil pengujian dan menjelaskan kendala atau permasalahan yang terjadi dalam penelitian yang telah dilakukan. Berdasarkan hasil pengujian menunjukkan bahwa *Levenshtein Distance* dapat melakukan perbaikan koreksi ejaan kata dengan baik pada beberapa *query* yang kurang tepat dalam kesalahan penulisan kata dapat dilihat pada Gambar 4. 1 di halaman sebelumnya.

Berdasarkan hasil pengujian pada setiap kesalahan kata pada *query* tersebut diperoleh nilai presisi pada skenario *Levenshtein Distance* dan *Part-Of-Speech (POS) Tagging* menghasilkan nilai presisi 97.59%. Adapun kendala yang dihadapi yaitu ketika menghasilkan nilai presisi kecil karena dataset yang digunakan dalam *search engine* atau mesin pencarian datasetnya kurang, karena seharusnya data yang digunakan dalam *search engine* berjumlah miliaran. Sehingga ketika mesin pencarian ingin melakukan pencarian data atau dokumen, data yang diperoleh dan ditampilkan sesuai *query* yang diinputkan menghasilkan hasil pencarian yang sedikit. Kemudian untuk memperoleh data relevan (sesuai) yang dimana sebanyak 5 data relevan (sesuai) dari hasil *query* pencariannya mengalami kesulitan, karena data yang digunakan berjumlah sedikit.

Selanjutnya pada pengujian koreksi kesalahan ejaan terhadap *query* pencarian artikel pariwisata menggunakan metode *Levenshtein Distance* tanpa *Part-of-Speech (POS) Tagging* menghasilkan hasil koreksi dengan nilai presisi yang signifikan yaitu 51.41%, karena nilai presisinya kecil disebabkan terdapat kata yang harusnya dikoreksi akan tetapi sistem tidak mampu dikoreksi, karena tagnya berbeda.

Berdasarkan hasil pengujian yang dilakukan sebelumnya, dapat disimpulkan bahwa terdapat beberapa faktor yang mempengaruhi yaitu

1. Data yang digunakan sedikit
2. Setiap hasil koreksi ejaan kata, terdapat bagian yang tidak sesuai dengan hasil koreksinya, yang dimana kata tersebut seharusnya dikoreksi akan tetapi tidak dikoreksi.
3. Pada setiap satu kali pencarian dari *query* yang diinputkan tidak selalu menampilkan 5 (lima) pencarian dokumen, dan hasil pencariannya dokumen yang relevan sesuai dengan *query* yang diinputkan hasilnya sedikit.

## **BAB V**

### **KESIMPULAN**

#### **5.1 Kesimpulan**

Berdasarkan rangkaian uraian penelitian sistem koreksi ejaan terhadap *query* pencarian artikel pariwisata berbahasa Indonesia dengan menggunakan *Levenshtein Distance* dan *Part-of-Speech (POS) Tagging*, menggunakan 100 data uji *query* diperoleh nilai presisi 97.59% dan tanpa *Part-of-Speech (POS) Tagging* nilai presisi 51.41%. Hasil analisa dan pembahasan dalam proses koreksi ejaan menggunakan metode *Levenshtein Distance* mampu melakukan koreksi ejaan dengan baik, dan sistem dengan cepat mengoreksi dan metode *Part-of-Speech (POS) Tagging* melakukan pelabelan *tag* untuk mengetahui adanya ambiguitas kata atau kata yang bermakna banyak.

