Implementasi Algoritma Jaro-Winkler Distance dan N-gram untuk Deteksi dan Prediksi Perbaikan Kesalahan Penulisan Kata Bahasa Indonesia pada Karya Tulis Ilmiah Mahasiswa

Nuzhul Citrasari Dewi[1], Anita Qoiriah[2]

[1,2]Teknik Informatika/S1-Teknik Informatika, Universitas Negeri Surabaya

[1]nuzhuldewi16051204012@mhs.unesa.ac.id
[2]anitaqoiriah@unesa.ac.id

***Abstrak*— Karya tulis ilmiah merupakan sebuah tulisan yang berisi suatu permasalahan yang ditulis dan disajikan berdasarkan fakta dan data hasil penelitian. Kesalahan berbahasa yang sering terjadi pada karya tulis ilmiah mahasiswa adalah kesalahan penulisan kata. Untuk menghasilkan karya tulis ilmiah tanpa ada kesalahan penulisan kata, diperlukannya teknik tertentu untuk memperbaikinya. Penelitian ini membuat sebuah sistem deteksi dan prediksi perbaikan kesalahan penulisan kata. Algoritma *Jaro-Winkler Distance* merupakan algoritma yang digunakan untuk mengukur kemiripan antara 2 string. Algoritma *Jaro-Winkler Distance* pada sistem ini digunakan untuk daftar saran kandidat perbaikan kata. Saran kandidat perbaikan kata tersebut akan dicari yang sesuai dengan kalimat dengan menggunakan metode *N-Gram*. Metode *N-Gram* pada sistem ini digunakan untuk mencari saran perbaikan terbaik dari daftar kandidat yang dihasilkan oleh Algoritma *Jaro-Winkler Distance*. Hasil terbaik yang diberikan oleh sistem ini adalah sebesar 85.7% dan hasil terkecil sebesar 45%. Hasil tersebut dipengaruhi oleh kuantitas korpus yang digunakan untuk deteksi maupun prediksi perbaikan kata. Semakin baik kuantitas korpus/kamus yang digunakan pada sistem, maka sistem dapat memberikan prediksi perbaikan kata yang sesuai dengan perbaikan kata salah dalam kalimat.**

Kata Kunci*— *Karya Tulis Ilmiah, Jaro-Winkler Distance, N-Gram

I. PENDAHULUAN

Karya tulis ilmiah merupakan sebuah karya yang memuat suatu permasalahan yang ditulis dan diajukan berdasarkan fakta dan data penelitian. Menurut Brotowidjoyo, karya tulis ilmiah ialah karangan ilmu pengetahuan yang didasarkan pada fakta dan ditulis dengan cara penyusunan yang baik dan benar [9]. Dalam penulisan karya tulis ilmiah harus sesuai dengan kaidah-kaidah keilmuan penulisan karya tulis ilmiah seperti penulisan yang bersifat objektif, logis, sistematis, empiris, jelas dan konsisten. Kata merupakan kunci untuk membentuk karangan ketika menulis karya tulis ilmiah. Penulisan kata Bahasa Indonesia pada karya tulis ilmiah harus benar, agar ide maupun pesan yang disampaikan penulis dapat dimengerti oleh pembaca. Kesalahan berbahasa yang sering terjadi pada karya tulis ilmiah mahasiswa adalah kesalahan penulisan kata.

Kesalahan penulisan kata atau biasa disebut *typographical error* adalah salah satu kesalahan yang sering dilakukan oleh penulis pada saat menulis suatu karya tulis ilmiah. Kesalahan penulisan kata atau tipografi adalah kesalahan yang dilakukan saat proses mengetik. Kesalahan tersebut merupakan kesalahan penulisan kata yang disebabkan karena ketidaksengajaan, kegagalan mekanis atau slip pada jari, dan letak huruf pada *keyboard* yang berdekatan. Pada aplikasi pengolah kata terdapat fitur untuk perbaikan terhadap kesalahan kata secara otomatis, namun fitur tersebut belum digunakan secara maksimal sehingga kesalahan penulisan kata masih sering terjadi. Penulis cenderung tidak menyadari adanya kesalahan penulisan kata, karena pada umumnya penulis hanya berfokus pada isi karya tulis

