# DETEKSI DAN KOREKSI KESALAHAN EJAAN KATA BAHASA INDONESIA MENGGUNAKAN N-GRAM DAN JARO WINKLER PADA SISTEM PENILAIAN ESAI

**Proposal**


Disusun Oleh:

**JUDANTI CAHYANING TYAS**

**123170014**

**PROGRAM STUDI INFORMATIKA**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK INDUSTRI**

**UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"**

**YOGYAKARTA**


**2021**

**PERNYATAAN BEBAS PLAGIASI**

Saya yang bertanda tangan di bawah ini :

Nama : Judanti Cahyaning Tyas

NIM : 123170014

Fakultas/Prodi : Teknik Industri/Informatika

dengan ini saya menyatakan bahwa judul Proposal Tugas Akhir

Deteksi dan Koreksi Kesalahan Ejaan Kata Bahasa Indonesia Menggunakan N-Gram dan Jaro Winkler Pada Sistem Penilaian Esai

adalah hasil kerja saya sendiri dan benar bebas dari plagiasi kecuali cuplikan serta ringkasan yang terdapat di dalamnya telah saya jelaskan sumbernya (Sitasi) dengan jelas. Apabila pernyataan ini terbukti tidak benar maka saya bersedia menerima sanksi sesuai peraturan Mendiknas RI No 17 Tahun 2010 dan Peraturan Perundang-undangan yang berlaku.

Demikian surat pernyataan ini saya buat dengan penuh tanggung jawab.

Yogyakarta,

Yang membuat pernyataan

Judanti Cahyaning Tyas

NIM. 123170014

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

## DAFTAR PERSAMAAN

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Penilaian adalah sebuah komponen penting di dalam proses pembelajaran untuk mengukur tingkat kepahaman peserta didik terhadap materi yang diberikan dengan cara memberikan sebuah ujian. Saat ini, pelaksanaan ujian tidak hanya dapat dilakukan secara *offline* tetapi dapat dilakukan dengan cara *online*. Salah satu ujian yang biasa digunakan yaitu ujian dengan tipe uraian (*essay*). Ujian dengan tipe uraian (*essay*) merupakan sebuah ujian yang tidak disediakan pilihan jawaban dan seorang peserta didik harus menjawabnya dalam uraian kalimat sehingga dengan ujian tipe uraian (*essay*) akan dapat diketahui tingkat pemahaman seorang peserta didik terhadap materi yang diujikan (Rahimi & Asyikin, 2015). Akan tetapi, dalam mengoreksi ujian dengan tipe uraian (*essay*) ini membutuhkan waktu yang cukup lama sehingga mungkin saja kualitas penilaian akan menurun dan tidak bersifat objektif lagi. Kekurangan tersebut dapat diatasi dengan penilaian secara otomatis menggunakan komputer.

Ujian dengan tipe uraian (*essay*) secara *online* dibutuhkan ketelitian dalam menuliskan jawaban yang berarti tidak boleh adanya kesalahan pengetikan. Kesalahan pengetikan dapat disebabkan beberapa faktor seperti posisi *keyboard* yang berdekatan sehingga terjadi *slip* pada tangan atau jari, dan kurangnya konsentrasi dalam pengetikan (Frando et al., 2019). Kesalahan pengetikan menyebabkan sebuah kata tidak sesuai dengan ejaan kata sehingga akan mengubah makna kata seharusnya. Selain itu, kesalahan pengetikan dalam menjawab akan mengakibatkan berkurangnya nilai atau *scoring* ujian tersebut (Wihardodo, 2018). Oleh karena itu, kesalahan pengetikan dalam menjawab ujian tipe uraian (*essay*) perlu diatasi seperti deteksi kesalahan ejaan.

Beberapa penelitian telah dilakukan untuk mengatasi permasalahan kesalahan ejaan. Penelitian yang dilakukan oleh Samanta & Chaudhuri (2013) menggunakan 2 jenis metode n-gram yaitu bigram dan trigram. Metode tersebut diimplementasikan ke dalam bahasa Inggris dengan hasil presisi sebesar 71%-79% dan recall sebesar 81%-88%. Selain itu, metode tersebut dapat melakukan deteksi dan koreksi kata lebih dari satu kesalahan ejaan di dalam sebuah kalimat. Akan tetapi, masih banyak bigram dan trigram yang belum ditemukan karena kurangnya jumlah data n-gram. Kemudian, dari penelitian Samanta & Chaudhuri (2013) dilakukan pengembangan oleh Siregar (2017) yang diimplementasikan ke dalam bahasa Indonesia dengan hasil akurasi 11%. Akurasi yang rendah disebabkan karena penggunaan metode *additive smoothing* pada proses sebelum perhitungan skor peringkat sehingga kata yang salah dianggap benar dan kecilnya ukuran korpus yang digunakan. Penelitian lainnya yang menggunakan n-gram dilakukan oleh Wardhana et al (2011) dengan hasil penelitian bahwa bigram memiliki performansi yang lebih baik jika dibandingkan dengan trigram. Akan tetapi, jika dilihat dari waktu proses maka trigram lebih unggul dibandingkan dengan bigram. Selain itu, metode n-gram masih kurang baik untuk melakukan pengoreksian kata dengan jenis

kesalahan yang disebabkan oleh *substitusi* dan *transposisi*. Penelitian lain yang dilakukan oleh Christanti et al. (2018) menggunakan metode n-gram yang dikombinasikan dengan metode lainnya seperti *damerau levenshtein distance* dan trie. Hasil yang diperoleh yaitu tingkat akurasi terbaik pada metode bigram dan *damerau levenshtein distance* dengan kategori kalimat sebesar 50%, kategori kata sebesar 84,62% dan waktu proses di setiap kalimat sebesar 18,89 ms. Selain itu, metode *damerau levenshtein distance* jika dibandingkan dengan *levenshtein distance* akan memberikan hasil yang lebih baik karena *damerau levenshtein* lebih mampu menjangkau beberapa kesalahan ejaan. Metode deteksi kesalahan ejaan lainnya yang dapat digunakan yaitu *dictionary lookup* dan *damerau levenshtein distance* pada penelitian Maghfira et al. (2017). Dari penelitian tersebut dikatakan bahwa metode *dictionary look up* efektif untuk mengidentifikasi suatu kata dikatakan salah atau benar berdasarkan *lexical resource* dan *damerau levenshtein distance* lebih unggul dibandingkan *levenshtein distance* dengan memperoleh nilai presisi dan *recall* sebesar 0,76 dan 0,99. Akan tetapi, pada saat koreksi kata terdapat kekurangan dimana hasil koreksi kata lebih dari 1 kandidat dan tidak sesuai dengan nilai aktual tetapi memberikan koreksi dengan jarak edit yang sama. Selain itu, terdapat juga metode *jaro winkler* pada penelitian Wihardodo (2018) yang diimplementasikan ke dalam *essay scoring*. Metode tersebut mampu mendeteksi kesalahan ejaan berupa kelebihan huruf, kekurangan huruf, dan salah penempatan/transposisi huruf dengan tingkat akurasi sebesar 57%-73%. Akan tetapi, metode tersebut belum dapat mengoreksi kata yang penulisannya berdempet tanpa spasi (Frando et al., 2019). Kemudian, pada penelitian Rochmawati & Kusumaningrum (2015) melakukan perbandingan beberapa metode untuk deteksi kesalahan ejaan diantaranya metode *hamming distance*, *levenshtein distance*, *damerau levenshtein distance* dan *jaro winkler* yang diuji berdasarkan tingkat *Measuare Average Precision* (MAP). Hasil penelitian tersebut menunjukkan bahwa metode *jaro winkler* memperoleh nilai tertinggi *Measure Average Precision* (MAP) dibandingkan dengan metode lainnya sebesar 0,87 dengan pembagian beberapa tipe kesalahan seperti kesalahan penghapusan huruf sebesar 0,92, kesalahan penambahan huruf sebesar 0,90, kesalahan penggantian huruf sebesar 0,70, dan kesalahan penukaran huruf sebesar 0,95.

Dari beberapa metode yang pernah digunakan pada penelitian sebelumnya, maka pada penelitian ini akan menggunakan metode n-gram dan *jaro winkler* untuk mengatasi kesalahan ejaan yang akan diimplementasikan pada *essay scoring*. Jenis n-gram yang akan digunakan yaitu bigram dan trigram seperti pada penelitian yang pernah dilakukan oleh Samanta dan Chaudhuri (2013) karena hasil akurasi yang cukup baik dan kemampuan deteksi dan koreksi lebih dari satu kata dalam sebuah kalimat. Kemudian, pemilihan *jaro winkler* didasarkan pada nilai *Measure Average Precision* (MAP) yang lebih tinggi dibandingkan metode *approximate string matching* lainnya (Rochmawati & Kusumaningrum, 2015). Selain itu, menurut Okta'mal et al. (2015) dalam Indriyono (2020) bahwa *jaro winkler* memiliki kemampuan menjalankan proses yang lebih cepat dibandingkan algoritma *edit distance*/*levenshtein distance*. Metode *jaro winkler* digunakan untuk perhitungan jarak ketika pembuatan *confusion set* dimana pada penelitian sebelumnya menggunakan metode *levenshtein distance*. Diterapkannya metode *jaro winkler*

pada penelitian ini juga diharapkan dapat mengatasi kekurangan yang ada di n-gram yang masih kurang baik dalam mendeteksi jenis kesalahan ejaan *substitusi* dan transposisi. Untuk alur pemrosesan penelitian, pengguna dapat memasukkan teks berupa jawaban ujian bertipe uraian (*essay*) ke dalam sistem untuk selanjutnya dilakukan proses identifikasi kesalahan ejaan dan akan diberikan rekomendasi kata yang tepat menggunakan metode n-gram dan *jaro winkler* yang kemudian dilanjutkan dengan perhitungan nilai atau *scoring*. Penelitian ini mengharapkan metode n-gram dan *jaro winkler* mampu mengatasi kesalahan ejaan pada jawaban ujian bertipe uraian (*essay*).

## 1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan di atas, maka rumusan masalah dalam penelitian ini adalah bagaimana tingkat akurasi metode n-gram dan *jaro winkler* untuk mendeteksi dan koreksi kesalahan ejaan pada sistem *essay scoring*.

## 1.3 Batasan Masalah

Berdasarkan rumusan masalah tersebut, terdapat beberapa batasan masalah dalam penelitian ini yaitu :

1. Kamus kata diperoleh dari Kamus Besar Bahasa Indonesia (KBBI).
2. Data uji berupa jawaban soal ujian bertipe uraian (*essay*) dalam bentuk kalimat definisi berbahasa Indonesia.
3. Korpus kata diperoleh dari UI Tagged Corpus dan POS Tag Indonesia.
4. Tipe kesalahan ejaan yang digunakan yaitu kesalahan ejaan *real word* dan *non word*.

## 1.4 Tujuan Penelitian

Tujuan pada penelitian ini adalah mengetahui tingkat akurasi dari implementasi metode n-gram dan *jaro winkler* untuk deteksi dan koreksi kesalahan ejaan pada sistem *essay scoring.*

## 1.5 Manfaat Penelitian

Manfaat dari penelitian yang diharapkan dengan adanya deteksi dan koreksi kesalahan ejaan yang diterapkan di dalam *essay scoring* yaitu dapat mempermudah seorang pengguna dalam mendeteksi kesalahan ejaan pada suatu essay dimana kesalahan ejaan tersebut dapat mempengaruhi berkurangnya nilai pada saat proses *scoring*.

## 1.6 Metodologi Penelitian dan Pengembangan Sistem

Bagian ini menjelaskan metode yang digunakan di dalam penelitian. Metode tersebut sebagai berikut.

### 1.6.1 Metodologi Penelitian

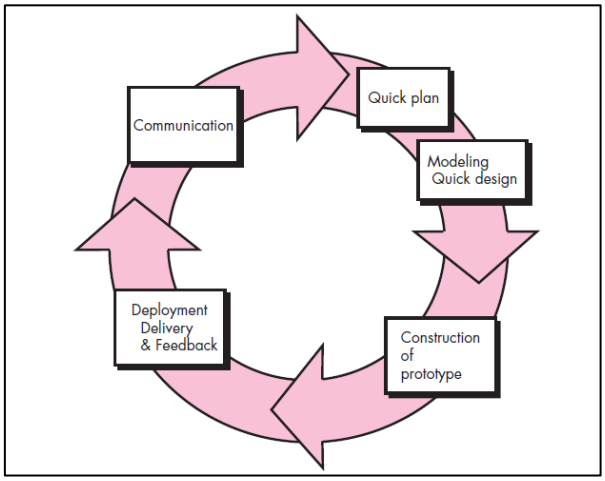
Metodologi penelitian yang digunakan pada penelitian ini menggunakan metode penelitian kuantitatif dengan beberapa tahap sebagai berikut.

1. Pengumpulan Data
2. Pembuatan Korpus N-gram
3. *Preprocessing* Deteksi dan Koreksi

4. Deteksi dan Koreksi Kesalahan Ejaan
5. *Scoring*
6. Pengujian

### 1.6.2 Metodologi Pengembangan Sistem

Metode pengembangan sistem dalam penelitian ini yaitu metode *prototyping*. Metode *prototyping* adalah salah satu metode yang memberikan pendekatan terbaik dan lebih umum untuk digunakan. Metode ini, dapat membantu seorang *developer* dan *client* untuk lebih memahami apa yang akan dibangun ketika kebutuhan yang dimiliki kurang jelas (Pressman, 2005). Langkah dalam metode *prototyping* sebagai berikut :




**Gambar 1. 1 Metode *Prototyping* (Pressman, 2005)**

1. Komunikasi
   Salah satu tahap awal dengan cara melakukan identifikasi kebutuhan perangkat lunak.
2. Membangun *prototype*
   Tahap ini merupakan tahap pembuatan rancangan sementara dari sebuah sistem.
3. Evaluasi *prototype*
   Tahapan ini digunakan untuk mengetahui sebuah *prototype* sudah sesuai yang diinginkan atau belum. Jika belum, perlu adanya perbaikan *prototype*.
4. *Coding*
   Tahap ini merupakan tahap dimana sebuah rancangan dari *prototype* diubah ke dalam bentuk kode program.
5. Pengujian dan evaluasi sistem
   Tahap ini digunakan untuk melakukan pengujian dan evaluasi sistem untuk mengetahui sistem telah dibuat sudah sesuai dengan yang diinginkan atau belum.
6. Penggunaan sistem

Tahap ini merupakan tahap terakhir dimana sistem siap digunakan dan sudah sesuai dengan yang diinginkan.

## 1.7 Sistematika Penulisan

Sistematika penulisan laporan penelitian ini sebagai berikut :

Bab I Pendahuluan

Bab ini membahas tentang latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan.

Bab II Tinjauan Literatur

Bab ini berisi tentang uraian dasar teori yang berkaitan dengan objek, masalah, dan metode dari pustaka ilmiah sebagai bahan referensi dan pondasi untuk memperkuat argumentasi dalam penelitian ini.

Bab III Metodologi Penelitian

Bab ini membahas tentang tahapan-tahapan metode penelitian dilakukan dimana tiap tahap akan menghasilkan hasilnya masing-masing.

Bab IV Hasil dan Pembahasan

Bab ini berisi tentang hasil dan pembahasan dari pengujian dan implementasi bab sebelumnya.

Bab V Kesimpulan dan Saran

Bab ini membahas tentang kesimpulan dan saran berdasarkan penelitian yang sudah dilakukan untuk penelitian selanjutnya agar lebih baik.

# BAB II
# TINJAUAN LITERATUR

## 2.1 *Automated Essay Scoring* (AES)

*Automated Essay Scoring* (AES) merupakan sistem yang digunakan untuk penilaian *essay* secara otomatis dengan membandingkan dua jawaban yang kemudian hasil perbandingan dihitung dengan menggunakan metode yang ada. Metode yang dapat diterapkan yaitu metode *corpus based similarity* seperti *Latent Semantic Analysis* (LSA) dan *string based similarity* seperti *cosine similarity* (Wihardodo, 2018). Penggunaan AES ini memiliki kelebihan dan kekurangan. Kelebihan penggunaan AES yaitu dapat mengurangi waktu yang dibutuhkan penilai dalam melakukan penilaian sebuah *essay* dalam jumlah banyak dan adanya konsistensi dalam penilaian. Sedangkan kelemahannya yaitu *similarity* perhitungan nilai yang dihasilkan oleh sistem dengan penilai masih sulit dicapai (Adhitia & Purwarianti, 2012).

## 2.2 Kalimat definisi

Kalimat definisi merupakan salah satu jenis kalimat yang memberikan sebuah penjelasan atau keterangan atau uraian tentang sebuah objek yang dapat membantu seorang pembaca dapat untuk memahami sebuah tulisan (Setiawan, 2020). Objek yang dapat digunakan berupa benda, orang, proses atau sebuah aktivitas(Wihardodo, 2018). Ciri-ciri kalimat definisi sebagai berikut (Wihardodo, 2018) :

1. Menjelaskan sebuah makna atau arti dari sebuah objek.
2. Penjelasan pada sebuah objek bersifat umum bukan khusus.
3. Dapat ditemukan di laporan ilmiah atau karya tulis lainnya.
4. Apabila kalimat definisi dibalik maka tidak akan mengubah makna atau arti dari kalimat definisi tersebut.

## 2.3 Kesalahan Ejaan

Kesalahan ejaan merupakan kesalahan penulisan susunan kata yang disebabkan karena kesalahan pengetikan atau kurang pahamnya seseorang tentang ejaan kata yang benar (Maghfira et al., 2017). Kesalahan ejaan terdiri dari 2 kategori diantaranya :

1. Kesalahan ejaan *non word*.
   Kesalahan ejaan *non word* merupakan kategori kesalahan ejaan dimana kata yang salah tidak termasuk kata yang *valid* dan tidak dapat ditemukan di dalam *lexicon* (Kaur et al., 2019). Contoh : “Jawabsn soal ini sudah benar”. Kata “jawabsn” di dalam kalimat tersebut termasuk kategori kesalahan *non word* karena kata “jawabsn” tidak terdapat di dalam Kamus Besar Bahasa Indonesia (KBBI).
2. Kesalahan ejaan *real word*.
   Kesalahan ejaan *real word* merupakan kategori kesalahan ejaan dimana kata yang salah termasuk kata *valid* yang dapat ditemukan di dalam *lexicon*. Akan tetapi, kata tersebut tidak sesuai dengan susunan kata yang dibentuk di dalam kalimat (Kaur et al., 2019). Contoh : “Jawatan soal ini sudah benar”. Kata “jawatan” di dalam kalimat

tersebut termasuk kategori kesalahan *real word* karena kata tersebut bukan kata yang dimaksud yaitu "jawaban" walaupun kata tersebut terdapat di dalam Kamus Besar Bahasa Indonesia (KBBI).

## 2.4 *Natural Language Processing* (NLP)

*Natural Language Processing* (NLP) merupakan bagian dari bidang ilmu *artificial intelligent* yang berfokus pada interaksi antara manusia dengan komputer dengan menggunakan bahasa komputer untuk membedakan bahasa alami manusia sehingga ketika berkomunikasi dengan komputer seolah-olah sedang berkomunikasi dengan manusia (Kumar,2011). Beberapa metode yang dapat diterapkan seperti *automatic summarization*, *part of speech*, *machine translation*, *disambiguation*, dan lain sebagainya. *Natural language processing* ini dalam menjalankan prosesnya membutuhkan sebuah basis pengetahuan seperti *lexicon of word* dan beberapa macam dataset yang berisi grammatikal atau aturan tentang linguistik yang terbaru (Maghfira et al., 2017).

## 2.5 *Text Mining*

*Text mining* adalah proses penggalian data berupa teks yang tidak terstruktur dari sebuah dokumen yang tujuannya untuk menemukan informasi penting dari teks yang dapat mewakili isi dari dokumen yang kemudian dapat digunakan sebagai bahan analisis yang lebih lanjut. Kemudian, teks yang tidak terstruktur perlu diproses terlebih dahulu dengan beberapa tahapan awal yang bertujuan untuk mempersiapkan teks agar lebih terstruktur yang biasanya disebut dengan *preprocessing* (Maghfira et al., 2017). Beberapa tahapan yang dapat dilakukan pada preprocessing diantaranya :

1. *Case folding*.
   *Case folding* atau yang dapat disebut dengan *toLowerCase* merupakan sebuah proses yang digunakan untuk mengubah semua karakter huruf besar menjadi huruf kecil. Kemudian, karakter huruf yang diterima hanya huruf a sampai dengan z (Ariyani et al., 2016). Contoh proses *case folding* pada kata "MAIN", "Main", "maIn", "MaiN" diubah menjadi "main" dimana karakter hurufnya sudah menjadi huruf kecil.
2. *Tokenizing*.
   *Tokenizing* merupakan sebuah proses memisahkan atau memotong kata penyusun teks seperti kalimat, paragraf atau dokumen menjadi token kata tunggal (Rochmawati & Kusumaningrum, 2015). Proses *tokenizing* memerlukan sebuah *delimiter* yang berfungsi sebagai pemisah satu kata dengan kata lainnya dengan menggunakan karakter di luar alfabet seperti karakter spasi. Output dari proses ini yaitu berupa array token kata (Frando et al., 2019).
3. *Filtering*.
   *Filtering* merupakan salah satu tahap yang bertujuan untuk pengambilan beberapa kata yang dianggap penting dari hasil proses *tokenizing*. Proses filtering ini dapat menggunakan algoritma *stoplist* atau *wordlist*. *Stoplist* digunakan untuk membuang

kata yang dianggap kurang penting seperti kata “yang”, “dan”, “di”, dan lain sebagainya. Sedangkan *wordlist* digunakan untuk menyimpan kata yang dianggap penting (Suryaningrum, 2019).

4. *Stemming*.

   *Stemming* merupakan sebuah proses yang digunakan untuk mengubah sebuah kata yang terdapat di dalam teks menjadi bentuk kata dasar dengan menghilangkan imbuhan yang terdapat pada kata. Salah satu algoritma *stemming* yaitu algoritma Nazief dan Andriani. Algoritma tersebut merupakan algoritma yang sering digunakan untuk *stemming* bahasa Indonesia. Selain itu, algoritma tersebut dikembangkan berdasarkan morfologi bahasa Indonesia dengan mengelompokkan imbuhan menjadi awalan, sisipan, akhiran, dan gabungan awalan dan akhiran (Ariyani et al., 2016).

Tahapan *preprocessing* diatas, tidak harus semuanya digunakan. Akan tetapi, penggunaannya disesuaikan dengan kebutuhan analisis teks yang akan dilakukan. Beberapa analisis yang dapat dilakukan seperti *Natural Language Processing* (NLP), *Named Entity Recognition* (NER), *Information Retrieval* (IR), dan lain sebagainya. (Maghfira et al., 2017).