#### **5.2 Saran**

Berdasarkan hasil kinerja sistem yang dilakukan pada koreksi kesalahan ejaan terhadap *query* pencarian artikel pariwisata, terdapat permasalahan yaitu sebagai berikut:

1. Nilai presisi yang diperoleh kecil karena data yang digunakan sedikit.
2. Data kamus yang digunakan memuat beberapa kata yang tidak baku, sehingga data *query* yang dilakukan dengan pengoreksian secara manual tidak sesuai dengan hasil koreksi pada sistem. Sehingga daftar hasil *query* benar mengikuti hasil koreksi pada kamus.
3. Adanya *Part-Of-Speech (POS) Tagging* tidak direkomendasikan dalam penerapan koreksi ejaan karena selain mempengaruhi nilai presisinya, *corpus* yang terdapat dalam *Part-Of-Speech (POS) Tagging* yang tersedia, datanya tidak terdapat kelas kata secara signifikan.

## REFERENSI

- [1] A. Subhan Yazid, A. Fatwanto, T. Informatika Fakultas Sains dan Tekbaris ke logi, and U. Sunan Kalijaga Yogyakarta Jl Laksda Adisucipto, “Penentuan Kelas Kata Pada Part Of Speech Tagging Kata Ambigu Bahasa Indonesia,” *Jurnal Informatika Sunan Kalijaga*), vol. 2, baris ke 3, pp. 157–166, 2018, Accessed: Jan. 19, 2024. [Online]. Available: DOI: <https://doi.org/10.14421/jiska.2018.23-05>
- [2] R. Martin, D. Santun Naga, and V. Christanti Mawardi, “Penggunaan Spelling Correction Dengan Metode Peter Baris ke rvig dan N-Gram.” Accessed: Jan. 19, 2024. [Online]. Available: DOI: <https://doi.org/10.24912/jiksi.v9i1.11591>
- [3] A. Y. R. Fitriani and L. E. Rahmawati, “Analisis kesalahan penggunaan tanda baca dan huruf miring dalam teks berita online detiknews dan tribunnews,” *BAHA STRA*, vol. 40, baris ke 1, p. 10, Apr. 2020, doi: 10.26555/bahastra.v40i1.14695.
- [4] U. Pembangunan *et al.*, “Penggunaan Bahasa pada Perkembangan Industri Pariwisata di Surabaya: Studi Kasus Objek Wisata Museum Sepuluh Baris ke pember Ilmatus Sa’diyah,” 2023. Accessed: Jan. 19, 2024. [Online]. Available: DOI: <https://doi.org/10.12928/mms.v4i2.8072>
- [5] R. Ariyansyah, R. Nanda, and O. Wiranda, “Search Engine Menggunakan Metode Information Retrival,” 2022. Accessed: Jan. 19, 2024. [Online]. Available: DOI: <https://doi.org/10.58794/santi.v2i1.68>
- [6] Husni, I. O. Suzanti, Y. D. Pramudita, S. S. Putro, and L. Heryawan, “Web Service for Search Engine Bahasa Indonesia (SEBI),” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Jul. 2020. doi: 10.1088/1742-6596/1569/2/022087.
- [7] S. Kusumadewi, L. Rosita, and C. I. Ratnasari, “A Baris ke n-Word Error Spell Checker for Patient Complaints in Bahasa Indonesia,” 2017. [Online]. Available: <https://www.researchgate.net/publication/318927640>
- [8] M. O. Braddley, M. Fachrurrozi, and N. Yusliani, “Pengoreksian Ejaan Kata Berbahasa Indonesia Menggunakan Algoritma Levenshtein Distance,” 2017.
- [9] M. S. Simanjuntak, H. Sujaini, and N. Safriadi, “Spelling Corrector Bahasa Indonesia dengan Kombinasi Metode Peter Baris ke rvig dan N-Gram,” *Jurnal Edukasi dan Penelitian Informatika (JEPIN)*, vol. 4, baris ke 1, p. 17, Jun. 2018, doi: 10.26418/jp.v4i1.24075.
- [10] D. Markuci *et al.*, “Implementasi Spelling Corrector untuk mengatasi Typographical Error pada Fitur Pencarian Aplikasi Kamus Istilah Informatika,” vol. 17, baris ke 1, 2023, doi: 10.47111/JTI.
- [11] A. I. Fahma, I. Cholissodin, and R. S. Perdana, “Identifikasi Kesalahan Penulisan Kata (Typographical Error) pada Dokumen Berbahasa Indonesia Menggunakan Metode N-gram dan Levenshtein Distance,” 2018. [Online]. Available: <http://j-ptiik.ub.ac.id>