ilmiah yang disusunnya dan tidak memperhatikan penulisan kata dalam karya tulis ilmiahnya. Untuk menghasilkan sebuah karya tulis ilmiah yang baik, penulis harus mengecek kembali semua tulisannya secara manual. Proses pengecekan kesalahan penulisan kata secara manual akan membutuhkan banyak waktu. Oleh karena itu, diperlukannya teknik tertentu untuk memperbaiki kesalahan penulisan kata pada karya tulis ilmiah. Perbaikan kesalahan penulisan kata dapat menggunakan salah satu fitur *natural language processing* yaitu pengecekan kesalahan ejaan/penulisan kata (*spelling correction*) dengan metode pencocokan *string*.

String matching / pencocokan *string* dibedakan menjadi dua yaitu *exact string matching* dan *inexact string matching*. *Inexact sting matching* ialah metode pencocokan *string* yang melaksanakan pencarian terhadap *string* yang sama dan *string* yang mirip dengan kata pada kamus/korpus. *Inexact string matching* dapat dibedakan menajdi dua yaitu *approximate string matching* dan *phonetic string matching*. *Approximate string matching* dapat digunakan untuk pencarian *string* berdasar kemiripan penulisan *string* dengan *string* dalam kamus dan berdasarkan string yang sama. Algoritma-algoritma yang telah digunakan untuk pencarian *string* adalah algoritma *Hamming Distance*, *Jaccard Distance*, *Levenshtein Distance*, *Damerau Levenshtein Distance*, *Jaro Distance* dan *Jaro-Winkler Distance*.

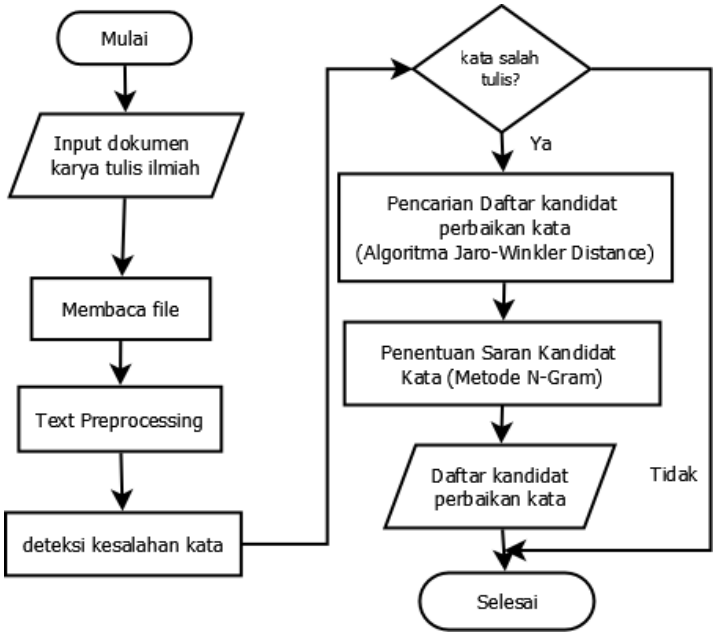
Berdasarkan penelitian tentang algoritma-algoritma *approximate string matching* untuk identifikasi kesalahan penulisan teks yang dilakukan oleh Yeny Rochmawati dan Retno Kusumaningrum tahun 2016, algoritma *Jaro-Winkler Distance* memberikan hasil yang paling baik dari algoritma yang lain yaitu dengan nilai *Mean Average Precision* (MAP) sebesar 0,87 [1]. Beberapa penelitian menggunakan algoritma *Jaro-Winkler Distance* sudah dilakukan, antara lain adalah penelitian yang dilakukan oleh Agung Prasetyo, dkk yang berjudul "Algoritma *Jaro-Winkler* Distance: Fitur *Autocorrect* dan *Spelling Sugesstion* pada Penulisan Naskah Bahasa Indonesia di BMS TV". Dalam penelitian yang telah dilakukan, dapat disimpulkan bahwa fitur yang dibuat dapat menangani kesalahan penulisan ejaan pada naskah bahasa Indonesia. Fitur ini dapat menangani kesalahan ejaan sebanyak 49 kata dari 60 kata salah dengan baik. Namun masih terjadi kesalahan dalam menampilkan saran ejaan. Untuk menangani permasalahan tersebut dapat ditambahkannya pembobotan kata pada daftar kata untuk meningkatkan akurasi fitur dalam menangani kesalahan penulisan kata [2].