## 2.6 N-gram

N-gram merupakan sebuah algoritma yang memperhatikan urutan n-huruf yang berdekatan berupa kata, huruf, suku kata atau fonem yang dilakukan dengan cara membagi sebuah kalimat atau kata menjadi lebih kecil yang kemudian dilakukan perhitungan probabilitas secara keseluruhan (Christanti et al., 2018). Selain itu, n-gram tidak membutuhkan sebuah kamus atau pengetahuan linguistik untuk menjalankan prosesnya (Azmi et al., 2019). Nilai n yang dapat digunakan yaitu 1, 2, 3, dan 4. Untuk n=1 disebut unigram, n=2 disebut bigram, n=3 disebut trigram dan n=4 disebut quadgram (Indriani et al., 2018). Penggunaan nilai n pada n-gram dapat dilihat pada contoh kalimat “saya suka makan” sebagai berikut :

Unigram : “saya”, “suka”, “makan”
Bigram : “_ saya”, “saya suka”, “suka makan”, “makan _”
Trigram : “_ saya suka”, “saya suka makan”, “suka makan _“
Quadgram : “_ saya suka makan”, “saya suka makan _”

Perhitungan probabilitas pada metode n-gram salah satunya dengan menggunakan asumsi *Markov Chain* dimana asumsi tersebut menyatakan bahwa probabilitas beberapa kata berikutnya hanya bergantung pada kata sebelumnya (Samanta & Chaudhuri, 2013). Dari asumsi tersebut, perhitungan probabilitas dihitung dengan menggunakan *Maximum Likelihood Estimation* (MLE) dengan cara menghitung kemunculan n-gram pada korpus yang kemudian dinormalisasikan dengan membaginya dengan nilai total agar hasil yang didapatkan diantara 0 dan 1 (Jurafsky & Martin, 2020). Persamaan probabilitas unigram, bigram dan trigram sebagai berikut.

Unigram

$$P_0(C_j^i) = \frac{count\ (C_j^i)}{\sum_{r=0}^{k_i} count\ (C_r^i)} \quad \text{.....(2. 1)}$$

Bigram kiri

$$P_1\left(W_j^i \middle| W^{i-1}\right) = \frac{count\ (W^{i-1}C_j^i)}{\sum_{r=0}^{k_i} count\ (W^{i-1}C_r^i)} \quad \text{.....(2. 2)}$$

Bigram kanan

$$P_2\left(W_j^i \middle| W^{i+1}\right) = \frac{count\ (W^{i+1}C_j^i)}{\sum_{r=0}^{k_i} count\ (W^{i+1}C_r^i)} \quad \text{.....(2. 3)}$$

Trigram

$$P_3\left(W_j^i \middle| W^{i-1}, W^{i+1}\right) = \frac{count\ (W^{i-1}C_j^iW^{i+1})}{\sum_{r=0}^{k_i} count\ (W^{i-1}C_r^iW^{i+1})} \quad \text{.....(2. 4)}$$

Keterangan :

| | |
|---|---|
| $count\ (C_j^i)$ | : jumlah kemunculan pada unigram. |
| $\sum_{r=0}^{k_i} count\ (C_r^i)$ | : total jumlah kemunculan pada unigram. |
| $count\ (W^{i-1}C_j^i)$ | : jumlah kemunculan pada bigram kiri. |
| $\sum_{r=0}^{k_i} count\ (W^{i-1}C_r^i)$ | : total jumlah kemunculan pada bigram kiri. |
| $count\ (W^{i+1}C_j^i)$ | : jumlah kemunculan pada bigram kanan. |
| $\sum_{r=0}^{k_i} count\ (W^{i+1}C_r^i)$ | : total jumlah kemunculan pada bigram kanan. |
| $count\ (W^{i-1}C_j^iW^{i+1})$ | : jumlah kemunculan pada trigram. |
| $\sum_{r=0}^{k_i} count\ (W^{i-1}C_r^iW^{i+1})$ | : total jumlah kemunculan pada trigram. |

Pada penelitian (Siregar, 2017), perhitungan probabilitas dengan menggunakan persamaan 2.1, 2.2 dan 2.3 menghasilkan banyak probabilitas yang bernilai 0 yang kemudian diatasi dengan menambahkan salah satu perhitungan metode *smoothing* yaitu *additive smoothing*. Cara kerja *additive smoothing* yaitu menambahkan nilai $\delta$ pada perhitungan count yang bernilai 0,01. Penambahan nilai tersebut dengan maksud agar dalam perhitungan tidak menghasilkan nilai yang akan mempengaruhi perhitungan selanjutnya. Perhitungan probabilitas dengan penambahan metode *additive smoothing* sebagai berikut.

Unigram

$$P_0(C_j^i) = \frac{count\left(C_j^i\right)+\delta}{\sum_{r=0}^{k_i} count\ (C_r^i)+\delta * V_j} \quad \text{.....(2. 5)}$$

Bigram kiri

$$P_1(W_j^i|W^{i-1}) = \frac{count\left(W^{i-1}C_j^i\right)+\delta}{\sum_{r=0}^{k_i} count\,(W^{i-1}C_r^i)+\delta*V_j} \quad \text{.....(2. 6)}$$

Bigram kanan

$$P_2(W_j^i|W^{i+1}) = \frac{count\left(W^{i+1}C_j^i\right)+\delta}{\sum_{r=0}^{k_i} count\,(W^{i+1}C_r^i)+\delta*V_j} \quad \text{.....(2. 7)}$$

Trigram

$$P_3(W_j^i|W^{i-1}, W^{i+1}) = \frac{count\left(W^{i-1}C_j^iW^{i+1}\right)+\delta}{\sum_{r=0}^{k_i} count\,(W^{i-1}C_r^iW^{i+1})+\delta*V_j} \quad \text{.....(2. 8)}$$

Keterangan :

| | |
|---|---|
| $count\ (C_j^i)$ | : jumlah kemunculan pada unigram. |
| $\sum_{r=0}^{k_i} count\ (C_r^i)$ | : total jumlah kemunculan pada unigram. |
| $count\ (W^{i-1}C_j^i)$ | : jumlah kemunculan pada bigram kiri. |
| $\sum_{r=0}^{k_i} count\ (W^{i-1}C_r^i)$ | : total jumlah kemunculan pada bigram kiri. |
| $count\ (W^{i+1}C_j^i)$ | : jumlah kemunculan pada bigram kanan. |
| $\sum_{r=0}^{k_i} count\ (W^{i+1}C_r^i)$ | : total jumlah kemunculan pada bigram kanan. |
| $count\ (W^{i-1}C_j^iW^{i+1})$ | : jumlah kemunculan pada trigram. |
| $\sum_{r=0}^{k_i} count\ (W^{i-1}C_r^iW^{i+1})$ | : total jumlah kemunculan pada trigram. |
| $\delta$ | : nilai konstanta sebesar 0,01. |
| $V_j$ | : jumlah anggota *confusion set.* |

### *2.6.1 Jelinek mercer*

*Jelinek mercer* merupakan salah satu metode *smoothing* yang digunakan untuk mengatasi hasil probabilitas bernilai 0 yang diberikan oleh model bahasa (Christanti et al., 2018). Selain itu, *jelinek mercer* ini termasuk ke dalam metode *smoothing* yang menggunakan cara interpolasi dimana metode ini berhubungan dengan n-gram di bawah tingkatannya (Sihole, 2018). Persamaan sebagai berikut :

Bigram kiri

$$P_1'(C_j^i|W^{i-1}) = \lambda P_1(C_j^i|W^{i-1}) + (1-\lambda)P_0(C_j^i) \quad \text{.....(2. 9)}$$

Bigram kanan

$$P_2'(C_j^i|W^{i+1}) = \lambda P_2(C_j^i|W^{i+1}) + (1-\lambda)P_0(C_j^i) \quad \text{.....(2. 10)}$$

Trigram

$$P_3'(C_j^i|W^{i-1}, W^{i+1})$$

$$= \lambda P_3(C_j^i|W^{i-1}, W^{i+1}) + (1 - \lambda)(P_1(C_j^i|W^{i+1}) + P_2(C_j^i|W^{i+1})/2) \text{.....(2. 11)}$$

Keterangan :

$\lambda$ : bobot dengan nilai 0,1.

$P_0(C_j^i)$ : nilai probabilitas *Maximum Likelihood Estimation* unigram pada persamaan 2.5.

$P_1(C_j^i|W^{i-1})$ : nilai probabilitas *Maximun Likelihood Estimation* bigram kiri pada persamaan 2.6.

$P_2(C_j^i|W^{i+1})$ : nilai probabilitas *Maximum Likelihood Estimation* bigram kanan pada persamaan 2.7.

$P_3(C_j^i|W^{i-1}, W^{i+1})$ : nilai probabilitas *Maximum Likelihood Estimation* trigram pada persamaan 2.8.

### *2.6.2 Weighted combination score*

N-gram memiliki orde tinggi dan rendah yang memiliki kelebihan dan kekurangannya masing-masing. N-gram dengan orde yang tinggi akan memiliki sifat lebih sensitif terhadap sebuah konteks dimana jumlah hitungannya hanya sedikit. Sedangkan n-gram dengan orde yang rendah memiliki keterbatasan untuk mengenal sebuah konteks dimana jumlah hitungannya akan lebih banyak jika dibandingkan dengan n-gram orde yang tinggi. Untuk mengatasi hal tersebut, kemudian diimplementasikan sebuah perhitungan menggunakan *weighted combination score* dengan cara menggabungkan hasil yang sudah diperoleh dari bigram dan trigram (Samanta & Chaudhuri, 2013).

$$Score\ (C_j^i)$$

$$= \lambda_1 P_1(C_j^i|W^{i-1}) + \lambda_2 P_2(C_j^i|W^{i+1}) + \lambda_3 P_3(C_j^i|W^{i-1}, W^{i+1}) \text{.....} \textit{(2. 12)}$$

Keterangan :

$\lambda_1$ : bobot untuk probabilitas bigram kiri dengan nilai 0,25.

$\lambda_2$ : bobot untuk probabilitas bigram kanan dengan nilai 0,25.

$\lambda_3$ : bobot untuk probabilitas trigram dengan nilai 0,5.

$P_1(C_j^i|W^{i-1})$ : nilai probabilitas pada bigram kiri.

$P_2\left(C_j^i \middle| W^{i+1}\right)$ : nilai probabilitas pada bigram kanan.

$P_3\left(C_j^i \middle| W^{i-1}, W^{i+1}\right)$ : nilai probabilitas pada trigram.

## 2.7 *Jaro Winkler*

*Jaro winkler* merupakan sebuah algoritma yang termasuk di dalam varian *jaro distance metric* untuk mengukur kesamaan antara dua buah string yang dibandingkan (Rochmawati & Kusumaningrum, 2015). Biasanya algoritma ini digunakan untuk pendeteksian duplikat dan cocok digunakan untuk string yang pendek (Yulianingsih, 2017). Selain itu, algoritma ini dapat bekerja lebih cepat dibandingkan dengan algoritma *edit distance* (Okta'mal et al., 2015). Kemudian, untuk mengetahui kesamaan diantara dua buah string dapat dilihat dari nilai *jaro winkler* yang dihasilkan. Semakin tinggi nilai *jaro winkler* yang dihasilkan maka menandakan adanya kesamaan. Nilai normal *jaro winkler* yaitu 0 yang menandakan bahwa tidak ada kesamaan dan bernilai 1 yang menandakan adanya kesamaan diantara dua buah string yang dibandingkan (Tinaliah & Elizabeth, 2018). Menurut Yulianingsih (2017), terdapat 3 tahapan dasar di dalam proses algoritma *jaro winkler* diantaranya:

1. Menghitung panjang string.
2. Menemukan jumlah karakter yang sama diantara dua buah string.
3. Menemukan jumlah transposisi diantara dua buah string.

Perhitungan jarak diantara dua buah string ditunjukkan pada persamaan sebagai berikut :

$$d_j = \frac{1}{3}\left(\frac{m}{|S_1|} + \frac{m}{|S_2|} + \frac{m-t}{m}\right) \quad (2.13)$$

Keterangan :

$d_j$ : nilai *jaro distance* untuk string $S_1$ dan $S_2$.

$|S_1|$ : jumlah huruf dari kata pertama.

$|S_2|$ : jumlah huruf dari kara kedua.

$m$ : jumlah huruf yang memiliki kesamaan.

$t$ : jumlah transposisi huruf.

Jarak teoritis dua buah string benar dianggap sama jika tidak melebihi batas dengan persamaan sebagai berikut :

$$Jarak\ teoritis = \left(\frac{\max(|S_1|,|S_2|)}{2}\right) - 1 \quad (2.14)$$

Perhitungan *jaro winkler* dengan menggunakan persamaan sebagai berikut :

$$d_w = d_j + (lp\ (1 - d_j)) \quad (2.15)$$

Keterangan :

$d_w$ : jarak *jaro winkler*.

$d_j$ : nilai *jaro distance*.

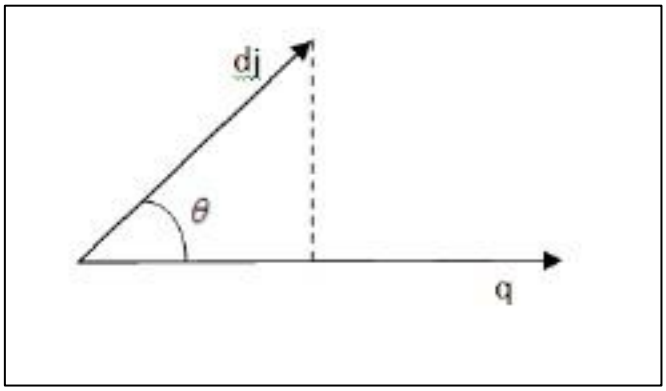
$l$ : panjang awalan karakter yang sama sebelum ditemukannya ketidaksamaan dengan nilai maksimum 4 karakter.

$p$ : konstanta *scaling factor*. Menurut winkler, nilai standar konstanta ini sebesar 0,1.

## 2.8 Vector Space Model (VSM)

*Vector Space Model* (VSM) merupakan sebuah metode yang digunakan untuk mencari kesamaan diantara sebuah dokumen dengan sebuah *query* dimana masing-masing dokumen dan *query* yang ada direpresentasikan ke dalam bentuk n dimensi vektor (Robinson, 2014). Jumlah kata yang signifikan di dalam sebuah dokumen menjadi sebuah penentu banyaknya dimensi ruang vektor yang digunakan (Sulistyo et al., 2015).

Sebuah dokumen dan *query* yang direpresentasikan ke dalam bentuk n dimensi vektor dapat dilihat pada gambar berikut :




**Gambar 2. 1 Representasi dokumen dan *query* pada ruang vektor**

Keterangan :

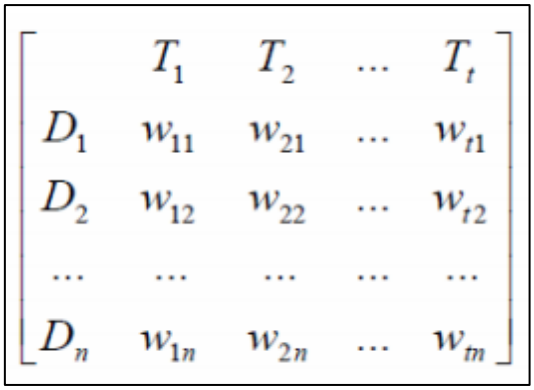
$d_j$ : dokumen.

$\theta$ : sudut.

$q$ : *query*.

Di dalam *Vector Space Model* (VSM), koleksi yang terdapat di dalam sebuah dokumen direpresentasikan ke dalam bentuk matriks *term document* atau *term frequency* dimana *input* di setiap sel matriksnya bersesuaian dengan bobot *term* yang dihitung dengan konsep *Term*

*Frequency* (TF) dan *Inverse Document Frequency* (IDF). Apabila terdapat nilai 0 maka keberadaan *term* tidak terdapat di dalam dokumen (Mandala, 2006). Matriks *term document* dapat dilihat pada gambar sebagai berikut :




**Gambar 2. 2 Matriks *Vector Space Model* (VSM)**

### 2.8.1 *Term Frequency* (TF) dan *Inverse Document Frequency* (IDF)

*Term Frequency* (TF) merupakan salah satu cara yang dapat digunakan untuk pembobotan sebuah *term* yang didasarkan pada jumlah kemunculan sebuah *term* di dalam dokumen. Apabila sebuah *term* sering muncul di dalam sebuah dokumen maka term tersebut akan memiliki nilai tf yang tinggi atau semakin besar bobot *term* di dalam dokumen tersebut jika dibandingkan dengan *term* yang lainnya (Sulistyo et al., 2015). Akan tetapi, dalam proses pembobotan sebuah *term* dengan konsep *Term Frequency* (TF) saja belum cukup maka diperlukannya faktor pembobotan lain yaitu dengan menggunakan konsep *Inverse Document Frequency* (IDF).

*Inverse Document Frequency* (IDF) merupakan *inverse* dari *Document Frequency* (DF) yang digunakan untuk mengetahui jumlah kemunculan sebuah *term* pada dokumen lain yang terdapat di dalam *database* (Wihardodo, 2018). Apabila nilai IDF pada sebuah *term* semakin tinggi maka *term* tersebut penting atau tidak banyak digunakan di dalam sebuah dokumen. Dan sebaliknya apabila nilai IDF semakin kecil maka *term* tersebut tidak penting atau banyak digunakan di dalam sebuah dokumen (Mandala, 2006).

Perhitungan *Term Frequency* (TF) pada sebuah *term* dengan menggunakan persamaan sebagai berikut :

$$tf = tfij \quad (2.16)$$

Keterangan :

$tf$ : nilai *Term Frequency* (TF) yang diperoleh.

$tfij$ : jumlah kemunculan *term* $ti$ di dalam suatu dokumen $dj$

Perhitungan *Inverse Document Frequency* (IDF) pada sebuah term dengan menggunakan persamaan sebagai berikut :

$$idfi = \log \frac{N}{dfi} \quad \text{.....(2.17)}$$

Keterangan :

$idfi$ ..... : nilai *Inverse Document Frequency* (IDF) yang diperoleh.

$N$ : jumlah dokumen yang terambil.

$dfi$ : jumlah dokumen di dalam koleksi dimana muncul *term* $ti$ di dalamnya.

Perhitungan *Term Frequency* (TF) dan *Inverse Document Frequency* (IDF) dengan menggunakan persamaan sebagai berikut :

$$W_{ij} = tfij \,.\log \frac{N}{dfi} \quad \text{.....(2.18)}$$

Keterangan :

$W_{ij}$ : bobot dokumen yang diperoleh dari kombinasi perkalian antara *Term Frequency* (TF) dengan *Inverse Document Frequency* (IDF)

$tfij$ : nilai *Term Frequency* (TF)

$\log \frac{N}{dfi}$ : nilai *Inverse Document Frequency* (IDF)

Apabila nilai pada $N = dfi$ maka hasil yang diperoleh $W_{ij}$ yaitu nol, berapapun nilai pada $tfij$ (Sulistyo et al., 2015). Untuk mengatasi hal tersebut, perlu ditambahkan nilai 1 pada sisi *Inverse Document Frequency* (IDF) dengan persamaan sebagai berikut :

$$W_{ij} = tfij \,.(\log \frac{N}{dfi} + 1) \quad \text{.....(2.19)}$$

### 2.8.2 *Matching Documents*

*Matching documents* merupakan sebuah tahapan yang digunakan untuk melakukan perhitungan tingkat kemiripan antar dokumen yaitu antar dokumen jawaban siswa dengan jawaban pengajar (kunci jawaban) (Sulistyo et al., 2015). Tahapan ini lebih tepat didasarkan pada perhitungan sudut yang dibentuk antara vektor dokumen dengan *query* dimana semakin kecil sudut yang dibentuk maka relevansi antar keduanya akan semakin besar. Untuk mengatasi hal tersebut, dapat menerapkan konsep yang terdapat di dalam metode *cosine similarity* dengan perhitungan kosinus sudut yang terbentuk diantara vektor dokumen dan *query*. Apabila sudut yang dibentuk kecil maka nilai kosinus akan besar, dan apabila sudut yang dibentuk besar maka nilai kosinus akan kecil (Mandala, 2006). Perhitungan *cosine similarity* dengan menggunakan persamaan sebagai berikut :

$$Sim\ (q,d) = \cos\theta = \frac{q\ .\ d}{|q||d|} = \frac{\sum_{k=1}^{t} W_{qk}.W_{dk}}{\sqrt{\sum_{k=1}^{t}(W_{qk})^2}\sqrt{\sum_{k=1}^{t}(W_{dk})^2}} \quad \text{.....(2.20)}$$

Keterangan :

$q$ : dokumen yang dijadikan sebagai acuan.

$d$ : dokumen ke-i.

$W_{qk}$ : bobot *query* atau *term* yang didapatkan dari dokumen acuan.

$W_{dk}$ : bobot *term* yang diperoleh pada dokumen ke-i.

Apabila nilai $D = D_i$ maka nilai yang diperoleh pada perhitungan $Sim\ (D, D_i)$ yaitu 1. Selain itu, apabila $D$dan $D_i$ berbeda atau tidak adanya kesamaan maka nilai yang diperoleh pada perhitungan $Sim\ (D, D_i)$ yaitu 0 (Sulistyo et al., 2015). Hasil perhitungan tingkat kesamaan akan mendekati nilai 1 atau lebih besar dibandingkan dengan perhitungan kesamaan yang dihitung dengan perhitungan *inner product* (Wihardodo, 2018).

### 2.8.3 Konversi Nilai Kemiripan

Konversi nilai kemiripan ini digunakan untuk melakukan konversi nilai yang diperoleh dari perhitungan similaritas ke dalam bentuk nilai ujian uraian (*essay*) pada umumnya. Rentang nilai menurut Fuat (2010) dalam Rahimi & Asyikin (2015) dapat dilihat pada tabel 2.1 sebagai berikut.

**Tabel 2. 1 Rentang Nilai Untuk Konversi Nilai (Fuat, 2010)**

| Nilai Perhitungan Similaritas | Nilai *Human Rates* |
|---|---|
| 0,01-0,10 | 10 |
| 0,11-0,20 | 20 |
| 0,21-0,30 | 30 |
| 0,31-0,40 | 40 |
| 0,41-0,50 | 50 |
| 0,51-0,60 | 60 |
| 0,61-0,70 | 70 |
| 0,71-0,80 | 80 |
| 0,81-0,90 | 90 |
| 0,91-1 | 100 |

Pada tabel 2.1 yang berisi rentang nilai yang digunakan untuk konversi nilai bergantung pada jumlah soal yang dibuat. Dimisalkan jika ada 5 soal dan sebuah jawaban soal memperoleh nilai similaritas sebesar 1, maka nilai dari jawaban soal tersebut yaitu 100/5=20.