- [12] Y. Purnama Sari, G. Aditra Pradnyana, and I. Made Agus Wirawa, "Pengembangan Aplikasi Bahasa BIMA-Bahasa Indonesia Menggunakan Algoritma Levenshtein Distance Sebagai Spell Checker Berbasis Android," 2001. Accessed: Jan. 19, 2024. [Online]. Available: DOI: <https://doi.org/10.23887/karmapati.v8i2.17964>
- [13] Nurul Huda and Muhammad Khafidurrohman, "Perbandingan Efisiensi Algoritma String Matching Knuth Morris Pratt Dan Algoritma Levenshtein Pada Aplikasi Pengarsipan Dan Pencarian Data Anggota Honda Megapro Club Indonesia," *JUPITER*, vol. 15 baris ke 1, baris ke Program Studi Teknik Informatika, Universitas Bina Darma Palembang. Jl. Jenderal A. Yani Baris ke 3, 9/10 Ulu, Seberang Ulu I, Palembang, Sumatera Selatan 30111 e-mail: \*nurul\_huda@binadarma.ac.id, \*mhd.khafi26@gmail.com, pp. 787–798, Apr. 4AD, Accessed: Jan. 19, 2024. [Online]. Available: DOI: <https://doi.org/10.5281./6639/15.jupiter.2023.04>
- [14] K. Widhiyanti and A. Harjoko, "POS Tagging Bahasa Indonesia Dengan HMM dan Rule Based."
- [15] B. Pham and B. P. Student, "Parts of Speech Tagging: Rule-Based Parts of Speech Tagging: Rule-Based Parts of Speech Tagging: Rule-Based," 2020. [Online]. Available: [https://digitalcommons.harrisburgu.edu/cisc\\_student-coursework](https://digitalcommons.harrisburgu.edu/cisc_student-coursework)
- [16] K. K. Purnamasari and I. S. Suwardi, "Rule-based Part of Speech Tagger for Indonesian Language," in *IOP Conference Series: Materials Science and Engineering*, Institute of Physics Publishing, Sep. 2018. doi: 10.1088/1757-899X/407/1/012151.
- [17] T. Aprilianto and A. Badawi, "Sistem Koreksi Kata dan Pengenalan Struktur Kalimat Berbahasa Indonesia dengan Pendekatan Kamus Berbasis Levenshtein Distance," 2017.
- [18] S. K. Dirjen *et al.*, "Terakreditasi SINTA Peringkat 2 Autocorrect pada Modul Pencarian Drugs e-Dictionary Menggunakan Algoritma Levenshtein Distance," *masa berlaku mulai*, vol. 1, baris ke 3, pp. 64–69, 2017.
- [19] D. Hládek, J. Staš, and M. Pleva, "Survey of automatic spelling correction," *Electronics (Switzerland)*, vol. 9, baris ke 10. MDPI AG, pp. 1–29, Oct. 01, 2020. doi: 10.3390/electronics9101670.
- [20] Y. Apriani *et al.*, "Basastra: Jurnal Kajian Bahasa dan Sastra Indonesia KESALAHAN EJAAN BAHASA INDONESIA PADA BERITA KORAN".
- [21] nuraini.ahmad and anis.masruri, "Penerapan Information Retrieval Pada Search Engine," *Jurnal Ibaris ke vasi Hasil Penelitian dan Pengembangan*, vol. 1, baris ke p-ISSN : 2809-4042 | e-ISSN : 2809-4034, pp. 15–23, Dec. 2021.
- [22] H. Artanto, F. Nurdiyansyah, and U. Widyagama Malang, "Penerapan SEO (Search Engine Optimization) Untuk Meningkatkan Penjualan Produk," *Journal of Information Techbaris ke logy and Computer Science*