Penelitian tentang koreksi kesalahan kata dengan menambahkan sebuah metode pembobotan kandidat kata telah dilakukan oleh Ariana Indana Fahma, dkk yang berjudul "Identifikasi Kesalahan Penulisan Kata (*Typographical Error*) pada Dokumen Bahasa Menggunakan Metode *N-Gram* dan *Levenshtein Distance*". Hasil nilai presisi dari penelitian tersebut yaitu sebesar 0,97 dan nilai *recall* sebesar 0,97. Penelitian ini menunjukkan bahwa metode *n-gram* dapat meningkatkan akurasi fitur dalam menangani kesalahan penulisan kata [3].

Berdasarkan latar belakang permasalahan tersebut, peneliti ingin mengimplementasikan algoritma *Jaro-Winkler Distance* dan *N-Gram* untuk deteksi dan prediksi perbaikan kesalahan penulisan kata bahasa Indonesia pada karya tulis ilmiah mahasiswa. Algoritma *Jaro-Winkler Distance* digunakan untuk mencari daftar saran kandidat kata, sedangkan algoritma *N-Gram* untuk memberikan saran terbaik dari daftar saran kandidat kata.

II. METODOLOGI PENELITIAN

Penelitian ini bertujuan untuk menerapkan salah satu metode dari *text mining* pencocokan kata yaitu *Jaro-Winkler Distance* dan *N-Gram* untuk deteksi dan prediksi perbaikan kesalahan penulisan kata bahasa Indonesia. Alur sistem ditunjukkan pada Gbr. 1.



Gbr 1. Alur sistem menggunakan algoritma *Jaro-Winkler Distance* dan *N-Gram*

Sistem mendapat *input* teks dokumen karya tulis ilmiah oleh *user*. Teks dokumen melalui proses pertama yaitu *text preprocessing*, deteksi kesalahan kata, pencarian kandidat perbaikan kata

A. Karya Tulis Ilmiah

Karya Tulis Ilmiah merupakan sebuah tulisan yang berisi suatu permasalahan yang ditulis dan diungkapkan sesuai dengan metode-metode ilmiah yang berdasarkan pada kaidah penulisan karya tulis ilmiah [4]. Tujuan dari karya tulis ilmiah untuk mahasiswa adalah untuk membuktikan wawasan dan potensi ilmiah yang dimiliki mahasiswa dalam menyelesaikan suatu permasalahan dalam bentuk karya tulis ilmiah. Menurut jenjang akademk, karya tulis ilmiah dibedakan menjadi 5 jenis yaitu makalah, laporan penelitian, skripsi, tesis dan disertasi.

B. Kesalahan Penulisan Kata

Kesalahan penulisan kata (*typographical error*) adalah kesalahan penulisan susunan kata atau ejaan kata yang disebabkan oleh penulisan secara manual ataupun pengetikan oleh mesin ketik dan komputer [1]. Kesalahan penulisan kata dibedakan menjadi 2 tipe yaitu kesalahan penulisan kata yang tidak ditemukan dalam kamus dan kesalahan penulisan kata yang ditemukan dalam kamus tetapi bukan kata yang dimaksud dalam dokumen.