## 2.9 Pengujian Deteksi dan Koreksi Kesalahan Ejaan

Pengujian pada penelitian ini digunakan untuk mengetahui seberapa akurat hasil dari deteksi dan koreksi kesalahan ejaan dengan menggunakan metode n-gram dan *jaro winkler*. Pengujian yang digunakan yaitu dengan cara melakukan perhitungan akurasi, presisi dan recall. Di dalam penelitian ini, presisi digunakan untuk melakukan perhitungan persentase dari metode yang digunakan dapat memberikan rekomendasi kata yang benar atau akurat dari semua rekomendasi kata yang diberikan oleh sistem. Sedangkan recall digunakan untuk melakukan perhitungan persentase sebuah metode dapat memberikan rekomendasi kata yang benar atau akurat dari semua rekomendasi kata sebenarnya yang dianggap benar (Simanjuntak et al., 2018). Matriks presisi dan recall dapat dilihat pada tabel 2.2 sebagai berikut.

**Tabel 2. 2 Recall dan Presisi (Wihardodo, 2018)**

| | | Nilai Sebenarnya | |
|---|---|---|---|
| | | True | False |
| Nilai Prediksi | True | TP | FP |
| | False | FN | TN |

Dari matriks pada tabel 2.2 dapat dilakukan perhitungan dengan menggunakan persamaan sebagai berikut.

$$presisi = \frac{TP}{TP+FP} \quad (2.21)$$

$$recall = \frac{TP}{TP+FN} \quad (2.22)$$

$$akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.23)$$

Keterangan :

$TP$ : *true positive*, jumlah kata yang dihasilkan oleh sistem dan benar

$TN$ : *true negative*, jumlah kata yang dihasilkan oleh sistem dan tidak benar

$FP$ : *false positive*, jumlah kata yang benar tetapi tidak dihasilkan oleh sistem

$FN$ : *false negative*,jumlah kata yang salah dan tidak dihasilkan oleh sistem

Dari keterangan diatas, dapat disimpulkan bahwa jika sistem menghasilkan hasil yang benar maka dikategorikan ke dalam TP (*True Positive*). Apabila sistem menghasilkan hasil yang salah, maka dikategorikan ke dalam TN (*True Negative*). Dan apabila sistem tidak dapat menghasilkan rekomendasi kata apapun maka dikategorikan ke dalam FN (*False Negative*) (Simanjuntak et al., 2018).

## 2.10 Studi Pustaka

Adapun penelitian yang telah dilakukan oleh peneliti sebelumnya yang relevan dengan penelitian yang akan dilakukan sebagai acuan dan referensi.

1. Penelitian yang dilakukan oleh (Samanta & Chaudhuri, 2013) dengan judul **A Simple Real Word Error Detection and Correction Using Local Word Bigram and Trigram.** Pada penelitian ini melakukan pendeteksian dan koreksi kesalahan ejaan jenis *real*

*word* menggunakan 3 jenis n-gram yaitu bigram kiri, bigram kanan, dan trigram yang diimplementasikan ke dalam bahasa Inggris. Proses pendeteksian terlebih dahulu membuat sebuah *confusion set* disetiap kandidat kata. Pembuatan *confusion set* dengan perhitungan jarak menggunakan metode *levenshtein distance*. Setelah pembuatan *confusion set*, dilanjutkan dengan pembuatan model n-gram yang kemudian dilakukan perhitungan peringkat disetiap elemen yang terdapat di dalam *confusion set*. Hasil dari perhitungan peringkat kemudian diurutkan dari nilai terbesar ke terkecil. Nilai terbesar merupakan sugesti kata yang dihasilkan. Pada penelitian ini menghasilkan presisi sebesar 71%-79% dan recall sebesar 81%-88%. Selain itu, metode pada penelitian ini memiliki kemampuan dapat mendeteksi dan koreksi kesalahan ejaan lebih dari satu di dalam sebuah kalimat. Akan tetapi, masih banyak bigram dan trigram yang belum dapat ditemukan karena kurangnya data atau korpus pada n-gram.

2. Penelitian yang dilakukan oleh (Siregar, 2017) dengan judul **Deteksi dan Koreksi Kesalahan Real Word Pada Penulisan Kata Menggunakan Metode N-Gram Dalam Teks Berbahasa Indonesia**. Penelitian ini merupakan pengembangan dari penelitian sebelumnya yang dilakukan oleh Samanta & Chaudhuri yang diimplementasikan ke dalam bahasa Indonesia. Perbedaan lain dari penelitian ini dengan sebelumnya yaitu pada proses sebelum perhitungan peringkat dimana dilakukan perhitungan dengan metode *additive smoothing* untuk mengatasi hasil probabilitas yang bernilai 0. Hasil penelitian yang diperoleh berupa akurasi sebesar 11%. Akurasi yang rendah disebabkan karena ukuran korpus n-gram yang digunakan dan metode *additive smoothing* yang menyebabkan kata yang seharusnya salah menjadi benar.

3. Penelitian yang dilakukan oleh (Rochmawati & Kusumaningrum, 2015) dengan judul **Studi Perbandingan Algoritma Pencarian String Dalam Metode Approximate String Matching Untuk Identifikasi Kesalahan Pengetikan Teks.** Penelitian ini melakukan perbandingan beberapa metode *approximate string matching* yang hasil pengujiannya berupa nilai *Measure Average Precision* (MAP). Beberapa metode yang dibandingkan diantaranya *hamming distance*, *levenshtein distance*, *damerau levenshtein distance*, dan *jaro winkler*. Hasil yang diperoleh yaitu *jaro winkler* memperoleh nilai Measure Average Precision (MAP) tertinggi dibandingkan metode lainnya sebesar 0,87 dengan pembagian beberapa tipe kesalahan seperti kesalahan penghapusan huruf sebesar 0,92, kesalahan penambahan huruf sebesar 0,90, kesalahan penggantian huruf sebesar 0,70, dan kesalahan penukaran huruf sebesar 0,95.

4. Penelitian yang dilakukan oleh (Wihardodo, 2018) dengan judul **Implementasi Perbaikan Kesalahan Ejaan Pada Sistem *Essay Scoring*.** Penelitian ini mengimplementasikan deteksi kesalahan ejaan di dalam *essay scoring* dengan menggunakan metode *jaro winkler*. Hasil yang diperoleh pada penelitian ini berupa tingkat akurasi sebesar 57%-73%. Selain itu, metode yang digunakan dikatakan dapat mendeteksi kesalahan ejaan berupa kelebihan huruf, kekurangan huruf, dan salah penempatan/transposisi huruf.

5. Penelitian yang dilakukan oleh (Christanti et al., 2018) dengan judul **Fast and Accurate Spelling Correction Using Trie and Damerau Levenshtein Bigram.** Penelitian ini melakukan pendeteksian dan koreksi kesalahan ejaan jenis *real word* dan *non word* dengan

menggunakan metode trie, bigram dan *damerau levenshtein distance*. Penelitian ini bertujuan untuk meningkatkan akurasi dan kecepatannya. Hasil yang diperoleh pada penelitian ini yaitu tingkat akurasi terbaik pada metode bigram dan *damerau levenshtein distance* dengan kategori kalimat sebesar 50%, kategori kata sebesar 84,62% dan waktu proses disetiap kalimat sebesar 18,89 ms. Selain itu, metode *damerau levenshtein distance* jika dibandingkan dengan *levenshtein distance* akan memberikan hasil yang lebih baik karena *damerau levenshtein* lebih mampu menjangkau beberapa kesalahan ejaan, hasil yang diperoleh n-gram jenis bigram lebih bagus dibandingkan dengan trigram, adanya metode trie pada penelitian ini meningkatkan kecepatan pemrosesan secara signifikan, dan penggunakan metode *smoothing* pada penelitian ini mampu untuk mereduksi kesebaran data yang tinggi.

6. Penelitian yang dilakukan oleh (Prasetyo et al., 2018) dengan judul **Fitur Autocorrect dan Spelling Suggestion Pada Penulisan Naskah Bahasa Indoenesia di BMS TV.** Penelitian ini bertujuan untuk membantu seorang *news director* di sebuah stasiun televisi dalam hal memeriksa dan memperbaiki kesalahan ejaan dalam bahasa Indonesia. Metode yang digunakan yaitu metode *jaro winkler*. Pada tahapan *preprocessing* juga dilengkapi dengan proses *stemming*. Pengujian menggunakan 60 kata dengan beberapa skenario penulisan kata yang sengaja disalahkan ejaannya. Dari 60 kata yang diuji, didapatkan 49 kata yang berhasil dideteksi secara tepat. Selain itu, jarak waktu yang dibutuhkan untuk memberikan rekomendasi kata yang benar dengan menggunakan fitur *autocorrect* lebih cepat dibandingkan tanpa menggunakan fitur *autocorrect*.

7. Penelitian yang dilakukan oleh (Tinaliah & Elizabeth, 2018) dengan judul **Perbandingan Hasil Deteksi Plagiarisme Dokumen dengan Metode Jaro Winkler Distance dan Metode Latent Semantic Analysis.** Penelitian ini melakukan perbandingan diantara 2 metode yaitu *jaro winkler* dan *Latent Semantic Analysis* (LSA) untuk pendeteksian plagiarisme. Metode *jaro winkler* tidak hanya dapat digunakan untuk pendeteksian kesalahan ejaan. Akan tetapi juga dapat digunakan dalam pendeteksian duplikat atau plagiarisme suatu dokumen. Data latih yang digunakan sejumlah 100 buah dan data uji sejumlah 5 buah yang dilakukan sebanyak 10 kali untuk analisis kinerja dari kedua metode yang digunakan. Dalam proses pengujian dilakukan 2 hal yang berbeda yaitu pengujian yang melalui proses *stemming* terlebih dahulu dan pengujian tanpa proses *stemming*. Dari perbandingan 2 pengujian tersebut dapat dikatakan bahwa metode *jaro winkler* yang melalui proses *stemming* terlebih dahulu dalam proses deteksi memiliki nilai yang lebih baik dibandingkan *jaro winkler* tanpa proses stemming. Selain itu, *jaro winkler* menghasilkan nilai plagiat sebesar 100% dan *Latent Semantic Analysis* (LSA) sebesar 97,14%.

8. Penelitian yang dilakukan oleh (Dewi & Qoiriah, 2020) dengan judul **Implementasi Algoritma Jaro Winkler Distance dan N-gram untuk Deteksi dan Prediksi Perbaikan Kesalahan Penulisan Kata Bahasa Indonesia pada Karya Tulis Ilmiah**. Penelitian ini membahas tentang deteksi dan koreksi kesalahan ejaan dengan pengkombinasian metode *jaro winkler* dengan n-gram. Objek yang digunakan pada penelitian ini yaitu karya tulis ilmiah mahasiswa sebanyak 14 data untuk proses pengujian. Selain, jumlah kata yang digunakan

juga berbeda-beda. Hasil yang diperoleh pada penelitian ini yaitu hasil terbaik sebesar 85,7% dan hasil terkecil sebesar 45%. Akan tetapi, penelitian ini memiliki kelemahan dimana masih adanya kata yang seharusnya tidak salah ejaan tetapi dianggap oleh sistem menjadi kata yang salah. Hal tersebut terjadi karena kurangnya kualitas korpus yang digunakan sehingga dapat dikatakan kualitas korpus penting dalam proses deteksi dan koreksi kesalahan ejaan pada penelitian ini.

**Tabel 2. 3 *State of The Art***

| Peneliti | Judul | Metode | Hasil |
|---|---|---|---|
| Pratip Samanta, Bidyut B. Chaudhuri (2013) | A Simple Real Word Error Detection and Correction Using Local Word Bigram and Trigram | N-gram | Menghasilkan presisi sebesar 71%-79% dan recall sebesar 81%-88%. |
| Muhammad Aburizal Siregar (2017) | Deteksi dan Koreksi Kesalahan Real Word Pada Penulisan Kata Menggunakan Metode N-Gram Dalam Teks Berbahasa Indonesia | N-gram | Menghasilkan akurasi yang sangat kecil sebesar 11%. |
| Yeny Rochmawati, Retno Kusumaningrum (2015) | Studi Perbandingan Algoritma Pencarian String Dalam Metode Approximate String Matching Untuk Identifikasi Kesalahan Pengetikan Teks | Hamming Distance, Levenshtein Distance, Damerau Levenshtein Distance, Jaro Winkler | *Jaro Winkler* memperoleh nilai Measure Average Precision (MAP) tertinggi dibandingkan metode lainnya sebesar 0,87 dengan pembagian beberapa tipe kesalahan seperti kesalahan penghapusan huruf sebesar 0,92, kesalahan penambahan huruf sebesar 0,90, kesalahan penggantian huruf sebesar 0,70, dan kesalahan penukaran huruf sebesar 0,95. |
| Wihardodo (2018) | Implementasi Perbaikan Kesalahan Ejaan Pada Sistem Essay Scoring | Jaro Winkler | *Jaro winkler* mampu 3 jenis kesalahan ejaan berupa kekurangan huruf, kelebihan huruf, dan salah penempatan/transposisi |

| | | | |
|---|---|---|---|
| | | | huruf. Selain itu, diperoleh tingkat akurasi sebesar 57%-73%. |
| Viny Christanti M , Rudy , Dali S. Naga (2018) | Fast and Accurate Spelling Correction Using Trie and Damerau-levenshtein Distance Bigram | Damerau Levenshtein Distance, N-gram, Trie | Menghasilkan tingkat akurasi terbaik pada metode bigram dan *damerau levenshtein distance* dengan kategori kalimat sebesar 50%, kategori kata sebesar 84,62% dan waktu proses disetiap kalimat sebesar 18,89 ms. Selain itu, metode *damerau levenshtein distance* jika dibandingkan dengan *levenshtein distance* akan memberikan hasil yang lebih baik karena *damerau levenshtein* lebih mampu menjangkau beberapa kesalahan ejaan, hasil yang diperoleh n-gram jenis bigram lebih bagus dibandingkan dengan trigram, adanya metode trie pada penelitian ini meningkatkan kecepatan pemrosesan secara signifikan, dan penggunakan metode *smoothing* pada penelitian ini mampu untuk mereduksi kesebaran data yang tinggi. |
| Agung Prasetyo, Wiga Maulana Baihaqi, Iqbaluddin Syam Had (2018) | Fitur Autocorrect dan Spelling Suggestion Pada Penulisan Naskah Bahasa Indoenesia di BMS TV | Jaro Winkler | Dari 60 kata kesalahan ejaan, sistem dapat mendeteksi 49 kata kesalahan ejaan. Selain itu, waktu yang dibutuhkan untuk pemberian rekomendasi kata dengan |

| | | | |
|---|---|---|---|
| | | | fitur *autocorrect* lebih cepat dibandingkan tanpa fitur *autocorrect*. |
| Tinaliah, Triana Elizabeth (2018) | Perbandingan Hasil Deteksi Plagiarisme Dokumen dengan Metode JaroWinkler Distance dan Metode Latent Semantic Analysis | Jaro Winkler, Latent Semantic Analysis | *Jaro winkler* menghasilkan nilai plagiat sebesar 100% dan *Latent Semantic Analysis* (LSA) sebesar 97,14%. Selain itu, *jaro winkler* dengan proses *stemming* lebih baik nilai deteksinya dibandingkan *jaro winkle*r tanpa proses *stemming*. |
| Nuzhul Citrasari Dewi, Anita Qoiriah (2020) | Implementasi Algoritma Jaro Winkler Distance dan N-gram untuk Deteksi dan Prediksi Perbaikan Kesalahan Penulisan Kata Bahasa Indonesia pada Karya Tulis Ilmiah | Jaro Winkler Distance, N-Gram | Hasil terbaik diperoleh sebesar 85,7% dan hasil terkecil sebesar 45%. |

Berdasarkan penelitian terdahulu, maka penelitian yang akan dilakukan memiliki karakteristik yang relatif sama dalam hal tema penelitian. Namun penelitian ini memiliki perbedaan dari sisi objek penelitian, sumber dan jumlah data, metode, dan hasil dari penelitian.

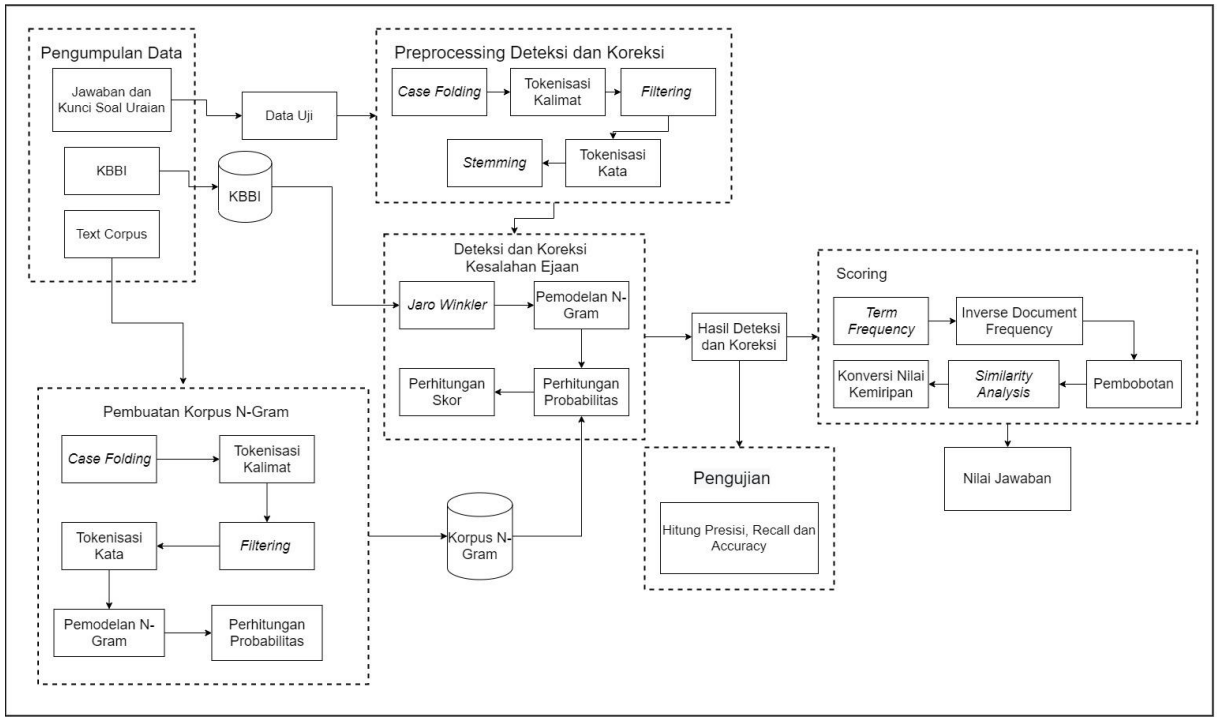
Berikut perbandingan dengan penelitian sebelumnya :

1. Pada penelitian (Samanta & Chaudhuri, 2013) dan (Siregar, 2017), proses pembuatan *confusion set* menggunakan metode *levenshtein distance*. Sedangkan pada penelitian ini menggunakan metode *jaro winkler* yang sebelumnya dilakukan proses *stemming* terlebih dahulu pada *preprocessing*.
2. Pada penelitian (Wihardodo, 2018) hanya menggunakan metode *jaro winkler* saja untuk mendeteksi kesalahan ejaan. Akan tetapi, pada penelitian ini akan mengkombinasikan metode n-gram dengan *jaro winkler*.
3. Pada penelitian (Dewi & Qoiriah, 2020), metode *smoothing* yang diterapkan pada n-gram hanya menggunakan metode *additive smoothing* saja. Sedangkan pada penelitian ini, metode smoothing yang diterapkan pada metode n-gram yaitu metode *additive smoothing* dan *jelinek mercer*.

# BAB III
# METODOLOGI PENELITIAN DAN PENGEMBANGAN SISTEM

## 3.1 Metodologi Penelitian

Metodologi penelitian yang digunakan pada penelitian ini yaitu metodologi penelitian kuantitatif dengan tahapan penelitian yang dapat dilihat pada gambar 3.1 sebagai berikut.




**Gambar 3. 1 Tahapan Penelitian**

### 3.1.1 Pengumpulan Data

Pengumpulan data merupakan salah satu tahapan di dalam penelitian ini yang digunakan untuk mencari informasi yang dibutuhkan. Informasi tersebut kemudian digunakan sebagai dasar atau pendukung di dalam penelitian ini. Pengumpulan data dilakukan dengan cara sebagai berikut.

1. Studi Literatur

   Studi literatur merupakan salah satu cara dalam pengumpulan data dengan mencari dan menghimpun informasi yang relevan dengan penelitian. Sumber-sumber yang dapat digunakan dalam studi literatur seperti buku, artikel, jurnal, dan sumber lain yang dapat dipercaya. Informasi yang dicari pada penelitian ini berupa metode atau teknik analisis yang didapatkan dari penelitian sebelumnya. Informasi tersebut dapat berupa kelebihan dan kekurangan pada metode yang pernah digunakan pada penelitian sebelumnya yang selanjutnya digunakan sebagai bahan pertimbangan dalam pemilihan metode pada penelitian ini. Selain itu, studi literatur juga digunakan sebagai pendukung dalam penyelesaian penelitian.

2. Kuesioner

   Pada penelitian ini dilakukan pengumpulan data primer dengan cara membagikan kuesioner kepada beberapa responden. Kuesioner ini berisi beberapa soal uraian yang kemudian para responden diminta untuk menjawab soal uraian tersebut dalam bentuk jawaban uraian. Kemudian jawaban kuesioner dari para responden akan dibuat skenario kesalahan ejaan yang selanjutnya akan digunakan sebagai data uji pada penelitian ini.

Kemudian, sumber data penelitian yang digunakan pada penelitian ini sebagai berikut.

1. Data Uji

   Data uji merupakan salah satu data yang digunakan dalam penelitian ini sebagai bahan uji untuk proses deteksi dan koreksi kesalahan ejaan yang diimplementasikan di dalam *essay scoring*. Data uji ini diambil dari hasil kuesioner yang telah dibagikan kepada responden berupa jawaban soal uraian dalam bentuk kalimat definisi yang kemudian diberikan skenario kesalahan ejaan.