- (JOINTECS), vol. 1, baris ke 2, 2017, [Online]. Available: <http://info.cern.ch/>
- [23] R. Ariyansyah, R. Nanda, and O. Wiranda, "Search Engine Menggunakan Metode Information Retrieval," 2022. Accessed: Jan. 19, 2024. [Online]. Available: DOI: <https://doi.org/10.58794/santi.v2i1.68>
  - [24] R. Ridlo Baihaqi, "Temu Kembali Informasi pada Berita Olahraga Berbahasa Indonesia dengan Seleksi Fitur Term Frequency dan Metode BM25," 2020. [Online]. Available: <http://j-ptiik.ub.ac.id>
  - [25] E. Esyudha Pratama and J. H. Hadari Nawawi, "Information Retrieval pada Proses Penyimpanan dan Pencarian Dokumen Digital Menggunakan Metode Text Mining," 2018.
  - [26] E. L. Amalia<sup>1</sup>, A. J. Jumadi, I. A. Mashudi<sup>3</sup>, W. Wibowo<sup>4</sup>, P. N. Malang, and P. Korespondensi, "Analisis Metode Cosine Similarity Pada Aplikasi Ujian Online Esai Otomatis (Studi Kasus JTI POLINEMA) Cosine Similarity Method Analysis On Automatic Esai Online Test Application", doi: 10.25126/jtiik.202184356.
  - [27] R. Fitri, A. Baris ke or Asyikin, and S. Pengajar Jurusan Teknik Elektro Politeknik Negeri Banjarmasin Ringkasan, "Aplikasi Penilaian Ujian Essay Otomatis Menggunakan Metode Cosine Similarity," vol. 7, baris ke 2, pp. 54–105, 2015.
  - [28] D. Rosmala and Z. M. Risyad, "Algoritma Levenshtein Distance dalam Aplikasi Pencarian Kata Isu di Kota Bandung pada Twitter," *MIND Journal | ISSN*, vol. ISSN, baris ke 2, pp. 1–12, 2017, doi: 10.26760/mindjournal.
  - [29] M. Adnan Nur, "Perbandingan Levenshtein Distance Dan Jaro-Winkler Distance Untuk Koreksi Kata Dalam Preprocessing Analisis Sentimen Pengguna Twitter," *Jurnal Fokus Elektroda Jurnal Fokus Elektroda: Energi Listrik, Telekomunikasi, Komputer, Elektronika dan Kendali) Teknik Elektro Universitas Halu Oleo Kendari Sulawesi Tenggara*, vol. 06 Baris ke 02, baris ke e-ISSN: 2502-5562, pp. 88–93, 2016.
  - [30] K. Sakaguchi, T. M. izumot, M. amoru Komachi, and Y. M. ji atsumot, "Joint English Spelling Error Correction and POS Tagging for Language Learners Writing," 2012.
  - [31] N. Hamidah, N. Yusliani, D. Rodiah, and ,<sup>3</sup>, "Spelling Checker using Algorithm Damerau Levenshtein Distance and Cosine Similarity," 2020. [Online]. Available: <http://sjia.ejournal.unsri.ac.id>
  - [32] R. Adawiyah and N. E. Saragih, "Implementasi Algoritma Levenshtein Distance Dalam Mendeteksi Plagiarisme."
  - [33] M. Kurniawan, K. Kusrini, and M. R. Arief, "Part of Speech Tagging Pada Teks Bahasa Indonesia dengan BiLSTM + CNN + CRF dan ELMo," *Jurnal Eksplora Informatika*, vol. 11, baris ke 1, pp. 29–37, Jan. 2022, doi: 10.30864/eksplora.v11i1.506.