C. Preprocessing

Pra-pemrosesan teks (*text preprocessing*) adalah proses transformasi suatu dokumen dan *query* yang menjadi kata-kata indeks. Pra-pemrosesan teks merupakan tahap awal dari *text mining* yang memiliki tujuan untuk menyiapkan teks yang telah menjadi data untuk diproses pada tahapan selanjutnya. Proses ini diperlukan untuk melakukan pengubahan bentuk suatu teks menjadi data yang terstruktur sesuai dengan kebutuhan. Dalam pra-pemrosesan teks terdapat tahapan seperti *case folding*, *filtering*, *tokenizing* dan *stemming* [5].

Pada penelitian ini terdapat 2 kali tahap pra pemrosesan teks yaitu pada data yang akan diuji dan pada pembuatan korpus *n-gram*. Pra-pemrosesan teks pada data uji dan korpus n-gram adalah *case folding, stemming, filtering dan tokenisasi*. Berikut adalah tahapan pra-pemrosesan teks:

1. *Case folding*

   Case folding atau *transform case* merupakan proses mengubah seluruh kata dalam teks dokumen menjadi huruf kecil atau *lower case*. Pada proses ini hanya huruf a hingga z yang dapat diterima dan karakter atau tanda baca lainnya akan dihilangkan.

2. *Stemming*

   Stemming merupakan proses menghapus imbuhan awalan atau akhiran pada suatu kata. Tujuan adanya proses *stemming* adalah untuk mendapatkan kata dasar dan mencegah kesalahan perhitungan saat proses ekstraksi fitur sintaksis. Dalam penelitian ini, peneliti menggunakan algoritma Nazief dan Adriani. Berikut tahapan algoritma *stemming* Nazief dan Adriani:

   a. Langkah pertama adalah memeriksa apakah kata tersebut merupakan kata dasar/root. Apabila kata tersebut merupakan kata dasar/root, algoritma berhenti.
   b. Menghapus infleksi akhiran (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”). Apabila kata tersebut berupa partikel (“-lah”, “-kah”, “-tah” atau “-pun”) maka, ulang langkah ini untuk menghapus kata ganti posesif (“-ku”, “-mu”, “-nya”), jika ada.
   c. Hapus sufiks / penurunan akhiran (“-i”, “-an”, “-kan”). Jika menemukan sebuah kata dalam kamus, maka kata berhenti. Apabila tidak, kemudian akan dilanjutkan ke langkah 1).
      1) Bila “-an” dihapus dan huruf terakhir dari kata tersebut merupakan huruf “-k”, huruf tersebut akan dihapus juga. Apabila tidak ditemukan, lanjutkan ke langkah 2).
      2) Kembalikan ke akhiran yang dihapus (“-i”, “-an”, atau “-kan”) dan lanjutkan ke langkah keempat.
   d. Hapus awalan dan buang. Jika ada sufiks/akhiran yang dihapus di langkah ketiga, maka lanjutkan ke langkah 1), apabila tidak, lanjutkan ke langkah 2).
      1) Periksa daftar kombinasi awalan dan akhiran yang tidak diizinkan. Apabila kombinasi ditemukan, maka algoritma akan berhenti, apabila tidak ditemukan maka akan melanjutkan ke langkah 2).
      2) Pada langkah ini akan diulang sebanyak tiga kali. Tentukan jenis awalan, dan kemudian hapus awalan. Apabila kata dasar belum ditemukan, maka akan melanjutkan langkah 5, apabila kata dasar sudah ditemukan, algoritma berhenti.

e. Melakukan *recoding*.
f. Apabila seluruh langkah selesai namun tidak berhasil juga, kata awal dapat dianggap sebagai kata dasar, dan proses berakhir.

3. *Filtering*
Filtering merupakan proses menghilangkan kata-kata yang tidak perlu atau menghilangkan kata yang ada dalam *list stopword*. Proses *filtering* bertujuan untuk meningkatkan tingkat keakurasian terhadap kata-kata penting.
4. *Tokenizing*
Tokenisasi atau *tokenizing* merupakan proses pemisahan setiap kata dari suatu dokumen dan sekumpulan kata tersebut dapat berdiri sendiri. Proses tokenisasi ini sebelumnya menghilangkan angka dan simbol-simbol yang tidak penting dalam suatu dokumen.