2. Kamus Kata

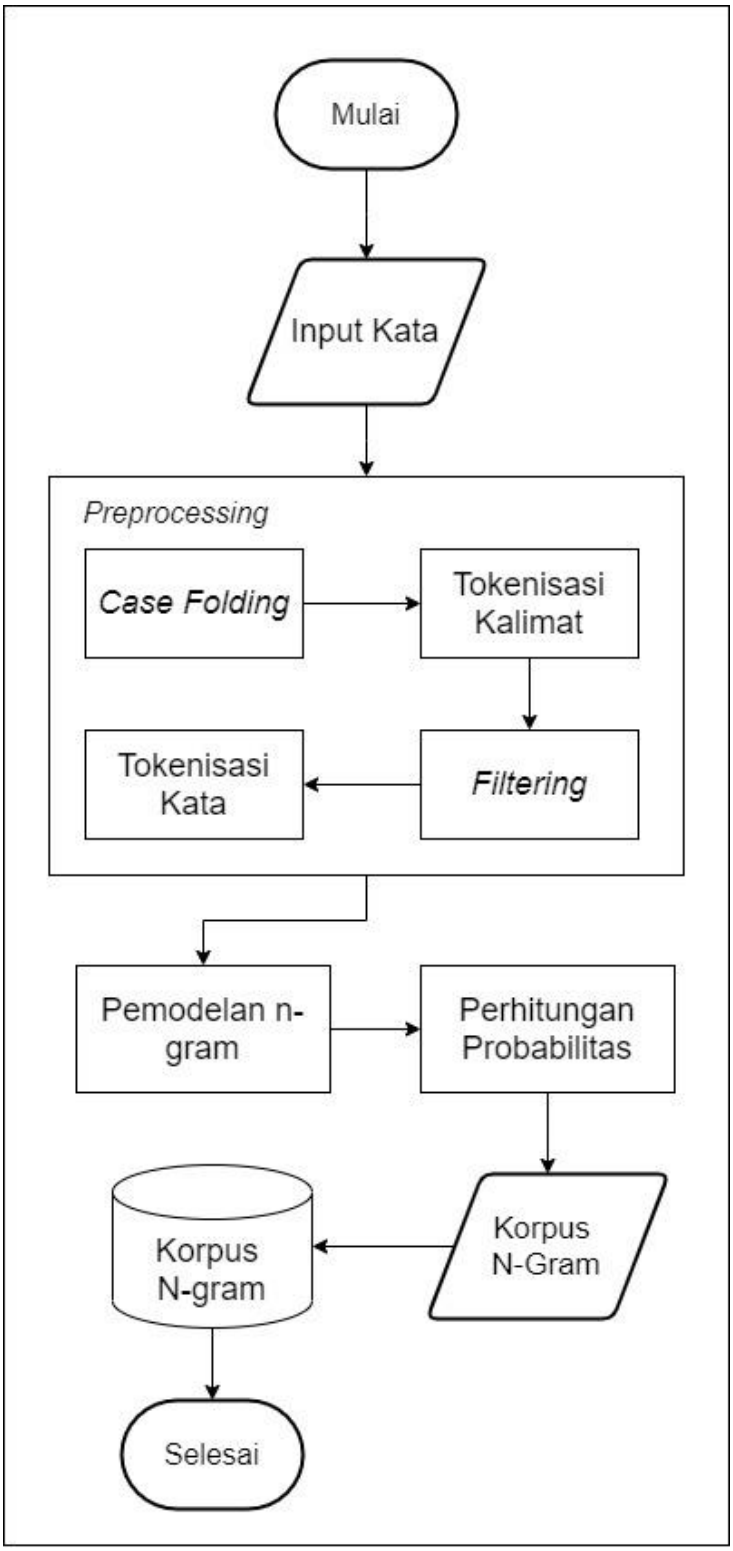
   Kamus kata berisi kumpulan kata yang nantinya akan digunakan sebagai acuan dalam proses deteksi dan koreksi kesalahan ejaan. Kamus kata ini diambil dari Kamus Besar Bahasa Indonesia (KBBI).

3. Korpus Kata

   Korpus kata berisi kumpulan kata yang digunakan sama seperti kamus kata yaitu sebagai acuan dalam proses deteksi dan koreksi kesalahan ejaan. Akan tetapi, korpus kata ini merupakan hasil dari kumpulan kata yang telah dilakukan pemodelan n-gram. Kumpulan kata yang digunakan pada korpus kata bersumber dari UI Tagged Corpus dan POS Tag Indonesia.

### 3.1.2 Pembuatan Korpus N-Gram (Korpus Kata)

Korpus n-gram merupakan daftar kumpulan kata yang dibentuk dengan model n-gram yang digunakan untuk perhitungan probabilitas n-gram data uji berdasarkan korpus n-gram yang telah dibuat. Sebelum dilakukan pemodelan dengan n-gram terlebih dahulu dilakukan *preprocessing*. Kemudian, ketika *preprocessing* telah dilalui, maka dilanjutkan dengan pembuatan model n-gram. Model n-gram pada penelitian ini dengan cara merangkai potongan token pada proses sebelumnya menjadi model unigram, bigram dan trigram. N-gram yang telah dibentuk dilanjutkan dengan perhitungan kemunculan n-gram di dalam teks. Hasil perhitungan tersebut kemudian ditambahkan ke dalam korpus. Alur pembuatan korpus n-gram dapat dilihat pada gambar 3.2.




**Gambar 3. 2 Alur Pembuatan Korpus N-gram**

Sumber kata yang digunakan untuk pembuatan korpus n-gram diambil dari UI Tagged Corpus dan POS Tag Indonesia. Cuplikan salah satu korpus dari UI Tagged Corpus dapat dilihat pada tabel 3.1.

**Tabel 3. 1 Cuplikan UI Tagged Corpus**

| |
|---|
| Biaya buruh tenaga kerja naik pada tingkat yang jauh lebih cepat dalam industri pelayanan kesehatan dibandingkan industri-industri lainnya. Sebagai contoh, gaji pekerja rumah sakit swasta meloncat hingga 6.6% dalam 12 bulan yang berakhir pada bulan September, dibanding dengan 4.4 % untuk pekerja di semua di industri. |

Pada gambar 3.2 hal pertama yang dilakukan di dalam proses pembuatan korpus n-gram yaitu *preprocessing* yang terdiri dari beberapa tahap sebagai berikut :

1. *Case Folding*

   Pada tahapan ini digunakan untuk mengubah semua huruf kapital menjadi huruf kecil seperti pada tabel 3.2.

**Tabel 3. 2 *Case Folding***

| | |
|---|---|
| Sebelum | **B**iaya buruh tenaga kerja naik pada tingkat yang jauh lebih cepat dalam industri pelayanan kesehatan dibandingkan industri-industri lainnya. **S**ebagai contoh, gaji pekerja rumah sakit swasta meloncat hingga 6.6% dalam 12 bulan yang berakhir pada bulan **S**eptember, dibanding dengan 4.4 % untuk pekerja di semua di industri. |
| Sesudah | **b**iaya buruh tenaga kerja naik pada tingkat yang jauh lebih cepat dalam industri pelayanan kesehatan dibandingkan industri-industri lainnya. **s**ebagai contoh, gaji pekerja rumah sakit swasta meloncat hingga 6.6% dalam 12 bulan yang berakhir pada bulan **s**eptember, dibanding dengan 4.4 % untuk pekerja di semua di industri. |

   Huruf tebal dan bergaris bawah merupakan huruf yang dilakukan proses *case folding*.

2. Tokenisasi Kalimat

   Setelah dilakukan proses *case folding*, maka perlu adanya tahapan dimana dilakukan tokenisasi kalimat. Tokenisasi kalimat ini digunakan untuk memisahkan kalimat yang terdapat di dalam paragraf yang dipisahkan oleh tanda titik. Hasil dari tahapan ini dapat dilihat pada tabel 3.3.

**Tabel 3. 3 Tokenisasi Kalimat**

| | |
|---|---|
| Sebelum | Biaya buruh tenaga kerja naik pada tingkat yang jauh lebih cepat dalam industri pelayanan kesehatan dibandingkan industri-industri lainnya. Sebagai contoh, gaji pekerja rumah sakit swasta meloncat hingga 6.6% dalam 12 bulan yang berakhir pada bulan September, dibanding dengan 4.4 % untuk pekerja di semua di industri. |
| Sesudah | biaya buruh tenaga kerja naik pada tingkat yang jauh lebih cepat dalam industri pelayanan kesehatan dibandingkan industri-industri lainnya. |
| | sebagai contoh, gaji pekerja rumah sakit swasta meloncat hingga 6.6% dalam 12 bulan yang berakhir pada bulan september, dibanding dengan 4.4 % untuk pekerja di semua di industri. |

3. *Filtering*

   *Filtering* merupakan salah satu tahapan yang digunakan untuk menghapus sesuatu yang dianggap tidak penting. *Filtering* pada tahapan ini, yang dilakukan penghapusan yaitu karakter selain angka, huruf, dan spasi. Hasil dari proses *filtering* dapat dilihat pada tabel 3.4.

**Tabel 3. 4 *Filtering***

| | |
|---|---|
| Sebelum | Biaya buruh tenaga kerja naik pada tingkat yang jauh lebih cepat dalam industri pelayanan kesehatan dibandingkan industri-industri lainnya. Sebagai contoh, gaji pekerja rumah sakit swasta meloncat hingga 6.6% dalam 12 bulan yang berakhir pada bulan September, dibanding dengan 4.4 % untuk pekerja di semua di industri. |
| Sesudah | biaya buruh tenaga kerja naik pada tingkat yang jauh lebih cepat dalam industri pelayanan kesehatan dibandingkan industriindustri lainnya |
| | sebagai contoh gaji pekerja rumah sakit swasta meloncat hingga 66 dalam 12 bulan yang berakhir pada bulan september dibanding dengan 44 untuk pekerja di semua di industri |

Dilihat dari tabel 3.4, kata “industri-industri” dilakukan proses *filtering* dengan menghapus tanda hubung sehingga dalam penulisannya menjadi digabung seperti “industriindustri”. Selain itu, terdapat juga penghapusan simbol % karena termasuk ke dalam karakter selain angka, huruf, dan spasi.

4. Tokenisasi Kata

   Tokenisasi kata merupakan tahapan yang serupa dengan tokenisasi kalimat. Perbedaannya, pada tahapan ini dilakukan pemotongan kalimat berdasarkan kata sehingga hasil yang didapatkan berupa kumpulan kata atau token. Hasil dari tokenisasi dapat dilihat pada tabel 3.5.

**Tabel 3. 5 Tokenisasi Kata**

| Kalimat 1 | Kalimat 2 |
|---|---|
| _ | _ |
| biaya | sebagai |
| buruh | contoh |
| tenaga | gaji |
| kerja | pekerja |
| naik | rumah |
| pada | sakit |
| tingkat | swasta |
| yang | meloncat |
| jauh | hingga |
| lebih | 66 |
| cepat | dalam |
| dalam | 12 |
| industri | bulan |
| pelayanan | yang |
| kesehatan | berakhir |
| dibandingkan | pada |
| industriindustri | bulan |
| lainnya | september |
| _ | dibanding |
| | dengan |
| | 44 |
| | untuk |
| | pekerja |
| | di |
| | semua |
| | industri |
| | _ |

Apabila *preprocessing* telah dilakukan, maka dapat dilanjutkan dengan pembuatan model n-gram. Jenis n-gram yang digunakan yaitu unigram, bigram dan trigram. Pembuatan model unigram hanya dari token kata asli saja. Kemudian, pembuatan model

bigram dengan cara menggabungkan sebuah token kata dengan token kata di depannya. Sedangkan trigram dengan cara menggabungkan token kata dengan 2 token kata di depannya. Proses pembuatan unigram dapat dilihat pada tabel 3.6.

**Tabel 3. 6 Pembuatan Unigram**

| Kalimat | Kata | Unigram |
|---|---|---|
| Pertama | _ | _ |
| | biaya | biaya |
| | buruh | buruh |
| | tenaga | tenaga |
| | kerja | kerja |
| | naik | naik |
| | pada | pada |
| | tingkat | tingkat |
| | yang | yang |
| | jauh | jauh |
| | lebih | lebih |
| | cepat | cepat |
| | dalam | dalam |
| | industri | industri |
| | pelayanan | pelayanan |
| | kesehatan | kesehatan |
| | dibandingkan | dibandingkan |
| | industriindustri | industriindustri |
| | lainnya | lainnya |
| | _ | _ |
| Kedua | _ | _ |
| | sebagai | sebagai |
| | contoh | contoh |
| | gaji | gaji |
| | pekerja | pekerja |
| | rumah | rumah |
| | sakit | sakit |
| | swasta | swasta |
| | meloncat | meloncat |
| | hingga | hingga |
| | 66 | 66 |
| | dalam | dalam |

| | | |
|---|---|---|
| | 12 | 12 |
| | bulan | bulan |
| | yang | yang |
| | berakhir | berakhir |
| | pada | pada |
| | bulan | bulan |
| | september | september |
| | dibanding | dibanding |
| | dengan | dengan |
| | 44 | 44 |
| | untuk | untuk |
| | pekerja | pekerja |
| | di | di |
| | semua | semua |
| | industri | industri |
| | _ | _ |

Proses pembuatan bigram dapat dilihat pada tabel 3.7.

**Tabel 3. 7 Pembuatan Bigram**

| Kalimat | Kata | Bigram |
|---|---|---|
| Pertama | _ | _biaya |
| | biaya | biaya buruh |
| | buruh | buruh tenaga |
| | tenaga | tenaga kerja |
| | kerja | kerja naik |
| | naik | naik pada |
| | pada | pada tingkat |
| | tingkat | tingkat yang |
| | yang | yang jauh |
| | jauh | jauh lebih |
| | lebih | lebih cepat |
| | cepat | cepat dalam |
| | dalam | dalam industri |
| | industri | industri pelayanan |
| | pelayanan | pelayanan kesehatan |
| | kesehatan | kesehatan dibandingkan |
| | dibandingkan | dibandingkan industriindustri |
| | industriindustri | industriindustri lainnya |

| | | |
|---|---|---|
| | lainnya | lainnya _ |
| | _ | |
| Kedua | _ | _sebagai |
| | sebagai | sebagai contoh |
| | contoh | contoh gaji |
| | gaji | gaji pekerja |
| | pekerja | pekerja rumah |
| | rumah | rumah sakit |
| | sakit | sakit swasta |
| | swasta | swasta meloncat |
| | meloncat | meloncat hingga |
| | hingga | hingga 66 |
| | 66 | 66 dalam |
| | dalam | dalam 12 |
| | 12 | 12 bulan |
| | bulan | bulan yang |
| | yang | yang berakhir |
| | berakhir | berakhir pada |
| | pada | pada bulan |
| | bulan | bulan September |
| | september | september dibanding |
| | dibanding | dibanding dengan |
| | dengan | dengan 44 |
| | 44 | 44 untuk |
| | untuk | untuk pekerja |
| | pekerja | pekerja di |
| | di | di semua |
| | semua | semua industri |
| | industri | industri _ |
| | _ | |

Kemudian, pembentukan trigram dapat dilihat pada tabel 3.8.

**Tabel 3. 8 Pembuatan Trigram**

| Kalimat | Kata | Trigram |
|---|---|---|
| Pertama | _ | _biaya buruh |
| | biaya | biaya buruh tenaga |
| | buruh | buruh tenaga kerja |
| | tenaga | tenaga kerja naik |

| | | |
|---|---|---|
| | kerja | kerja naik pada |
| | naik | naik pada tingkat |
| | pada | pada tingkat yang |
| | tingkat | tingkat yang jauh |
| | yang | yang jauh lebih |
| | jauh | jauh lebih cepat |
| | lebih | lebih cepat dalam |
| | cepat | cepat dalam industri |
| | dalam | dalam industri pelayanan |
| | industri | industri pelayanan kesehatan |
| | pelayanan | pelayanan kesehatan dibandingkan |
| | kesehatan | kesehatan dibandingkan industriindustri |
| | dibandingkan | dibandingkan industriindustri lainnya |
| | industriindustri | industriindustri lainnya _ |
| | lainnya | |
| | _ | |
| Kedua | _ | _ sebagai contoh |
| | sebagai | sebagai contoh gaji |
| | contoh | contoh gaji pekerja |
| | gaji | gaji pekerja rumah |
| | pekerja | pekerja rumah sakit |
| | rumah | rumah sakit swasta |
| | sakit | sakit swasta meloncat |
| | swasta | swasta meloncat hingga |
| | meloncat | meloncat hingga 66 |
| | hingga | hingga 66 dalam |
| | 66 | 66 dalam 12 |
| | dalam | dalam 12 bulan |
| | 12 | 12 bulan yang |
| | bulan | bulan yang berakhir |
| | yang | yang berakhir pada |
| | berakhir | berakhir pada bulan |
| | pada | pada bulan september |
| | bulan | bulan september dibanding |
| | september | september dibanding dengan |

| | | |
|---|---|---|
| | dibanding | dibanding dengan 44 |
| | dengan | dengan 44 untuk |
| | 44 | 44 untuk pekerja |
| | untuk | untuk pekerja di |
| | pekerja | pekerja di semua |
| | di | di semua industri |
| | semua | semua industri _ |
| | industri | |
| | _ | |

Dari kumpulan kata yang telah dibentuk unigram, bigram dan trigram, maka dilanjutkan dengan menghitung kemunculan n-gram di dalam teks sumber yang digunakan. Perhitungan kemunculan n-gram di dalam teks dapat dilihat pada tabel 3.9, tabel 3.10, dan tabel 3.11.

**Tabel 3. 9 Jumlah Kemunculan Unigram**

| Kata | Jumlah Kemunculan |
|---|---|
| _ | 4 |
| biaya | 1 |
| buruh | 1 |
| tenaga | 1 |
| kerja | 1 |
| naik | 1 |
| pada | 1 |
| tingkat | 1 |
| yang | 2 |
| jauh | 1 |
| lebih | 1 |
| cepat | 1 |
| dalam | 2 |
| industri | 2 |
| pelayanan | 1 |
| kesehatan | 1 |
| dibandingkan | 1 |
| industriindustri | 1 |
| lainnya | 1 |
| sebagai | 1 |
| contoh | 1 |
| gaji | 1 |

| | |
|---|---|
| pekerja | 2 |
| rumah | 1 |
| sakit | 1 |
| swasta | 1 |
| meloncat | 1 |
| hingga | 1 |
| 66 | 1 |
| 12 | 1 |
| bulan | 1 |
| berakhir | 1 |
| pada | 1 |
| bulan | 1 |
| september | 1 |
| dibanding | 1 |
| dengan | 1 |
| 44 | 1 |
| untuk | 1 |
| di | 1 |
| semua | 1 |

**Tabel 3. 10 Jumlah Kemunculan Bigram**

| Bigram | Jumlah Kemunculan |
|---|---|
| _biaya | 1 |
| biaya buruh | 1 |
| buruh tenaga | 1 |
| tenaga kerja | 1 |
| kerja naik | 1 |
| naik pada | 1 |
| pada tingkat | 1 |
| tingkat yang | 1 |
| yang jauh | 1 |
| jauh lebih | 1 |
| lebih cepat | 1 |
| cepat dalam | 1 |
| dalam industri | 1 |
| industri pelayanan | 1 |
| pelayanan kesehatan | 1 |
| kesehatan dibandingkan | 1 |

| | |
|---|---|
| dibandingkan industriindustri | 1 |
| industriindustri lainnya | 1 |
| lainnya _ | 1 |
| _sebagai | 1 |
| sebagai contoh | 1 |
| contoh gaji | 1 |
| gaji pekerja | 1 |
| pekerja rumah | 1 |
| rumah sakit | 1 |
| sakit swasta | 1 |
| swasta meloncat | 1 |
| meloncat hingga | 1 |
| hingga 66 | 1 |
| 66 dalam | 1 |
| dalam 12 | 1 |
| 12 bulan | 1 |
| bulan yang | 1 |
| yang berakhir | 1 |
| berakhir pada | 1 |
| pada bulan | 1 |
| bulan september | 1 |
| september dibanding | 1 |
| dibanding dengan | 1 |
| dengan 44 | 1 |
| 44 untuk | 1 |
| untuk pekerja | 1 |
| pekerja di | 1 |
| di semua | 1 |
| semua industri | 1 |
| industri _ | 1 |

**Tabel 3. 11 Jumlah Kemunculan Trigram**

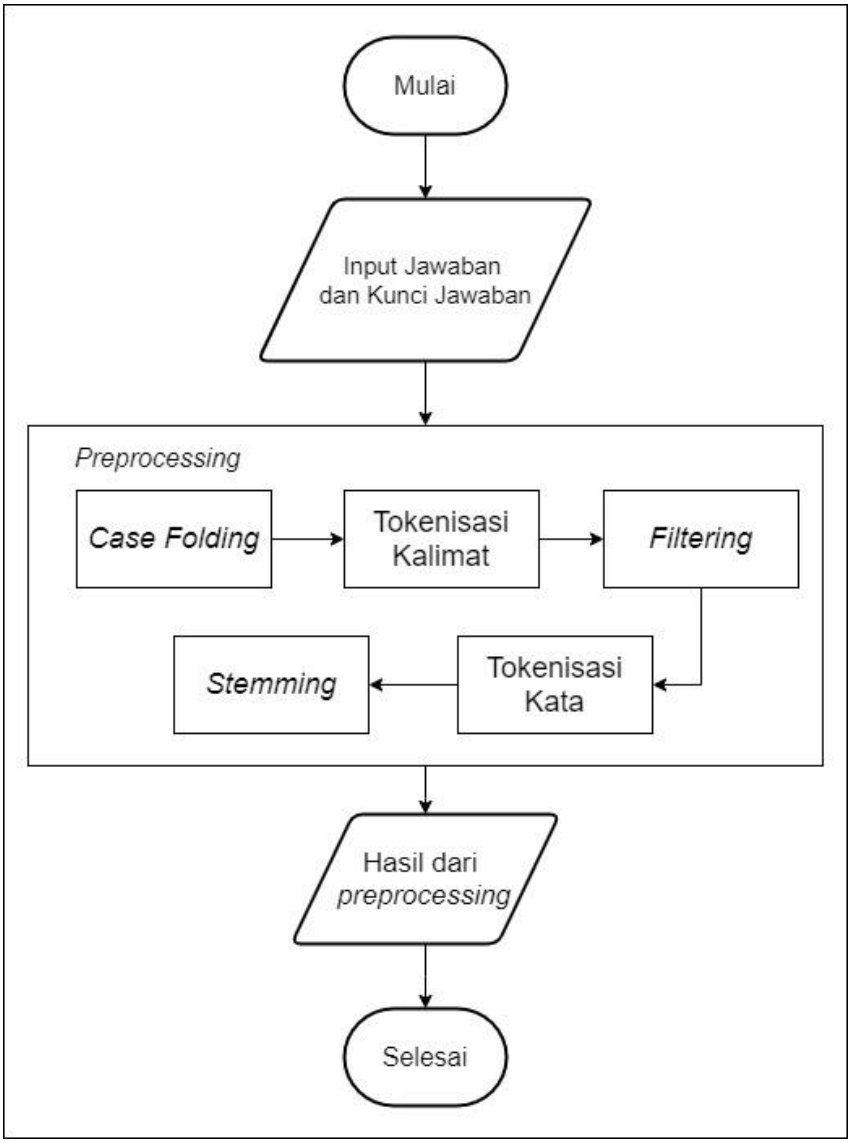
| Trigram | Jumlah Kemunculan |
|---|---|
| _biaya buruh | 1 |
| biaya buruh tenaga | 1 |
| buruh tenaga kerja | 1 |
| tenaga kerja naik | 1 |
| kerja naik pada | 1 |

| | |
|---|---|
| naik pada tingkat | 1 |
| pada tingkat yang | 1 |
| tingkat yang jauh | 1 |
| yang jauh lebih | 1 |
| jauh lebih cepat | 1 |
| lebih cepat dalam | 1 |
| cepat dalam industri | 1 |
| dalam industri pelayanan | 1 |
| industri pelayanan kesehatan | 1 |
| pelayanan kesehatan dibandingkan | 1 |
| kesehatan dibandingkan industriindustri | 1 |
| dibandingkan industriindustri lainnya | 1 |
| industriindustri lainnya _ | 1 |
| _ sebagai contoh | 1 |
| sebagai contoh gaji | 1 |
| contoh gaji pekerja | 1 |
| gaji pekerja rumah | 1 |
| pekerja rumah sakit | 1 |
| rumah sakit swasta | 1 |
| sakit swasta meloncat | 1 |
| swasta meloncat hingga | 1 |
| meloncat hingga 66 | 1 |
| hingga 66 dalam | 1 |
| 66 dalam 12 | 1 |
| dalam 12 bulan | 1 |
| 12 bulan yang | 1 |
| bulan yang berakhir | 1 |
| yang berakhir pada | 1 |
| berakhir pada bulan | 1 |
| pada bulan september | 1 |
| bulan september dibanding | 1 |
| september dibanding dengan | 1 |
| dibanding dengan 44 | 1 |

| | |
|---|---|
| dengan 44 untuk | 1 |
| 44 untuk pekerja | 1 |
| untuk pekerja di | 1 |
| pekerja di semua | 1 |
| di semua industri | 1 |
| semua industri _ | 1 |

### 3.1.3 *Preprocessing* Deteksi dan Koreksi

*Preprocessing* merupakan tahapan awal yang dilakukan sebelum data uji dilakukan proses deteksi dan koreksi kesalahan ejaan. Tahapan ini bertujuan agar data uji lebih siap dan lebih terstruktur untuk diproses deteksi dan koreksi kesalahan ejaan. Alur tahapan *preprocessing* deteksi dan koreksi dapat dilihat pada gambar 3.3.




**Gambar 3. 3 Alur Tahapan Preprocessing Deteksi dan Koreksi**

Gambar 3.3 merupakan alur tahapan yang harus dilalui ketika *preprocessing* deteksi dan koreksi. *Preprocessing* deteksi dan koreksi sebenarnya hampir sama seperti *preprocessing* pada pembuatan korpus n-gram. Akan tetapi, terdapat perbedaan yaitu pada *preprocessing* ini terdapat proses *stemming*. Tahapan *preprocessing* sebagai berikut.

1. *Case folding*

   *Case folding* digunakan untuk mengubah huruf kapital menjadi huruf kecil. Hasil yang didapatkan pada tahapan ini dapat dilihat pada tabel 3.12 yang ditandai dengan huruf tebal dan bergarisbawah.

**Tabel 3. 12 *Case Folding* Data Uji**

| | |
|---|---|
| Sebelum | **B**iaya buruh tenaga keraj naik. |
| Sesudah | **b**iaya buruh tenaga keraj naik. |

2. Tokenisasi kalimat

   Tokenisasi kalimat digunakan untuk memisahkan masing-masing kalimat di dalam paragraf yang dipisahkan dengan sebuah tanda titik. Proses tokenisasi kalimat pada data uji dapat dilihat pada tabel 3.13.

**Tabel 3. 13 Tokenisasi Kalimat Data Uji**

| | |
|---|---|
| Sebelum | biaya buruh tenaga keraj naik. |
| Sesudah | biaya buruh tenaga keraj naik. |

   Pada tabel 3.13 tidak terjadi tokenisasi kalimat, karena data uji yang digunakan hanya terdiri dari 1 kalimat saja.

3. *Filtering*

   *Filtering* pada penelitian ini yaitu menghapus karakter selain angka, huruf, dan spasi. Proses *filtering* pada data uji dapat dilihat pada tabel 3.14 dimana yang dihapus hanya tanda baca titik saja.

**Tabel 3. 14 *Filtering* Data Uji**

| | |
|---|---|
| Sebelum | biaya buruh tenaga keraj naik. |
| Sesudah | biaya buruh tenaga keraj naik |

4. Tokenisasi kata

   Tokenisasi kata digunakan untuk memisahkan sebuah kalimat menjadi ke dalam bentuk kumpulan kata atau token. Tokenisasi kata pada data uji dapat dilihat pada tabel 3.15.

**Tabel 3. 15 Tokenisasi Kata**

| | |
|---|---|
| Sebelum | biaya buruh tenaga keraj naik |
| Sesudah | _ |
| | biaya |

| | |
|---|---|
| | buruh |
| | tenaga |
| | keraj |
| | naik |
| | _ |

5. *Stemming*

   *Stemming* merupakan salah satu tahap yang digunakan untuk mengubah sebuah kata yang berimbuhan ke dalam bentuk kata dasar. *Stemming* pada data uji dapat dilihat pada tabel 3.16.

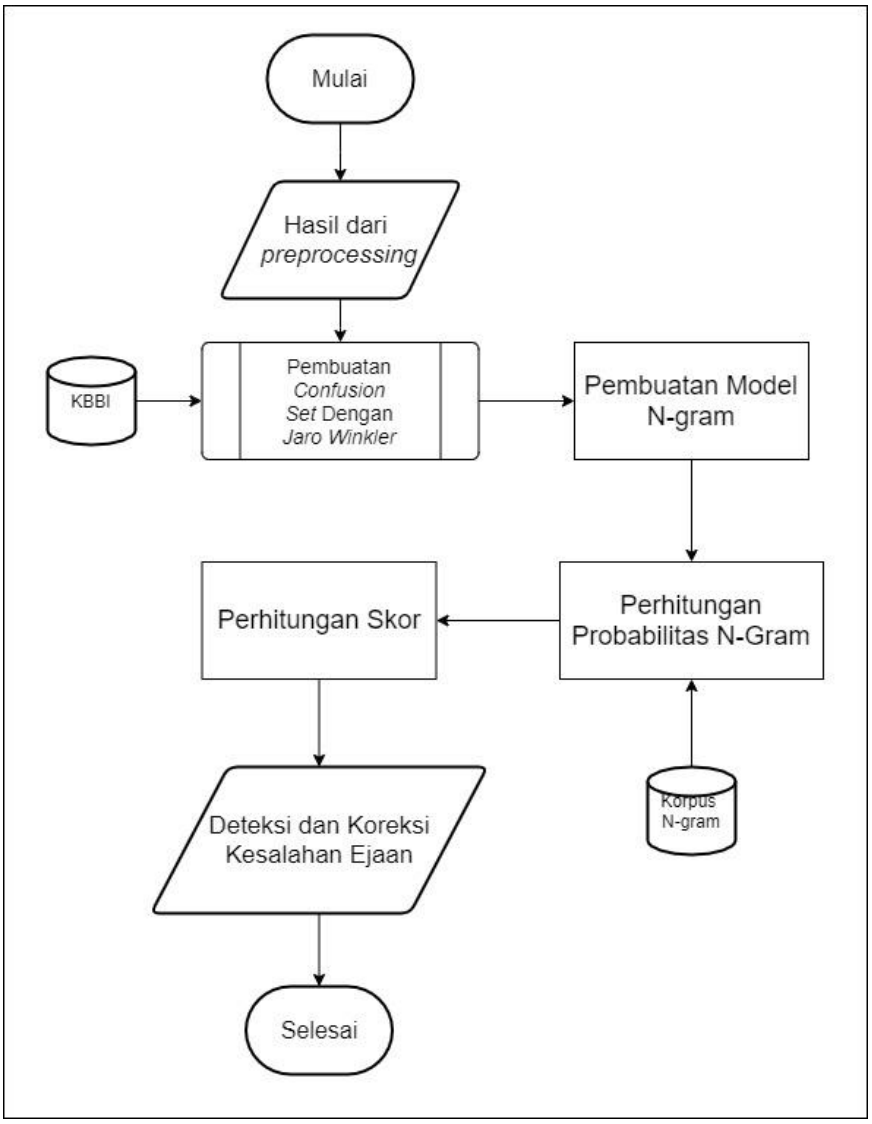
**Tabel 3. 16 *Stemming***

| | |
|---|---|
| Sebelum | biaya |
| | buruh |
| | tenaga |
| | keraj |
| | naik |
| Sesudah | _ |
| | biaya |
| | buruh |
| | tenaga |
| | keraj |
| | naik |
| | _ |

Dilihat dari tabel 3.16, maka dapat disimpulkan tidak terjadi perubahan sebelum dengan sesudah proses *stemming* karena kata yang digunakan sudah dalam bentuk kata dasar.

### 3.1.4 Deteksi dan Koreksi Kesalahan Ejaan

Deteksi dan koreksi kesalahan ejaan pada penelitian ini menggunakan metode n-gram dan *jaro winkler*. Tahap pertama yang dilakukan yaitu hasil dari *preprocessing* deteksi dan koreksi pada tahapan sebelumnya kemudian dilanjutkan dengan proses pembuatan *confusion set* dengan menghitung kedekatan 2 buah string dengan metode *jaro winkler* yang berdasarkan Kamus Besar Bahasa Indonesia (KBBI). Kemudian, hasil dari *confusion set* dilakukan proses pembuatan model n-gram. Model n-gram yang digunakan sama halnya seperti pembuatan korpus n-gram yaitu menggunakan unigram, bigram dan trigram. Hasil dari pembuatan model n-gram selanjutnya dilakukan perhitungan kemunculan setiap kata pada model n-gram hasil *confusion set* di dalam korpus n-gram. Setelah perhitungan kemunculan telah didapatkan, dapat dilanjutkan dengan perhitungan skor dimana dari hasil perhitungan ini dapat dideteksi kata yang salah beserta rekomendasi kata. Alur tahapan deteksi dan koreksi kesalahan ejaan penelitian ini dapat dilihat pada gambar 3.4.

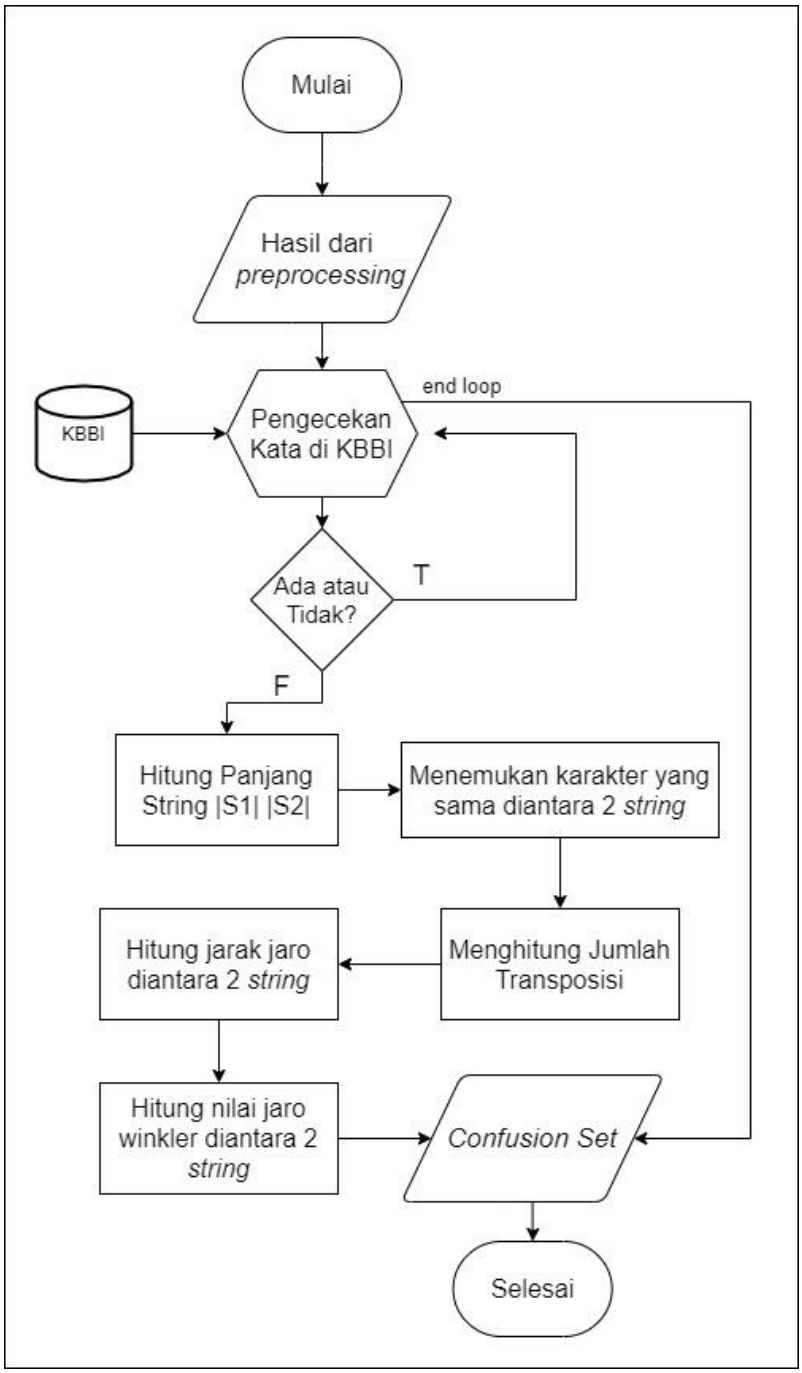



**Gambar 3. 4 Alur Tahapan Deteksi dan Koreksi Kesalahan Ejaan**

Berdasarkan gambar 3.4, terdapat beberapa tahapan yang perlu dilalui ketika proses deteksi dan koreksi kesalahan ejaan sebagai berikut.

1. Pembuatan *confusion set* dengan *jaro winkler*

   Pembuatan *confusion set* ini berlaku untuk setiap token kata yang sudah dilakukan *preprocessing* deteksi dan koreksi yang telah dilakukan pada tahap sebelumnya. Pembuatan *confusion set* ini menggunakan metode *jaro winkler* yang berdasarkan dengan Kamus Besar Bahasa Indonesia (KBBI) seperti alur yang dapat dilihat pada gambar 3.5.




**Gambar 3. 5 Alur Pembuatan *Confusion Set* Dengan *Jaro Winkler***

Sebagai contoh perhitungan *jaro winkler* diambilah token kata "keraj" yang sudah diberikan skenario kesalahan ejaan. Kata "keraj" akan dibandingkan dengan beberapa kata yang diambil dari Kamus Besar Bahasa Indonesia (KBBI) seperti "a", "ab", "kerja", dan "kerai". Perhitungan sebagai berikut.

**Tabel 3. 17 Penentuan Transposisi dan Kesalahan Ejaan**

| | | 0 |
|---|---|---|
| | | a |
| 0 | k | |
| 1 | e | |
| 2 | r | |
| 3 | a | ✓ |
| 4 | j | |

Tabel 3.17 merupakan gambaran dalam penentuan transposisi dan kesalahan ejaan antara kata "keraj" dengan "a" ditandai dengan ada atau tidaknya tanda centang. Tanda centang menandakan adanya kesamaan karakter dan transposisi karakter diantara 2 buah string yang dibandingkan. Berdasarkan tabel tersebut, didapatkan nilai sebagai berikut.

$|S_1|$ = panjang *string* pertama (a) = 1

$|S_2|$ = panjang *string* kedua (keraj) = 5

$m$ = karakter yang sama = 1

$t$ = transposisi = 0

$l$ = *prefix length* = 0

Perhitungan nilai *jaro distance* dengan menggunakan persamaan 2.13 sebagai berikut.

$$d_j = \frac{1}{3}\left(\frac{1}{1} + \frac{1}{5} + \frac{1-0}{1}\right) = \frac{1}{3}\left(\frac{11}{5}\right) = \frac{11}{15} = 0{,}73$$

Perhitungan nilai jaro winkler dengan menggunakan persamaan 2.15 sebagai berikut.

$$d_w = 0{,}73 + \big(0\,.\,0{,}1\,(1 - 0{,}73)\big) = 0{,}73 + 0 = 0{,}73$$

Dari perhitungan diantara *string* "a" dan "keraj" didapatkan nilai *jaro winkler* sebesar 0,73.

**Tabel 3. 18 Penentuan Transposisi dan Kesalahan Ejaan**

| | | 0 | 1 |
|---|---|---|---|
| | | a | b |
| 0 | k | | |
| 1 | e | | |
| 2 | r | | |
| 3 | a | ✓ | |
| 4 | j | | |

Perbandingan diantara dua kata "ab" dengan "keraj" dapat dilihat pada tabel 3.18. Berdasarkan tabel tersebut, didapatkan nilai sebagai berikut.

$|S_1|$ = panjang string pertama (ab) = 2

$|S_2|$ = panjang string kedua (keraj) = 5

$m$ = karakter yang sama = 1

$t$ = transposisi = 0

$l$ = *prefix length* = 0

Perhitungan nilai *jaro distance* dengan menggunakan persamaan 2.13 sebagai berikut.

$$d_j = \frac{1}{3}\left(\frac{1}{2} + \frac{1}{5} + \frac{1-0}{1}\right) = \frac{1}{3}\left(\frac{17}{10}\right) = \frac{17}{30} = 0{,}57$$

Perhitungan nilai jaro winkler dengan menggunakan persamaan 2.15 sebagai berikut.

$$d_w = 0{,}57 + \left(0\,.\,0{,}1\,(1 - 0{,}57)\right) = 0{,}57 + 0 = 0{,}57$$

Dari perhitungan diantara *string* "ab" dan "keraj" didapatkan nilai *jaro winkler* sebesar 0,57.

**Tabel 3. 19 Penentuan Transposisi dan Kesalahan Ejaan**

| | | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|
| | | k | e | r | j | a |
| 0 | k | ✓ | | | | |
| 1 | e | | ✓ | | | |
| 2 | r | | | ✓ | | |
| 3 | a | | | | | ✓ |
| 4 | j | | | | ✓ | |

Perbandingan diantara dua kata "kerja" dengan "keraj" dapat dilihat pada tabel 3.19. Berdasarkan tabel tersebut, didapatkan nilai sebagai berikut.

$|S_1|$ = panjang string pertama (kerja) = 5

$|S_2|$ = panjang string kedua (keraj) = 5

$m$ = karakter yang sama = 5

$t$ = transposisi = 1

$l$ = *prefix length* = 3

Perhitungan nilai *jaro distance* dengan menggunakan persamaan 2.13 sebagai berikut.

$$d_j = \frac{1}{3}\left(\frac{5}{5} + \frac{5}{5} + \frac{5-1}{5}\right) = \frac{1}{3}\left(\frac{14}{5}\right) = \frac{14}{15} = 0{,}93$$

Perhitungan nilai jaro winkler dengan menggunakan persamaan 2.15 sebagai berikut.

$$d_w = 0{,}93 + \left(3\,.\,0{,}1\,(1 - 0{,}93)\right) = 0{,}93 + 0{,}021 = 0{,}951$$

Dari perhitungan diantara *string* "kerja" dan "keraj" didapatkan nilai *jaro winkler* sebesar 0,951

**Tabel 3. 20 Penentuan Transposisi dan Kesalahan Ejaan**

| | | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|
| | | k | e | r | a | i |
| 0 | k | ✓ | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | e | | ✓ | | | |
| 2 | r | | | ✓ | | |
| 3 | a | | | | ✓ | |
| 4 | j | | | | | |

Perbandingan diantara dua kata “kerai” dengan “keraj” dapat dilihat pada tabel 3.20. Berdasarkan tabel tersebut, didapatkan nilai sebagai berikut.

$|S_1|$ = panjang string pertama (kerai) = 5

$|S_2|$ = panjang string kedua (keraj) = 5

$m$ = karakter yang sama = 4

$t$ = transposisi = 0

$l$ = *prefix length* = 4

Perhitungan nilai *jaro distance* dengan menggunakan persamaan 2.13 sebagai berikut.

$$d_j = \frac{1}{3}\left(\frac{4}{5} + \frac{4}{5} + \frac{4-1}{4}\right) = \frac{1}{3}\left(\frac{47}{20}\right) = \frac{47}{60} = 0{,}78$$

Perhitungan nilai jaro winkler dengan menggunakan persamaan 2.15 sebagai berikut.

$$d_w = 0{,}78 + \left(4\,.\,0{,}1\,(1 - 0{,}78)\right) = 0{,}78 + 0{,}088 = 0{,}868$$

Dari perhitungan diantara *string* “kerai” dan “keraj” didapatkan nilai *jaro winkler* sebesar 0,868.

Kemudian, hasil confusion set disetiap kata di dalam data uji dapat dilihat pada tabel 3.21.

**Tabel 3. 21 Contoh *Confusion Set* Data Uji**

| $W^i$ | Kata | $C_j^i$ | Angota | Nilai *Jaro Winkler* |
|---|---|---|---|---|
| $W^1$ | biaya | $C_1^1$ | biaya | 1 |
| | | $C_2^1$ | biasa | 0,9069 |
| $W^2$ | buruh | $C_1^2$ | buruh | 1 |
| | | $C_2^2$ | buruk | 0,9069 |
| $W^3$ | tenaga | $C_1^3$ | tenaga | 1 |
| | | $C_2^3$ | tenahak | 0,9094 |
| $W^4$ | keraj | $C_1^4$ | kerja | 0,951 |
| | | $C_2^4$ | kerai | 0,868 |
| $W^5$ | naik | $C_1^5$ | naik | 1 |
| | | $C_2^5$ | naif | 0,881 |

2. Pembuatan model n-gram

   Pembuatan model n-gram dilakukan setelah pembuatan *confusion set* dengan *jaro winkler*. Setiap kata yang ada di dalam *confusion set* dibuatkan model n-gram dengan

jenis bigram kiri, bigram kanan, dan trigram. Hasil pembuatan model n-gram pada confusion set dapat dilihat pada tabel 3.22.