D. Algoritma *Jaro-Winkler Distance.*

Algoritma *Jaro-Winkler Distance* merupakan algoritma yang digunakan untuk mengukur kemiripan kata antara 2 string/kata [6]. Algoritma *Jaro-Winkler Distance* merupakan salah satu varian yang ditemukan oleh Mathew A. Kemudian dikembangkan oleh William E. Winkler dan Thibaudeau dengan memodifikasi Jaro Distance agar memiliki bobot yang lebih tinggi. Semakin tinggi nilai *Jaro-Winkler Distance* pada kedua *string* tersebut, maka *string* tersebut akan semakin mirip. Nilai normal pada algoritma ini ialah 0 untuk menunjukkan adanya ketidaksamaan antara 2 kata, dan bernilai 1 untuk menunjukkan adanya kesamaan antara 2 kata.

Algoritma *Jaro-Winkler Distance* memiliki kompleksitas waktu yang sangat efektif pada *string* pendek dan dapat bekerja lebih cepat daripada algoritma *edit distance* yang lainnya. Dasar-dasar dari algoritma *Jaro-Winkler Distance* adalah sebagai berikut:

1. Hitung panjang *string*.
2. Temukan jumlah karakter yang sama dalam dua *string*.
3. Temukan jumlah transposisi.

Untuk menghitung jarak kedekatan antara dua *string*, algoritma *Jaro-Winkler Distance* menggunakan rumus sebagai berikut.

$$d_j = \frac{1}{3} \times \left(\frac{m}{s_1} + \frac{m}{s_2} + \frac{m-t}{m}\right) \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

dimana:

m = jumlah karakter/huruf yang sama persis
$|s_1|$ = panjang *string*/kata 1
$|s_2|$ = panjang *string*/kata 2
t = jumlah transposisi

Jika jarak teoritis antara dua karakter yang sama, dapat dibenarkan bila tidak melebihi:

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

Apabila mengacu pada nilai yang hendak dihasilkan oleh algoritma *Jaro-Winkler Distance*, maka nilai maksimumnya ialah 1. Nilai tersebut menunjukkan kemirkpan *string* yang telah dibandingkan dan mencapai 100% atau sama persis. Biasanya s_1 digunakan sebagai urutan acuan untuk mencari nilai transposisi. Transposisi sendiri adalah karakter/huruf yang sama dalam *string,* tetapi karakter/huruf tersebut tertukar urutannya.

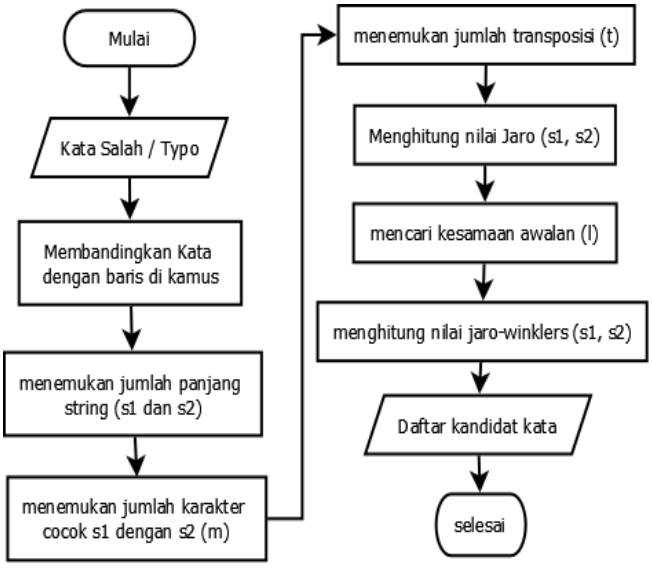
Algoritma *jaro-winkler distance* menggunakan *prefix scale* (p) dengan nilai konstanta yaitu 0.1 serta *prefix length* (l) untuk merepresentasikan panjang karakter yang sama hingga menemukan ketidaksamaan nilai yang tdak melebihi 4 karakter/huruf. Berikut rumus yang digunakan untuk menghitung *Jaro-Winkler Distance* (d_w).