**Tabel 3. 22 Contoh Pembuatan Model N-Gram Data Uji**

| $C_j^i$ | | $W^{i-1}$ | | $W^{i+1}$ | | $W^{i-1}C_j^i$ (bigram kiri) | $C_j^i W^{i+1}$ (bigram kanan) | $W^{i-1}C_j^i W^{i+1}$ (trigram) |
|---|---|---|---|---|---|---|---|---|
| $C_1^0$ | _ | | | | | | | |
| $C_1^1$ | biaya | $W^0$ | _ | $W^2$ | buruh | _ biaya | biaya buruh | _ biaya buruh |
| $C_2^1$ | biasa | | | | | _ biasa | biasa buruh | _ biasa buruh |
| $C_1^2$ | buruh | $W^1$ | biaya | $W^3$ | tenaga | biaya buruh | buruh tenaga | biaya buruh tenaga |
| $C_2^2$ | buruk | | | | | biaya buruk | buruk tenaga | biaya buruk tenaga |
| $C_1^3$ | tenaga | $W^2$ | buruh | $W^4$ | keraj | buruh tenaga | tenaga keraj | buruh tenaga keraj |
| $C_2^3$ | tenahak | | | | | buruh tenahak | tenahak keraj | buruh tenahak keraj |
| $C_1^4$ | kerja | $W^3$ | tenaga | $W^5$ | naik | tenaga kerja | kerja naik | tenaga kerja naik |
| $C_2^4$ | kerai | | | | | tenaga kerai | kerai naik | tenaga kerai naik |
| $C_1^5$ | naik | $W^4$ | keraj | $W^6$ | _ | keraj naik | naik _ | keraj naik _ |
| $C_2^5$ | naif | | | | | keraj naif | naif _ | keraj naif _ |
| $C_1^6$ | _ | | | | | | | |

Setelah pembuatan model n-gram, maka dilanjutkan dengan menghitung kemunculan tiap unigram, bigram dan trigram disetiap masing-masing kata yang terdapat di dalam *confusion set* berdasarkan dengan tabel 3.9, tabel 3.10 dan tabel 3.11. Apabila sebuah n-gram tidak ditemukan di dalam korpus kata, maka nilai akan 0. Perhitungan jumlah kemunculan unigram dapat dilihat pada tabel 3.23, bigram kiri pada tabel 3.24, bigram kanan pada tabel 3.25, dan trigram pada tabel 3.26.

**Tabel 3. 23 Contoh Jumlah Kemunculan Unigram Data Uji**

| $C_j^0$ | | $count(C_j^0)$ |
|---|---|---|
| $C_1^0$ | _ | 1 |
| $C_j^1$ | | $count(C_j^1)$ |
| $C_1^1$ | biaya | 1 |
| $C_2^1$ | biasa | 0 |
| $C_j^2$ | | $count(C_j^2)$ |

| | | |
|---|---|---|
| $C_1^2$ | buruh | 1 |
| $C_2^2$ | buruk | 0 |
| $C_j^3$ | | $count(C_j^3)$ |
| $C_1^3$ | tenaga | 1 |
| $C_2^3$ | tenahak | 0 |
| $C_j^4$ | | $count(C_j^4)$ |
| $C_1^4$ | kerja | 1 |
| $C_2^4$ | kerai | 0 |
| $C_j^5$ | | $count(C_j^5)$ |
| $C_1^5$ | naik | 1 |
| $C_2^5$ | naif | 0 |
| $C_j^6$ | | $count(C_j^6)$ |
| $C_1^6$ | _ | 1 |

**Tabel 3. 24 Contoh Jumlah Kemunculan Bigram Kiri Data Uji**

| $C_j^i$ | | $W^0C_j^1$ | $count(W^0C_j^1)$ |
|---|---|---|---|
| $C_1^1$ | biaya | _ biaya | 1 |
| $C_2^1$ | biasa | _ biasa | 0 |
| $C_j^i$ | | $W^1C_j^2$ | $count(W^1C_j^2)$ |
| $C_1^2$ | buruh | biaya buruh | 1 |
| $C_2^2$ | buruk | biaya buruk | 0 |
| $C_j^i$ | | $W^2C_j^3$ | $count(W^2C_j^3)$ |
| $C_1^3$ | tenaga | buruh tenaga | 1 |
| $C_2^3$ | tenahak | buruh tenahak | 0 |
| $C_j^i$ | | $W^3C_j^4$ | $count(W^3C_j^4)$ |
| $C_1^4$ | kerja | tenaga kerja | 1 |
| $C_2^4$ | kerai | tenaga kerai | 0 |
| $C_j^i$ | | $W^4C_j^5$ | $count(W^4C_j^5)$ |
| $C_1^5$ | naik | keraj naik | 0 |
| $C_2^5$ | naif | keraj naif | 0 |

**Tabel 3. 25 Contoh Jumlah Kemunculan Bigram Kanan Data Uji**

| $C_j^i$ | | $C_j^1W^2$ | $count(C_j^1W^2)$ |
|---|---|---|---|
| $C_1^1$ | biaya | biaya buruh | 1 |
| $C_2^1$ | biasa | biasa buruh | 0 |
| $C_j^i$ | | $C_j^2W^3$ | $count(C_j^2W^3)$ |

| | | | |
|---|---|---|---|
| $C_1^2$ | buruh | buruh tenaga | 1 |
| $C_2^2$ | buruk | buruk tenaga | 0 |
| $C_j^i$ | | $C_j^3 W^4$ | $count(C_j^3 W^4)$ |
| $C_1^3$ | tenaga | tenaga keraj | 0 |
| $C_2^3$ | tenahak | tenahak keraj | 0 |
| $C_j^i$ | | $C_j^4 W^5$ | $count(C_j^4 W^5)$ |
| $C_1^4$ | kerja | kerja naik | 1 |
| $C_2^4$ | kerai | kerai naik | 0 |
| $C_j^i$ | | $C_j^5 W^6$ | $count(C_j^5 W^6)$ |
| $C_1^5$ | naik | naik _ | 0 |
| $C_2^5$ | naif | naif _ | 0 |

**Tabel 3. 26 Contoh Jumlah Kemunculan Trigram Data Uji**

| | | | |
|---|---|---|---|
| $C_j^i$ | | $W^0 C_j^1 W^2$ | $count(W^0 C_j^1 W^2)$ |
| $C_1^1$ | biaya | _ biaya buruh | 1 |
| $C_2^1$ | biasa | _ biasa buruh | 0 |
| $C_j^i$ | | $W^1 C_j^2 W^3$ | $count(W^1 C_j^2 W^3)$ |
| $C_1^2$ | buruh | biaya buruh tenaga | 1 |
| $C_2^2$ | buruk | biaya buruk tenaga | 0 |
| $C_j^i$ | | $W^2 C_j^3 W^4$ | $count(W^2 C_j^3 W^4)$ |
| $C_1^3$ | tenaga | buruh tenaga keraj | 0 |
| $C_2^3$ | tenahak | buruh tenahak keraj | 0 |
| $C_j^i$ | | $W^3 C_j^4 W^5$ | $count(W^3 C_j^4 W^5)$ |
| $C_1^4$ | kerja | tenaga kerja naik | 1 |
| $C_2^4$ | kerai | tenaga kerai naik | 0 |
| $C_j^i$ | | $W^4 C_j^5 W^6$ | $count(W^4 C_j^5 W^6)$ |
| $C_1^5$ | naik | keraj naik _ | 0 |
| $C_2^5$ | naif | keraj naif _ | 0 |

Kemudian, dilanjutkan dengan perhitungan total jumlah kemunculan n-gram disetiap jenis n-gram pada *confusion set* yang telah dibuat. Perhitungan total jumlah kemunculan n-gram dapat dilihat pada tabel 3.27.

**Tabel 3. 27 Contoh Total Jumlah Kemunculan N-Gram Data Uji**

| $W^i$ | Token | $\sum_{r=1}^{k_i} count(C_r^i)$ (unigram) | $\sum_{r=1}^{k_i} count(W^{i-1} C_r^i)$ (bigram kiri) | $\sum_{r=1}^{k_i} count(C_r^i W^{i+1})$ (bigram kanan) | $\sum_{r=1}^{k_i} count(W^{i-1} C_r^i W^{i+1})$ (trigram) |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| $W^1$ | biaya | $count(C_1^1) + count(C_2^1) = 1+0=1$ | $count(W^0C_1^1) + count(W^0C_2^1) = 1+0=1$ | $count(C_1^1W^2) + count(C_2^1W^2) = 1+0=1$ | $count(W^0C_1^1W^2) + count(W^0C_2^1W^2) = 1+0=1$ |
| $W^2$ | buruh | $count(C_1^2) + count(C_2^2) = 1+0=1$ | $count(W^1C_1^2) + count(W^1C_2^2) = 1+0=1$ | $count(C_1^2W^3) + count(C_2^2W^3) = 1+0=1$ | $count(W^1C_1^2W^3) + count(W^1C_2^2W^3) = 1+0=1$ |
| $W^3$ | tenaga | $count(C_1^3) + count(C_2^3) = 1+0=1$ | $count(W^2C_1^3) + count(W^2C_2^3) = 1+0=1$ | $count(C_1^3W^4) + count(C_2^3W^4) = 0+0=0$ | $count(W^2C_1^3W^4) + count(W^2C_2^3W^4) = 0+0=0$ |
| $W^4$ | keraj | $count(C_1^4) + count(C_2^4) = 1+0=1$ | $count(W^3C_1^4) + count(W^3C_2^4) = 1+0=1$ | $count(C_1^4W^5) + count(C_2^4W^5) = 1+0=1$ | $count(W^3C_1^4W^5) + count(W^3C_2^4W^5) = 1+0=1$ |
| $W^5$ | naik | $count(C_1^5) + count(C_2^5) = 1+0=1$ | $count(W^4C_1^5) + count(W^4C_2^5) = 0+0=0$ | $count(C_1^5W^6) + count(C_2^5W^6) = 0+0=0$ | $count(W^4C_1^5W^6) + count(W^4C_2^5W^6) = 0+0=0$ |

3. Perhitungan probabilitas n-gram

   Perhitungan probabilitas n-gram ini dilakukan setelah proses pembuatan c*onfusion set* dengan menggunakan metode *jaro winkler*. Perhitungan probabilitas dilakukan disetiap anggota yang ada di dalam *confusion set*. Perhitungan probabilitas ini menggunakan persamaan 2.5, 2.6, 2.7 dan 2.8. Persamaan 2.5 digunakan untuk perhitungan probabilitas unigram, 2.6 perhitungan probabilitas bigram kiri, 2.7 perhitungan probabilitas bigram kanan dan 2.8 perhitungan probabilitas trigram. Perhitungan probabilitas pada tabel 3.28 sebagai berikut.

**Tabel 3. 28 Contoh Perhitungan Probabilitas**

| Anggota ($C_j^i$) | | Unigram, bigram, dan trigram | Probabilitas |
|---|---|---|---|
| $C_1^1$ | biaya | biaya | $P_0(C_j^i) = \frac{count\,(C_j^i) + \delta}{\sum_{r=0}^{k_i} count\,(C_r^i) + \delta * V_j}$ $= \frac{1+0{,}01}{1+0{,}01*2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |
| | | _ biaya | $P_1(W_j^i \mid W^{i-1}) = \frac{count\,(W^{i-1}C_j^i) + \delta}{\sum_{r=0}^{k_i} count\,(W^{i-1}C_r^i) + \delta * V_j}$ $= \frac{1+0{,}01}{1+0{,}01*2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |

| | | | |
|---|---|---|---|
| | | biaya buruh | $P_2(W_j^i \mid W^{i+1}) = \frac{count\left(W^{i+1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i+1}C_r^i\right) + \delta * V_j}$ $= \frac{1 + 0{,}01}{1 + 0{,}01 * 2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |
| | | _ biaya buruh | $P_3(W_j^i \mid W^{i-1}, W^{i+1})$ $= \frac{count\left(W^{i-1}C_j^iW^{i+1}\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^iW^{i+1}\right) + \delta * V_j}$ $= \frac{1 + 0{,}01}{1 + 0{,}01 * 2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |
| $C_2^1$ | biasa | biasa | $P_0(C_j^i) = \frac{count\left(C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(C_r^i\right) + \delta * V_j}$ $= \frac{0 + 0{,}01}{1 + 0{,}01 * 2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| | | _ biasa | $P_1(W_j^i \mid W^{i-1}) = \frac{count\left(W^{i-1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^i\right) + \delta * V_j}$ $= \frac{0 + 0{,}01}{1 + 0{,}01 * 2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| | | biasa buruh | $P_2(W_j^i \mid W^{i+1}) = \frac{count\left(W^{i+1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i+1}C_r^i\right) + \delta * V_j}$ $= \frac{0 + 0{,}01}{1 + 0{,}01 * 2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| | | _ biasa buruh | $P_3(W_j^i \mid W^{i-1}, W^{i+1})$ $= \frac{count\left(W^{i-1}C_j^iW^{i+1}\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^iW^{i+1}\right) + \delta * V_j}$ $= \frac{0 + 0{,}01}{1 + 0{,}01 * 2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| $C_1^2$ | buruh | buruh | $P_0(C_j^i) = \frac{count\left(C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(C_r^i\right) + \delta * V_j}$ $= \frac{1 + 0{,}01}{1 + 0{,}01 * 2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |
| | | biaya buruh | $P_1(W_j^i \mid W^{i-1}) = \frac{count\left(W^{i-1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^i\right) + \delta * V_j}$ $= \frac{1 + 0{,}01}{1 + 0{,}01 * 2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |

| | | | |
|---|---|---|---|
| | | buruh tenaga | $P_2(W_j^i \mid W^{i+1}) = \frac{count\left(W^{i+1}C_j^i\right)+\delta}{\sum_{r=0}^{k_i} count\left(W^{i+1}C_r^i\right)+\delta * V_j}$ $= \frac{1+0{,}01}{1+0{,}01*2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |
| | | biaya buruh tenaga | $P_3(W_j^i \mid W^{i-1}, W^{i+1})$ $= \frac{count\left(W^{i-1}C_j^iW^{i+1}\right)+\delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^iW^{i+1}\right)+\delta * V_j}$ $= \frac{1+0{,}01}{1+0{,}01*2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |
| $C_2^2$ | buruk | buruk | $P_0(C_j^i) = \frac{count\left(C_j^i\right)+\delta}{\sum_{r=0}^{k_i} count\left(C_r^i\right)+\delta * V_j}$ $= \frac{0+0{,}01}{1+0{,}01*2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| | | biaya buruk | $P_1(W_j^i \mid W^{i-1}) = \frac{count\left(W^{i-1}C_j^i\right)+\delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^i\right)+\delta * V_j}$ $= \frac{0+0{,}01}{1+0{,}01*2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| | | buruk tenaga | $P_2(W_j^i \mid W^{i+1}) = \frac{count\left(W^{i+1}C_j^i\right)+\delta}{\sum_{r=0}^{k_i} count\left(W^{i+1}C_r^i\right)+\delta * V_j}$ $= \frac{0+0{,}01}{1+0{,}01*2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| | | biaya buruk tenaga | $P_3(W_j^i \mid W^{i-1}, W^{i+1})$ $= \frac{count\left(W^{i-1}C_j^iW^{i+1}\right)+\delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^iW^{i+1}\right)+\delta * V_j}$ $= \frac{0+0{,}01}{1+0{,}01*2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| $C_1^3$ | tenaga | tenaga | $P_0(C_j^i) = \frac{count\left(C_j^i\right)+\delta}{\sum_{r=0}^{k_i} count\left(C_r^i\right)+\delta * V_j}$ $= \frac{1+0{,}01}{1+0{,}01*2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |
| | | buruh tenaga | $P_1(W_j^i \mid W^{i-1}) = \frac{count\left(W^{i-1}C_j^i\right)+\delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^i\right)+\delta * V_j}$ $= \frac{1+0{,}01}{1+0{,}01*2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |

| | | | |
|---|---|---|---|
| | | tenaga keraj | $P_2(W_j^i \mid W^{i+1}) = \frac{count\left(W^{i+1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i+1}C_r^i\right) + \delta * V_j}$ $= \frac{0+0{,}01}{0+0{,}01*2} = \frac{0{,}01}{0{,}02} = 0{,}5$ |
| | | buruh tenaga keraj | $P_3(W_j^i \mid W^{i-1}, W^{i+1})$ $= \frac{count\left(W^{i-1}C_j^iW^{i+1}\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^iW^{i+1}\right) + \delta * V_j}$ $= \frac{0+0{,}01}{0+0{,}01*2} = \frac{0{,}01}{0{,}02} = 0{,}5$ |
| $C_2^3$ | tenahak | tenahak | $P_0(C_j^i) = \frac{count\left(C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(C_r^i\right) + \delta * V_j}$ $= \frac{0+0{,}01}{1+0{,}01*2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| | | buruh tenahak | $P_1(W_j^i \mid W^{i-1}) = \frac{count\left(W^{i-1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^i\right) + \delta * V_j}$ $= \frac{0+0{,}01}{1+0{,}01*2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| | | tenahak keraj | $P_2(W_j^i \mid W^{i+1}) = \frac{count\left(W^{i+1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i+1}C_r^i\right) + \delta * V_j}$ $= \frac{0+0{,}01}{0+0{,}01*2} = \frac{0{,}01}{0{,}02} = 0{,}5$ |
| | | buruh tenahak keraj | $P_3(W_j^i \mid W^{i-1}, W^{i+1})$ $= \frac{count\left(W^{i-1}C_j^iW^{i+1}\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^iW^{i+1}\right) + \delta * V_j}$ $= \frac{0+0{,}01}{0+0{,}01*2} = \frac{0{,}01}{0{,}02} = 0{,}5$ |
| $C_1^4$ | kerja | kerja | $P_0(C_j^i) = \frac{count\left(C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(C_r^i\right) + \delta * V_j}$ $= \frac{1+0{,}01}{1+0{,}01*2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |
| | | tenaga kerja | $P_1(W_j^i \mid W^{i-1}) = \frac{count\left(W^{i-1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^i\right) + \delta * V_j}$ $= \frac{1+0{,}01}{1+0{,}01*2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |

| | | | |
|---|---|---|---|
| | | kerja naik | $P_2(W_j^i \mid W^{i+1}) = \frac{count\left(W^{i+1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i+1}C_r^i\right) + \delta * V_j}$ $= \frac{1+0{,}01}{1+0{,}01*2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |
| | | tenaga kerja naik | $P_3(W_j^i \mid W^{i-1}, W^{i+1})$ $= \frac{count\left(W^{i-1}C_j^iW^{i+1}\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^iW^{i+1}\right) + \delta * V_j}$ $= \frac{1+0{,}01}{1+0{,}01*2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |
| $C_2^4$ | kerai | kerai | $P_0(C_j^i) = \frac{count\left(C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(C_r^i\right) + \delta * V_j}$ $= \frac{0+0{,}01}{1+0{,}01*2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| | | tenaga kerai | $P_1(W_j^i \mid W^{i-1}) = \frac{count\left(W^{i-1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^i\right) + \delta * V_j}$ $= \frac{0+0{,}01}{1+0{,}01*2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| | | kerai naik | $P_2(W_j^i \mid W^{i+1}) = \frac{count\left(W^{i+1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i+1}C_r^i\right) + \delta * V_j}$ $= \frac{0+0{,}01}{1+0{,}01*2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| | | tenaga kerai naik | $P_3(W_j^i \mid W^{i-1}, W^{i+1})$ $= \frac{count\left(W^{i-1}C_j^iW^{i+1}\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^iW^{i+1}\right) + \delta * V_j}$ $= \frac{0+0{,}01}{1+0{,}01*2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| $C_1^5$ | naik | naik | $P_0(C_j^i) = \frac{count\left(C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(C_r^i\right) + \delta * V_j}$ $= \frac{1+0{,}01}{1+0{,}01*2} = \frac{1{,}01}{1{,}02} = 0{,}9901$ |
| | | keraj naik | $P_1(W_j^i \mid W^{i-1}) = \frac{count\left(W^{i-1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^i\right) + \delta * V_j}$ $= \frac{0+0{,}01}{0+0{,}01*2} = \frac{0{,}01}{0{,}02} = 0{,}5$ |

| | | | |
|---|---|---|---|
| | | naik _ | $P_2(W_j^i \mid W^{i+1}) = \frac{count\left(W^{i+1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i+1}C_r^i\right) + \delta * V_j}$ $= \frac{0 + 0{,}01}{0 + 0{,}01 * 2} = \frac{0{,}01}{0{,}02} = 0{,}5$ |
| | | keraj naik _ | $P_3(W_j^i \mid W^{i-1}, W^{i+1})$ $= \frac{count\left(W^{i-1}C_j^iW^{i+1}\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^iW^{i+1}\right) + \delta * V_j}$ $= \frac{0 + 0{,}01}{0 + 0{,}01 * 2} = \frac{0{,}01}{0{,}02} = 0{,}5$ |
| $C_2^5$ | naif | naif | $P_0(C_j^i) = \frac{count\left(C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(C_r^i\right) + \delta * V_j}$ $= \frac{0 + 0{,}01}{1 + 0{,}01 * 2} = \frac{0{,}01}{1{,}02} = 0{,}0098$ |
| | | keraj naif | $P_1(W_j^i \mid W^{i-1}) = \frac{count\left(W^{i-1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^i\right) + \delta * V_j}$ $= \frac{0 + 0{,}01}{0 + 0{,}01 * 2} = \frac{0{,}01}{0{,}02} = 0{,}5$ |
| | | naif _ | $P_2(W_j^i \mid W^{i+1}) = \frac{count\left(W^{i+1}C_j^i\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i+1}C_r^i\right) + \delta * V_j}$ $= \frac{0 + 0{,}01}{0 + 0{,}01 * 2} = \frac{0{,}01}{0{,}02} = 0{,}5$ |
| | | keraj naif _ | $P_3(W_j^i \mid W^{i-1}, W^{i+1})$ $= \frac{count\left(W^{i-1}C_j^iW^{i+1}\right) + \delta}{\sum_{r=0}^{k_i} count\left(W^{i-1}C_r^iW^{i+1}\right) + \delta * V_j}$ $= \frac{0 + 0{,}01}{0 + 0{,}01 * 2} = \frac{0{,}01}{0{,}02} = 0{,}5$ |

Kemudian, hasil perhitungan probabilitas dimasing-masing token dilanjutkan dengan salah satu perhitungan metode *smoothing* yaitu *jelinek mercer*. Persamaan yang digunakan yaitu persamaan pada 2.9, 2.10 dan 2.11. Perhitungan probabilitas dengan *jelinek mercer* dapat dilihat pada tabel 3.29 sebagai berikut.

**Tabel 3. 29 Contoh Perhitungan Probabilitas *Jelinek Mercer***

| Anggota ($C_j^i$) | | Bigram kiri, bigram kanan dan trigram | Probabilitas *Jelinek Mercer* |
|---|---|---|---|
| $C_1^1$ | biaya | _ biaya | $P_1'(C_j^i \mid W^{i-1}) = \lambda P_1(C_j^i \mid W^{i-1}) + (1 - \lambda)P_0(C_j^i)$ $= 0{,}1(0{,}9901) + (1 - 0{,}1)0{,}9901 = 0{,}9901$ |

| | | | |
|---|---|---|---|
| | | biaya buruh | $P_2'(C_j^i \mid W^{i+1}) = \lambda P_2(C_j^i \mid W^{i+1}) + (1-\lambda)P_0(C_j^i)$<br>$= 0{,}1(0{,}9901) + (1-0{,}1)0{,}9901 = 0{,}9901$ |
| | | _ biaya buruh | $P_3'(C_j^i \mid W^{i-1}, W^{i+1})$<br>$= \lambda P_3(C_j^i \mid W^{i-1}, W^{i+1}) + (1-\lambda)(P_1(C_j^i \mid W^{i+1}) + P_2(C_j^i \mid W^{i+1})/2)$<br>$= 0{,}1(0{,}9901) + (1-0{,}1)\left(0{,}9901 + \frac{0{,}9901}{2}\right)$<br>$= 0{,}09901 + (0{,}9)(1{,}48515) = 1{,}435645 = 1{,}44$ |
| $C_2^1$ | biasa | _ biasa | $P_1'(C_j^i \mid W^{i-1}) = \lambda P_1(C_j^i \mid W^{i-1}) + (1-\lambda)P_0(C_j^i)$<br>$= (0{,}1)(0{,}0098) + (0{,}9)(0{,}0098) = 0{,}0098$ |
| | | biasa buruh | $P_2'(C_j^i \mid W^{i+1}) = \lambda P_2(C_j^i \mid W^{i+1}) + (1-\lambda)P_0(C_j^i)$<br>$= (0{,}1)(0{,}0098) + (0{,}9)(0{,}0098) = 0{,}0098$ |
| | | _ biasa buruh | $P_3'(C_j^i \mid W^{i-1}, W^{i+1})$<br>$= \lambda P_3(C_j^i \mid W^{i-1}, W^{i+1}) + (1-\lambda)(P_1(C_j^i \mid W^{i+1}) + P_2(C_j^i \mid W^{i+1})/2)$<br>$= (0{,}1)(0{,}0098) + (0{,}9)\left(0{,}0098 + \frac{0{,}0098}{2}\right)$<br>$= 0{,}01421 = 0{,}014$ |
| $C_1^2$ | buruh | biaya buruh | $P_1'(C_j^i \mid W^{i-1}) = \lambda P_1(C_j^i \mid W^{i-1}) + (1-\lambda)P_0(C_j^i)$<br>$= 0{,}1(0{,}9901) + (1-0{,}1)0{,}9901 = 0{,}9901$ |
| | | buruh tenaga | $P_2'(C_j^i \mid W^{i+1}) = \lambda P_2(C_j^i \mid W^{i+1}) + (1-\lambda)P_0(C_j^i)$<br>$= 0{,}1(0{,}9901) + (1-0{,}1)0{,}9901 = 0{,}9901$ |
| | | biaya buruh tenaga | $P_3'(C_j^i \mid W^{i-1}, W^{i+1})$<br>$= \lambda P_3(C_j^i \mid W^{i-1}, W^{i+1}) + (1-\lambda)(P_1(C_j^i \mid W^{i+1}) + P_2(C_j^i \mid W^{i+1})/2)$<br>$= 0{,}1(0{,}9901) + (1-0{,}1)\left(0{,}9901 + \frac{0{,}9901}{2}\right)$<br>$= 0{,}09901 + (0{,}9)(1{,}48515) = 1{,}435645 = 1{,}44$ |
| $C_2^2$ | buruk | biaya buruk | $P_1'(C_j^i \mid W^{i-1}) = \lambda P_1(C_j^i \mid W^{i-1}) + (1-\lambda)P_0(C_j^i)$<br>$= (0{,}1)(0{,}0098) + (0{,}9)(0{,}0098) = 0{,}0098$ |
| | | buruk tenaga | $P_2'(C_j^i \mid W^{i+1}) = \lambda P_2(C_j^i \mid W^{i+1}) + (1-\lambda)P_0(C_j^i)$<br>$= (0{,}1)(0{,}0098) + (0{,}9)(0{,}0098) = 0{,}0098$ |
| | | biaya buruk tenaga | $P_3'(C_j^i \mid W^{i-1}, W^{i+1})$<br>$= \lambda P_3(C_j^i \mid W^{i-1}, W^{i+1}) + (1-\lambda)(P_1(C_j^i \mid W^{i+1}) + P_2(C_j^i \mid W^{i+1})/2)$ |

| | | | |
|---|---|---|---|
| | | | $= (0{,}1)(0{,}0098) + (0{,}9)\left(0{,}0098 + \frac{0{,}0098}{2}\right)$<br>$= 0{,}01421 = 0{,}014$ |
| $C_1^3$ | tenaga | buruh tenaga | $P_1'(C_j^i \mid W^{i-1}) = \lambda P_1(C_j^i \mid W^{i-1}) + (1 - \lambda)P_0(C_j^i)$<br>$= 0{,}1(0{,}9901) + (1 - 0{,}1)0{,}9901 = 0{,}9901$ |
| | | tenaga keraj | $P_2'(C_j^i \mid W^{i+1}) = \lambda P_2(C_j^i \mid W^{i+1}) + (1 - \lambda)P_0(C_j^i)$<br>$= 0{,}1(0{,}5) + (1 - 0{,}1)0{,}9901 = 0{,}94109 = 0{,}941$ |
| | | buruh tenaga keraj | $P_3'(C_j^i \mid W^{i-1}, W^{i+1})$<br>$= \lambda P_3(C_j^i \mid W^{i-1}, W^{i+1}) + (1 - \lambda)(P_1(C_j^i \mid W^{i+1}) + P_2(C_j^i \mid W^{i+1})/2)$<br>$= 0{,}1(0{,}5) + (0{,}9)\left(0{,}9901 + \frac{0{,}5}{2}\right)$<br>$= 0{,}05 + (0{,}9)(1{,}2401) = 1{,}16609 = 1{,}166$ |
| $C_2^3$ | tenahak | buruh tenahak | $P_1'(C_j^i \mid W^{i-1}) = \lambda P_1(C_j^i \mid W^{i-1}) + (1 - \lambda)P_0(C_j^i)$<br>$= (0{,}1)(0{,}0098) + (0{,}9)(0{,}0098) = 0{,}0098$ |
| | | tenahak keraj | $P_2'(C_j^i \mid W^{i+1}) = \lambda P_2(C_j^i \mid W^{i+1}) + (1 - \lambda)P_0(C_j^i)$<br>$= 0{,}1(0{,}5) + (1 - 0{,}1)0{,}0098 = 0{,}05882 = 0{,}059$ |
| | | buruh tenahak keraj | $P_3'(C_j^i \mid W^{i-1}, W^{i+1})$<br>$= \lambda P_3(C_j^i \mid W^{i-1}, W^{i+1}) + (1 - \lambda)(P_1(C_j^i \mid W^{i+1}) + P_2(C_j^i \mid W^{i+1})/2)$<br>$= 0{,}1(0{,}5) + (0{,}9)\left(0{,}0098 + \frac{0{,}5}{2}\right) = 0{,}28382 = 0{,}284$ |
| $C_1^4$ | kerja | tenaga kerja | $P_1'(C_j^i \mid W^{i-1}) = \lambda P_1(C_j^i \mid W^{i-1}) + (1 - \lambda)P_0(C_j^i)$<br>$= 0{,}1(0{,}9901) + (1 - 0{,}1)0{,}9901 = 0{,}9901$ |
| | | kerja naik | $P_2'(C_j^i \mid W^{i+1}) = \lambda P_2(C_j^i \mid W^{i+1}) + (1 - \lambda)P_0(C_j^i)$<br>$= 0{,}1(0{,}9901) + (1 - 0{,}1)0{,}9901 = 0{,}9901$ |
| | | tenaga kerja naik | $P_3'(C_j^i \mid W^{i-1}, W^{i+1})$<br>$= \lambda P_3(C_j^i \mid W^{i-1}, W^{i+1}) + (1 - \lambda)(P_1(C_j^i \mid W^{i+1}) + P_2(C_j^i \mid W^{i+1})/2)$<br>$= 0{,}1(0{,}9901) + (1 - 0{,}1)\left(0{,}9901 + \frac{0{,}9901}{2}\right)$<br>$= 0{,}09901 + (0{,}9)(1{,}48515) = 1{,}435645 = 1{,}44$ |
| $C_2^4$ | kerai | tenaga kerai | $P_1'(C_j^i \mid W^{i-1}) = \lambda P_1(C_j^i \mid W^{i-1}) + (1 - \lambda)P_0(C_j^i)$<br>$= (0{,}1)(0{,}0098) + (0{,}9)(0{,}0098) = 0{,}0098$ |
| | | kerai naik | $P_2'(C_j^i \mid W^{i+1}) = \lambda P_2(C_j^i \mid W^{i+1}) + (1 - \lambda)P_0(C_j^i)$<br>$= (0{,}1)(0{,}0098) + (0{,}9)(0{,}0098) = 0{,}0098$ |

| | | | |
|---|---|---|---|
| | | tenaga kerai naik | $P_3'(C_j^i \mid W^{i-1}, W^{i+1})$<br>$= \lambda P_3(C_j^i \mid W^{i-1}, W^{i+1}) + (1 - \lambda)(P_1(C_j^i \mid W^{i+1}) + P_2(C_j^i \mid W^{i+1})/2)$<br>$= (0{,}1)(0{,}0098) + (0{,}9)\left(0{,}0098 + \frac{0{,}0098}{2}\right)$<br>$= 0{,}01421 = 0{,}014$ |
| $C_1^5$ | naik | keraj naik | $P_1'(C_j^i \mid W^{i-1}) = \lambda P_1(C_j^i \mid W^{i-1}) + (1 - \lambda)P_0(C_j^i)$<br>$= 0{,}1(0{,}5) + (0{,}9)(0{,}9901) = 0{,}94109 = 0{,}941$ |
| | | naik _ | $P_2'(C_j^i \mid W^{i+1}) = \lambda P_2(C_j^i \mid W^{i+1}) + (1 - \lambda)P_0(C_j^i)$<br>$= 0{,}1(0{,}5) + (0{,}9)(0{,}9901) = 0{,}94109 = 0{,}941$ |
| | | keraj naik _ | $P_3'(C_j^i \mid W^{i-1}, W^{i+1})$<br>$= \lambda P_3(C_j^i \mid W^{i-1}, W^{i+1}) + (1 - \lambda)(P_1(C_j^i \mid W^{i+1}) + P_2(C_j^i \mid W^{i+1})/2)$<br>$= 0{,}1(0{,}5) + (0{,}9)\left(0{,}5 + \frac{0{,}5}{2}\right)$<br>$= 0{,}05 + 0{,}675 = 0{,}725$ |
| $C_2^5$ | naif | keraj naif | $P_1'(C_j^i \mid W^{i-1}) = \lambda P_1(C_j^i \mid W^{i-1}) + (1 - \lambda)P_0(C_j^i)$<br>$= 0{,}1(0{,}5) + 0{,}9(0{,}0098) = 0{,}05882 = 0{,}059$ |
| | | naif _ | $P_2'(C_j^i \mid W^{i+1}) = \lambda P_2(C_j^i \mid W^{i+1}) + (1 - \lambda)P_0(C_j^i)$<br>$= 0{,}1(0{,}5) + 0{,}9(0{,}0098) = 0{,}05882 = 0{,}059$ |
| | | keraj naif _ | $P_3'(C_j^i \mid W^{i-1}, W^{i+1})$<br>$= \lambda P_3(C_j^i \mid W^{i-1}, W^{i+1}) + (1 - \lambda)(P_1(C_j^i \mid W^{i+1}) + P_2(C_j^i \mid W^{i+1})/2)$<br>$= 0{,}1(0{,}5) + 0{,}9\left(0{,}5 + \frac{0{,}5}{2}\right)$<br>$= 0{,}05 + 0{,}675 = 0{,}725$ |

4. Perhitungan skor

   Perhitungan skor merupakan tahapan terakhir dalam proses deteksi dan koreksi kesalahan ejaan pada penelitian ini. Perhitungan skor diterapkan di masing-masing kata yang sebelumnya telah dihitung probabilitasnya. Perhitungan skor ini menggunakan persamaan *weighted combination score* pada persamaan 2.12. Persamaan tersebut dilakukan dengan cara menjumlahkan probabilitas yang telah didapatkan pada bigram dan trigram. Perhitungan skor dapat dilihat pada tabel 3.30 sebagai berikut.

**Tabel 3. 30 Perhitungan Skor**

| Anggota ($C_j^i$) | $score(C_j^i)$ |
|---|---|

| | | |
|---|---|---|
| $C_1^1$ | biaya | $Score\left(C_j^i\right) = \lambda_1 P_1\left(C_j^i \middle\| W^{i-1}\right) + \lambda_2 P_2\left(C_j^i \middle\| W^{i+1}\right) + \lambda_3 P_3\left(C_j^i \middle\| W^{i-1}, W^{i+1}\right)$ $= 0{,}25(0{,}9901) + 0{,}25(0{,}9901) + 0{,}5(1{,}44) = 1{,}21505 = 1{,}22$ |
| $C_2^1$ | biasa | $Score\left(C_j^i\right) = \lambda_1 P_1\left(C_j^i \middle\| W^{i-1}\right) + \lambda_2 P_2\left(C_j^i \middle\| W^{i+1}\right) + \lambda_3 P_3\left(C_j^i \middle\| W^{i-1}, W^{i+1}\right)$ $= 0{,}25(0{,}0098) + 0{,}25(0{,}0098) + 0{,}5(0{,}014) = 0{,}0119$ |
| $C_1^2$ | buruh | $Score\left(C_j^i\right) = \lambda_1 P_1\left(C_j^i \middle\| W^{i-1}\right) + \lambda_2 P_2\left(C_j^i \middle\| W^{i+1}\right) + \lambda_3 P_3\left(C_j^i \middle\| W^{i-1}, W^{i+1}\right)$ $= 0{,}25(0{,}9901) + 0{,}25(0{,}9901) + 0{,}5(1{,}44) = 1{,}21505 = 1{,}22$ |
| $C_2^2$ | buruk | $Score\left(C_j^i\right) = \lambda_1 P_1\left(C_j^i \middle\| W^{i-1}\right) + \lambda_2 P_2\left(C_j^i \middle\| W^{i+1}\right) + \lambda_3 P_3\left(C_j^i \middle\| W^{i-1}, W^{i+1}\right)$ $= 0{,}25(0{,}0098) + 0{,}25(0{,}0098) + 0{,}5(0{,}014) = 0{,}0119$ |
| $C_1^3$ | tenaga | $Score\left(C_j^i\right) = \lambda_1 P_1\left(C_j^i \middle\| W^{i-1}\right) + \lambda_2 P_2\left(C_j^i \middle\| W^{i+1}\right) + \lambda_3 P_3\left(C_j^i \middle\| W^{i-1}, W^{i+1}\right)$ $= 0{,}25(0{,}9901) + 0{,}25(0{,}941) + 0{,}5(1{,}166) = 1{,}065775 = 1{,}07$ |
| $C_2^3$ | tenahak | $Score\left(C_j^i\right) = \lambda_1 P_1\left(C_j^i \middle\| W^{i-1}\right) + \lambda_2 P_2\left(C_j^i \middle\| W^{i+1}\right) + \lambda_3 P_3\left(C_j^i \middle\| W^{i-1}, W^{i+1}\right)$ $= 0{,}25(0{,}0098) + 0{,}25(0{,}059) + 0{,}5(0{,}284) = 0{,}1592 = 0{,}16$ |
| $C_1^4$ | kerja | $Score\left(C_j^i\right) = \lambda_1 P_1\left(C_j^i \middle\| W^{i-1}\right) + \lambda_2 P_2\left(C_j^i \middle\| W^{i+1}\right) + \lambda_3 P_3\left(C_j^i \middle\| W^{i-1}, W^{i+1}\right)$ $= 0{,}25(0{,}9901) + 0{,}25(0{,}9901) + 0{,}5(1{,}44) = 1{,}21505 = 1{,}22$ |
| $C_2^4$ | kerai | $Score\left(C_j^i\right) = \lambda_1 P_1\left(C_j^i \middle\| W^{i-1}\right) + \lambda_2 P_2\left(C_j^i \middle\| W^{i+1}\right) + \lambda_3 P_3\left(C_j^i \middle\| W^{i-1}, W^{i+1}\right)$ $= 0{,}25(0{,}0098) + 0{,}25(0{,}0098) + 0{,}5(0{,}014) = 0{,}0119$ |
| $C_1^5$ | naik | $Score\left(C_j^i\right) = \lambda_1 P_1\left(C_j^i \middle\| W^{i-1}\right) + \lambda_2 P_2\left(C_j^i \middle\| W^{i+1}\right) + \lambda_3 P_3\left(C_j^i \middle\| W^{i-1}, W^{i+1}\right)$ $= 0{,}25(0{,}941) + 0{,}25(0{,}941) + 0{,}5(0{,}725) = 0{,}833$ |
| $C_2^5$ | naif | $Score\left(C_j^i\right) = \lambda_1 P_1\left(C_j^i \middle\| W^{i-1}\right) + \lambda_2 P_2\left(C_j^i \middle\| W^{i+1}\right) + \lambda_3 P_3\left(C_j^i \middle\| W^{i-1}, W^{i+1}\right)$ $= 0{,}25(0{,}059) + 0{,}25(0{,}059) + 0{,}5(0{,}725) = 0{,}392$ |

Setelah skor dihitung, hasil perhitungan perlu diurutkan dari yang terbesar hingga terkecil untuk mengetahui kata yang menjadi rekomendasi kata yang cocok sebagai pengganti kata yang terdeteksi salah dan sesuai konteks. Sebuah kata dianggap menjadi sebuah rekomendasi kata apabila hasil perhitungan skor tertinggi dibandingkan kata lainnya. Perangkingan kata dapat dilihat pada tabel 3.31 sebagai berikut.

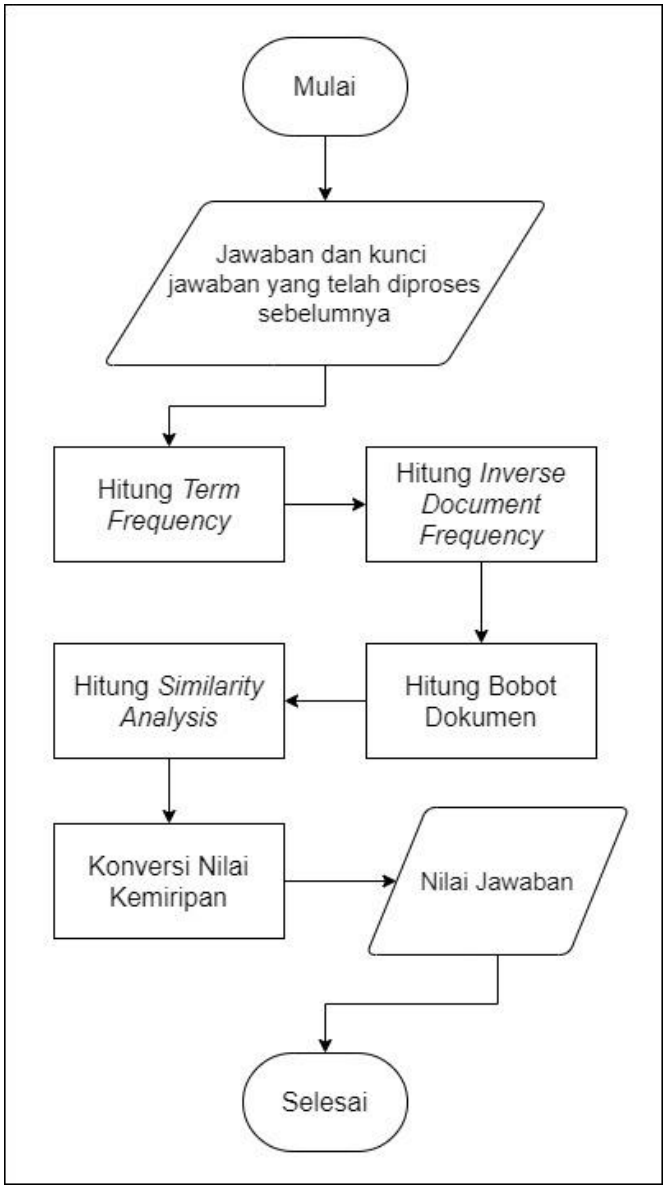
**Tabel 3. 31 Perangkingan Kata**

| $W^i$ | Anggota ($C_j^i$) | | $Score(C_j^i)$ | Status | Rekomendasi Kata |
|---|---|---|---|---|---|
| biaya | $C_1^1$ | **biaya** | 1,22 | Benar | - |
| | $C_2^1$ | biasa | 0,0119 | | |
| buruh | $C_1^2$ | **buruh** | 1,22 | Benar | - |
| | $C_2^2$ | buruk | 0,0119 | | |
| tenaga | $C_1^3$ | **tenaga** | 1,07 | Benar | - |
| | $C_2^3$ | tenahak | 0,16 | | |
| keraj | $C_1^4$ | **kerja** | 1,22 | Salah | kerja |

| | | | | | |
|---|---|---|---|---|---|
| | $C_2^4$ | kerai | 0,0119 | | |
| naik | $C_1^5$ | **naik** | 0,833 | Benar | - |
| | $C_2^5$ | naif | 0,392 | | |

### 3.1.5 *Scoring*

*Scoring* pada penelitian ini digunakan untuk melakukan penilaian data uji berupa jawaban dari sebuah soal yang dicocokkan dengan kunci jawaban. Kemudian, proses *scoring* ini menggunakan metode *Vector Space Model* (VSM). Selain itu, proses *scoring* ini sebelumnya sudah melalui tahap deteksi kesalahan ejaan terlebih dahulu. Alur tahapan scoring dapat dilihat pada gambar 3.6.




**Gambar 3. 6 Alur Tahapan *Scoring***

Berdasarkan gambar 3.6, terdapat beberapa tahapan yang perlu dilalui dalam proses *scoring*. Beberapa proses tersebut seperti perhitungan *Term Frequency* (TF), perhitungan *Inverse Document Frequency* (IDF), perhitungan bobot, perhitungan similaritas, yang kemudian dari hasil similaritas tersebut didapatkan nilai jawaban dari proses konversi nilai kemiripan berdasarkan tabel 2.1. Sebagai contoh pada proses *scoring* ini, dimisalkan terdapat sebuah kunci jawaban dan beberapa jawaban soal yang didapatkan dari responden yang digunakan sebagai data uji. Contoh jawaban dapat dilihat pada tabel 3.32 dan kunci jawaban pada tabel 3.33.

**Tabel 3. 32 Contoh Jawaban Responden**

| Jawaban Responden | |
|---|---|
| D1 | Alat hitung elektronik untuk mengolah data secara sistematis dan cermat |
| D2 | Alat pengolah data |

**Tabel 3. 33 Contoh Kunci Jawaban**

| Kunci Jawaban | |
|---|---|
| Pertanyaan | Apa yang dimaksud dengan komputer? |
| Kunci Jawaban (q) | Alat elektronik yang mampu mengolah data secara sistematis dan cermat |

Kemudian dari tabel 3.32 dan tabel 3.33 dilakukan proses pencocokan jawaban dengan kunci jawaban dengan metode Vector Space Model (VSM). Beberapa proses yang perlu dilalui pada pencocokan jawaban dan kunci jawaban sebagai berikut.

1. Perhitungan *Term Frequency* (TF)
   *Term Frequency* (TF) ini digunakan untuk menghitung bobot yang didasarkan pada jumlah kemunculan *term* di dalam dokumen. Contoh perhitungan *Term Frequency* (TF) dapat dilihat pada tabel 3.34 sebagai berikut.

**Tabel 3. 34 Contoh Perhitungan *Term Frequency* (TF)**

| Term | $term\ frequency = tfij$ | | | |
|---|---|---|---|---|
| | q | d1 | d2 | $dfi$ |
| alat | 1 | 1 | 1 | 3 |
| elektronik | 1 | 1 | 0 | 2 |
| yang | 1 | 0 | 0 | 1 |
| mampu | 1 | 0 | 0 | 1 |
| mengolah | 1 | 1 | 0 | 2 |
| data | 1 | 1 | 1 | 3 |
| secara | 1 | 1 | 0 | 2 |
| sistematis | 1 | 1 | 0 | 2 |
| dan | 1 | 1 | 0 | 2 |
| cermat | 1 | 1 | 0 | 2 |

| | | | | |
|---|---|---|---|---|
| hitung | 0 | 1 | 0 | 1 |
| untuk | 0 | 1 | 0 | 1 |
| pengolah | 0 | 0 | 1 | 1 |

2. Perhitungan *Inverse Document Frequency* (IDF)

   *Inverse Document Frequency* (IDF) merupakan salah satu pembobotan untuk melengkapi perhitungan pembobotan *Term Frequency* (TF). Contoh perhitungan dapat dilihat pada tabel 3.35 sebagai berikut.

**Tabel 3. 35 Contoh Perhitungan *Inverse Document Frequency* (IDF)**

| Term | $\frac{N}{dfi}$ | $idfi = log\frac{N}{dfi}$ |
|---|---|---|
| alat | 3/3 = 1 | 0 |
| elektronik | 3/2 = 1,5 | 0,176 |
| yang | 3/1 = 3 | 0,477 |
| mampu | 3/1 = 3 | 0,477 |
| mengolah | 3/2 = 1,5 | 0,176 |
| data | 3/3 = 1 | 0 |
| secara | 3/2 = 1,5 | 0,176 |
| sistematis | 3/2 = 1,5 | 0,176 |
| dan | 3/2 = 1,5 | 0,176 |
| cermat | 3/2 = 1,5 | 0,176 |
| hitung | 3/1 = 3 | 0,477 |
| untuk | 3/1 = 3 | 0,477 |
| pengolah | 3/1 = 3 | 0,477 |

3. Perhitungan bobot

   Perhitungan bobot ini dilakukan dengan cara melakukan operasi perkalian dari hasil *Term Frequency* (TF) dengan *Inverse Document Frequency* (IDF) menggunakan persamaan 2.19. Contoh perhitungan bobot dapat dilihat pada tabel 3.36.

**Tabel 3. 36 Contoh Perhitungan Bobot**

| Term | $W_{ij} = tfij\ (log\frac{N}{dfi} + 1)$ | | |
|---|---|---|---|
| | q | d1 | d2 |
| alat | 1 | 1 | 1 |
| elektronik | 1,176 | 1,176 | 0 |
| yang | 1,477 | 0 | 0 |
| mampu | 1,477 | 0 | 0 |
| mengolah | 1,176 | 1,176 | 0 |
| data | 1 | 1 | 1 |

| | | | |
|---|---|---|---|
| secara | 1,176 | 1,176 | 0 |
| sistematis | 1,176 | 1,176 | 0 |
| dan | 1,176 | 1,176 | 0 |
| cermat | 1,176 | 1,176 | 0 |
| hitung | 0 | 1,477 | 0 |
| untuk | 0 | 1,477 | 0 |
| pengolah | 0 | 0 | 1,477 |

4. Perhitungan *matching documents*

   Perhitungan vektor pada $q$ dan $d$ sebagai berikut.

   $|q| = \sqrt{2(1)^2 + 6(1{,}176)^2 + 2(1{,}477)^2}$

   $= \sqrt{2 + 6(1{,}38) + 2(1{,}18)} = \sqrt{14{,}64} = 3{,}83$

   $|d_1| = \sqrt{2(1)^2 + 6(1{,}176)^2 + 2(1{,}477)^2}$

   $= \sqrt{2 + 6(1{,}38) + 2(1{,}18)} = \sqrt{14{,}64} = 3{,}83$

   $|d_2| = \sqrt{2(1)^2 + (1{,}477)^2} = \sqrt{1 + 1{,}477} = \sqrt{2{,}477} = 1{,}57$

   Perhitungan *index* dari *term query*(q) dan dokumen (d) dengan cara menjumlahkan hasil perkalian bobot pada query dan dokumen sebagai berikut.

   $q * d_1 = 1(1) + 1{,}176(1{,}176) + 1{,}477(0) + 1{,}477(0) + 1{,}176(1{,}176) + 1(1)$
   $+1{,}176(1{,}176) + 1{,}176(1{,}176) + 1{,}176(1{,}176) + 1{,}176(1{,}176)$
   $+0(1{,}477) + 0(1{,}477) + 0(0)$
   $= 1 + 1{,}38 + 0 + 0 + 1{,}38 + 1 + 1{,}38 + 1{,}38 + 1{,}38 + 1{,}38 + 0 + 0 + 0$
   $= 10{,}28$

   $q * d_2 = 1(1) + 1{,}176(0) + 1{,}477(0) + 1{,}477(0) + 1{,}176(0) + 1(1)$
   $+1{,}176(0) + 1{,}176(0) + 1{,}176(0) + 1{,}176(0) + 0(0) + 0(0)$
   $+0(1{,}477)$
   $= 1 + 0 + 0 + 0 + 0 + 1 + 0 + 0 + 0 + 0 + 0 + 0 + 0$
   $= 2$

   Perhitungan similaritas dokumen dengan persamaan 2.20 sebagai berikut.

   $Sim\ (q, d) = \frac{q.d_1}{|q||d_1|} = \frac{10{,}28}{3{,}83 * 3{,}83} = \frac{10{,}28}{14{,}7} = 0{,}699$

   $Sim\ (q, d) = \frac{q.d_2}{|q||d_2|} = \frac{2}{3{,}83 * 1{,}57} = \frac{2}{6{,}01} = 0{,}33$

5. Konversi Nilai

   Pada tahapan ini dilakukan proses konversi nilai dari hasil perhitungan similaritas ke dalam bentuk nilai nilai ujian uraian (*essay*) dengan menggunakan rentang nilai pada tabel 2.1. Jika dengan menggunakan contoh hasil perhitungan similaritas pada analisis ini berarti nilai similaritas sebesar 0,699 dikonversikan berdasarkan tabel 2.1 akan memperoleh nilai 70 kemudian dibagi dengan jumlah soal yang ada karena

proses konversi ini bergantung pada jumlah soal yang ada seperti yang sudah dijelaskan pada bab sebelumnya.

### 3.1.6 Pengujian

Pengujian dalam penelitian ini digunakan untuk mengetahui seberapa akurat hasil yang dihasilkan sistem dalam proses deteksi dan koreksi kesalahan ejaan dengan menggunakan metode n-gram dan *jaro winkler*. Pengujian ini dilakukan dengan cara melakukan perhitungan presisi, recall dan akurasi menggunakan persamaan 2.21, 2.22, dan 2.23. Setiap pengujian akan dicatat ke dalam tabel matriks seperti pada tabel 3.37 sebagai berikut.

**Tabel 3. 37 Pengujian Deteksi dan Koreksi Kesalahan Ejaan**

| No | Kata yang salah ejaan | Hasil deteksi | | TP | FP | FN | TN |
|---|---|---|---|---|---|---|---|
| | | Sistem | Manual | | | | |
| 1. | | | | | | | |
| 2. | | | | | | | |

## 3.2 Analisis Kebutuhan

Analisis kebutuhan pada penelitian ini terdiri dari kebutuhan fungsional dan kebutuhan non fungsional.

### 3.2.1 Kebutuhan Fungsional

Kebutuhan fungsional merupakan salah satu kebutuhan berupa proses atau hal apa saja yang dapat dilakukan oleh sistem pada penelitian ini. Kebutuhan fungsional pada penelitian ini sebagai berikut :

1. Sistem ini dapat melakukan pendeteksian dan koreksi kesalahan ejaan dan kalkulasi penilaian pada jawaban uraian responden.
2. Sistem ini akan menampilkan rekomendasi kata berdasarkan hasil proses metode yang diterapkan yaitu n-gram dan *jaro winkler*.
3. Sistem ini akan menampilkan hasil kalkulasi penilaian pada jawaban uraian responden setelah dilakukan proses deteksi dan koreksi kesalahan ejaan.

### 3.2.2 Kebutuhan Non Fungsional

Kebutuhan non fungsional merupakan salah satu kebutuhan yang berisi spesifikasi dalam penggunaan sebuah sistem. Spesifikasi yang dibutuhkan yaitu perangkat keras (*hardware*) dan perangkat lunak (*software*).

1. Analisis kebutuhan perangkat keras (*hardware*)

   Kebutuhan perangkat keras (*hardware*) yang digunakan untuk mengembangkan sistem pada penelitian ini dapat dilihat pada tabel 3.38.

**Tabel 3. 38 Kebutuhan Perangkat Keras**

| **No** | **Perangkat keras (*hardware*)** | **Keterangan** |
|---|---|---|
| 1 | *Processor* | Intel i5-825OU |

| | | |
|---|---|---|
| 2 | RAM | 8,00 GB |
| 3 | *Storage* | Hardisk 1 TB |
| 4 | *Graphic* | NVIDIA GeForce 930MX |
| 5 | Perangkat *input* dan *output* | *Keyboard*, *mouse*, dan monitor |
| 6 | Koneksi internet | Wifi |

2. Analisis kebutuhan perangkat lunak (*software*)

Kebutuhan perangkat lunak (software) yang digunakan untuk mengembangkan sistem pada penelitian ini dapat dilihat pada tabel 3.39.

**Tabel 3. 39 Kebutuhan Perangkat Lunak**

| **No** | **Perangkat lunak (s*oftware*)** | **Keterangan** |
|---|---|---|
| 1 | Windows 10 64 bit | Sistem operasi |
| 2 | Python | Bahasa pemrograman |
| 3 | *Visual Studio Code* | *Text editor* |
| 4 | XAMPP | *Server* lokal |
| 5 | MySQL | *Database* |
| 6 | Microsoft Edge, Google Chrome | *Browser* |
| 7 | Draw.io | Perangkat lunak yang digunakan untuk perancangan seperti pembuatan diagram |

## 3.3 Metodologi Pengembangan Sistem

Metode pengembangan sistem pada penelitian ini menggunakan metode *prototyping*. Tahapan penggunaan metode *prototyping* yaitu komunikasi, membangun *prototype*, evaluasi *prototype*, *coding*, pengujian dan evaluasi sistem, penggunaan sistem. Komunikasi merupakan tahapan awal di dalam metode *prototyping* untuk mengetahui kebutuhan perangkat lunak yang akan dikembangkan. Setelah kebutuhan perangkat lunak telah diidentifikasi, maka dilanjutkan dengan membuat sebuah *prototype* atau sebuah rancangan. Rancangan tersebut digunakan sebagai acuan untuk tahap selanjutnya yaitu *coding*. Akan tetapi, sebelum tahap *coding* diperlukan sebuah evaluasi rancangan terlebih dahulu apakah sudah sesuai atau belum. Jika belum maka diperlukannya perbaikan rancangan terlebih dahulu. Setelah proses *coding*, dibutuhkannya sebuah pengujian dan evaluasi sistem. Jika ditahap evaluasi dan pengujian sudah berhasil maka dapat dikatakan sistem siap untuk digunakan (Pressman, 2005).

## 3.4 Jadwal Penelitian

**Tabel 3. 40 Jadwal Penelitian**

| **No** | **Kegiatan** | **Bulan I** | | | | **Bulan II** | | | | **Bulan III** | | | | **Bulan IV** | | | | **Bulan V** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** |
| 1 | **Pengumpulan Data** | | | | | | | | | | | | | | | | | | | | |
| | Pengambilan Data | | | | | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Analisis Data | | ■ | ■ | | | | | | | | | | | | | | | | |
| 2 | **Perancangan Sistem** | | | | | | | | | | | | | | | | | | | |
| | *Planning* | | | | ■ | ■ | | | | | | | | | | | | | | |
| | *Analysis* | | | | | ■ | ■ | | | | | | | | | | | | | |
| | *Design* | | | | | | ■ | ■ | ■ | | | | | | | | | | | |
| | *Implementation* | | | | | | | | | ■ | ■ | ■ | ■ | | | | | | | |
| 3 | **Pengujian Sistem** | | | | | | | | | | | | | | | | | | | |
| | Pengujian dan Evaluasi | | | | | | | | | | | | | ■ | ■ | ■ | | | | |
| 4 | **Pembuatan Laporan** | | | | | | | | | | | | | | | | | | | |
| | Pembuatan Laporan | | | | | | | | | | | | | | | | ■ | ■ | ■ | ■ |

# Daftar Pustaka


Adhitia, R., & Purwarianti, A. (2012). Penilaian Esai Jawaban Bahasa Indonesia Menggunakan Metode Svm - Lsa Dengan Fitur Generik. *Jurnal Sistem Informasi*, *5*(1), 33. https://doi.org/10.21609/jsi.v5i1.260

Ariyani, N. H., Sutardi, & Ramadhan, R. (2016). Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode Levenshtein Distance. *SemanTIK, Vol 2*(1), 279–286. http://ojs.uho.ac.id/index.php/semantik/article/download/1030/661

Azmi, A. M., Almutery, M. N., & Aboalsamh, H. A. (2019). Real-Word Errors in Arabic Texts: A Better Algorithm for Detection and Correction. *IEEE/ACM Transactions on Audio Speech and Language Processing*, *27*(8), 1308–1320. https://doi.org/10.1109/TASLP.2019.2918404

Christanti, M. V., Rudy, & Naga, D. S. (2018). Fast and accurate spelling correction using trie and Damerau-levenshtein distance bigram. *Telkomnika (Telecommunication Computing Electronics and Control)*, *16*(2), 827–833. https://doi.org/10.12928/TELKOMNIKA.v16i2.6890

Dewi, N. C., & Qoiriah, A. (2020). *Implementasi Algoritma Jaro Winkler Distance dan N-Gram Untuk Deteksi dan Prediksi Perbaikan Kesalahan Penulisan Kata Bahasa Indonesia Pada Karya Tulis Ilmiah*. *02*.

Frando, J., Ruslianto, I., & Hidayati, R. (2019). Penerapan Jaro Winkler Distance Dalam Aplikasi Pengoreksi Kesalahan Penulisan Bahasa Indonesia Berbasis Web [1]. *Jurnal Komputer Dan Aplikasi*, *07*(03), 44–53.

Fuat, R. (2010). *Sistem penilaian esai otomatis pada e-learning dengan metode cosine similarity*.

Indriani, A., Muhammad, M., Suprianto, S., & Hadriansa, H. (2018). Implementasi Jaccard Index Dan N-Gram Pada Rekayasa Aplikasi Koreksi Kata Berbahasa Indonesia. *Sebatik*, *22*(2), 95–101. https://doi.org/10.46984/sebatik.v22i2.314

Indriyono, B. V. (2020). Kombinasi Damerau Levenshtein dan Jaro-Winkler Distance Untuk Koreksi Kata Bahasa Inggris. *Jurnal Teknik Informatika Dan Sistem Informasi*, *6*(2),

162–173. https://doi.org/10.28932/jutisi.v6i2.2493

Jurafsky, D., & Martin, James H. (2020). Speech and Language Processing. 3rd ed. [e-book] Upper Saddle River : Prentice Hall. Tersedia di : <https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf> [Diakses 20 Januari 2021]

Kaur, G., Kaur, K., & Singh, P. (2019). Spell Checker for Punjabi Language Using Deep Neural Network. *2019 5th International Conference on Advanced Computing and Communication Systems, ICACCS 2019*, 147–151. https://doi.org/10.1109/ICACCS.2019.8728369

Kumar, Ela. (2011). Natural Language Processing. [e-book] New Delhi : I.K. International Publishing House Pvt. Ltd. Tersedia di : Google Books <https://books.google.co.id/books?hl=id&lr=&id=FpUBFNFuKWgC&oi=fnd&pg=PP2&dq=natural+language+processing+adalah&ots=GFu-9MeCUt&sig=7Tk05WivCqxap43S60l2upHFoT4&redir_esc=y#v=onepage&q=natural%20language%20processing%20adalah&f=false> [Diakses 20 Januari 2021]

Maghfira, T., Cholissodin, I., & Widodo, A. (2017). Deteksi Kesalahan Ejaan dan Penentuan Rekomendasi Koreksi Kata yang Tepat Pada Dokumen Jurnal JTIIK Menggunakan Dictionary Lookup dan Damerau-Levenshtein Distance. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *1*(6), 498–506.

Mandala, R. (2006). Evaluasi Kinerja Sistem Penyaringan Informasi Model Vektor. *Seminar Nasional Aplikasi Teknologi Informasi 2006 (SNATI 2006)*, *2006*(Snati), 1–6. https://journal.uii.ac.id/Snati/article/view/1547

Okta'mal, F., Saptono, R., & Sulistyo, M. E. (2015). Jaro-Winkler Distance Dan Stemming Untuk Deteksi. *Seminar Nasional Sistem Informasi Indonesia*, *November*, 305–312.

Prasetyo, A., Baihaqi, W. M., & Had, I. S. (2018). Algoritma Jaro-Winkler Distance: Fitur Autocorrect dan Spelling Suggestion pada Penulisan Naskah Bahasa Indonesia di BMS TV. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *5*(4), 435. https://doi.org/10.25126/jtiik.201854780

Rahimi, F., & Asyikin, A. N. (2015). Aplikasi Penilaian Ujian Essay Otomatis Menggunakan Metode Cosine Similarity. *Poros Teknik*, *7*(2), 88–94. http://ejurnal.poliban.ac.id/index.php/porosteknik/article/view/218

Robinson, L. (2014). Implementasi Metode Generalized Vector Space Model Pada Aplikasi Information Retrieval untuk Pencarian Informasi Pada Kumpulan Dokumen Teknik Elektro Di UPT BPI LIPI. *Jurnal Ilmiah Komputer Dan Informatika ( KOMPUTA )*.

Rochmawati, Y., & Kusumaningrum, R. (2015). *Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching untuk Identifikasi Kesalahan Pengetikan Teks*. 125–134.

Samanta, P., & Chaudhuri, B. B. (2013). A simple real-word error detection and correction using local word bigram and trigram. *Proceedings of the 25th Conference on Computational Linguistics and Speech Processing, ROCLING 2013*, *Rocling*, 211–220.

Setiawan,Samhis. (2020). Kalimat Definisi dan Kalimat Deskripsi. [online] Tersedia di : <https://www.gurupendidikan.co.id/kalimat-definisi-dan-kalimat-deskripsi/> [Diakses 20 Januari 2021]

Sihole, F. (2018). *Perbandingan Metode Smoothing Untuk Deteksi dan Koreksi Kesalahan Kata Dalam Teks Berbahasa Indonesia* [Universitas Komputer Indonesia]. https://elib.unikom.ac.id/gdl.php?mod=browse&op=read&id=jbptunikompp-gdl-fernandosh-39069

Simanjuntak, M. S., Sujaini, H., & Safriadi, N. (2018). Spelling Corrector Bahasa Indonesia dengan Kombinasi Metode Peter Norvig dan N-Gram. *Jurnal Edukasi Dan Penelitian Informatika (JEPIN)*, *4*(1), 17. https://doi.org/10.26418/jp.v4i1.24075

Siregar, M. A. (2017). *Deteksi dan Koreksi Kesalahan Real Word Pada Penulisan Kata Menggunakan Metode N-Gram Dalam Teks Berbahasa Indonesia* [Universitas Komputer Indonesia]. https://elib.unikom.ac.id/gdl.php?mod=browse&op=read&id=jbptunikompp-gdl-muhammadab-37794

Sulistyo, M. E., Saptono, R., & Asshidiq, A. (2015). Penilaian Ujian Bertipe Essay

Menggunakan Metode Text Similarity. *Telematika*, *12*(2), 146–158. https://doi.org/10.31315/telematika.v12i2.1422

Suryaningrum, K. M. (2019). Implementasi Algoritma Confix Stripping untuk Pendeteksian Kesalahan pada Tenses. *Aiti*, *16*(1), 88–98. https://doi.org/10.24246/aiti.v16i1.88-98

Tinaliah, T., & Elizabeth, T. (2018). Perbandingan Hasil Deteksi Plagiarisme Dokumen dengan Metode Jaro-Winkler Distance dan Metode Latent Semantic Analysis. *Jurnal Teknologi Dan Sistem Komputer*, *6*(1), 7–12. https://doi.org/10.14710/jtsiskom.6.1.2018.7-12

Wardhana, W. S., Wirayuda, T. A. B., & Shaufiah. (2011). *Pengoreksian Ejaan Kata Menggunakan Metode N-Gram (Studi Kasus Dokumen Teks Berbahasa Indonesia)*. 0–6.

Wihardodo. (2018). *Implementasi Perbaikan Kesalahan Ejaan Pada Sistem Essay Scoring* [Universitas Komputer Indonesia]. https://elib.unikom.ac.id/gdl.php?mod=browse&op=read&id=jbptunikompp-gdl-wihardodon-39073

Yulianingsih, Y. (2017). Implementasi Algoritma Jaro-Winkler dan Levenstein Distance dalam Pencarian Data pada Database. *STRING (Satuan Tulisan Riset Dan Inovasi Teknologi)*, *2*(1), 18. https://doi.org/10.30998/string.v2i1.1720