$$d_w = d_j\left(l \times p(1 - d_j)\right) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

dimana:

d_j = *Jaro distance* s_1 dan s_2
l = *panjang prefix* / panjang karakter yang sama sebelum ditemukannya ketidaksamaan dengan nilai maksimal 4 karakter.
p = konstanta *scalling factor,* dengan nilai standar 0.1.

Alur proses pencarian daftar kandidat perbaikan kata menggunakan algoritma *Jaro-Winkler Distance* dapat dilihat pada Gbr. 2.



Gbr. 2. Alur Algoritma *Jaro-Winkler Distance*

E. *N-Gram*

N-Gram merupakan bahasa probabilistik yang digunakan untuk memprediksi item berikutnya secara berurutan [7]. Model *n-gram* banyak digunakan dalam teori komputasi linguistik, probabilitas, komunikasi dan kompresi data [8]. *N-gram* dikumpulkan dari sebuah teks atau korpus yang memiliki berbagai ukuran, yaitu ukuran 1 disebut *unigram*, ukuran 2 disebut *bigram*, dan ukuran 3 disebut *trigram*.

Gagasan tentang kata prediksi perbaikan diresmikan dengan menggunakan model probabilitstik yang disebut *n-gam*. *N-gram* dapat memprediksi kata berikutnya dari N-1 sebelum kata. Perhitungan probabilitas dari kata selanjutnya akan berhubungan erat dengan menghitung probabilitas urutan kata-kata.

a. Membuat Model *N-Gram*

Setelah mencari kandidat perbaikan kata, langkah selanjutnya adalah membangkitkan himpunan *bigram* kiri, *bigram* kanan dan *trigram* untuk setiap anggota jumlah elemen kata $C(K^i)$ dari setiap kata K^i dalam kalimat [10]. Tahap ini didapat dengan mengambil kata di kiri, kanan dan keduanya dari kata tersebut. Untuk K^i, maka *bigram* dan *trigram* yang terbentuk adalah sebagai berikut.

Bigram kiri: $K^{i-1}c_j^i$…….. (4)

$K^{i-1}c_j^i$ adalah kata K^i sebelum kata c_j^i.

Bigram kanan: $c_j^i K^{i+1}$…….. (5)

$c_j^i K^{i+1}$ adalah kata K^i setelah kata c_j^i.

Trigram: $K^{i-1}c_j^i K^{i+1}$ …….. (6)

$K^{i-1}c_j^i K^{i+1}$ adalah kata K^i sebelum dan setelah kata c_j^i dimana $0 \leq j \leq j_i$ yang artinya 0 tidak lebih besar sama dengan anggota kandidat kata dan tidak lebih besar sama dengan dari banyaknya anggota kandidat kata.

Setelah itu, melihat jumlah kemunculan setiap *bigram* dan *trigram* dalam korpus *n-gram* dengan rumus seperti berikut.

$count(K^{i-1}c_j^i)$ untuk jumlah kemunculan *bigram* kiri.

$count(c_j^i K^{i+1})$ untuk jumlah kemunculan *bigram* kanan.

$count(K^{i-1}c_j^i K^{i+1})$ untuk jumlah kemunculan *trigram*.

Didapatkan dari perhitungan diatas, apabila suatu *n-gram* tidak ditemukan dalam korpus, maka akan diberi nilai 0. Langkah selanjutnya adalah menghitung total jumlah kemunculan n-gram untuk semua anggota kandidat kata untuk semua kata. Rumus total jumlah kemunculan *n-gram* dapat dilambangkan sebagai berikut.

$\sum_{r=1}^{j_i} count(K^{i-1}c_j^i)$ untuk total jumlah kemunculan *bigram* kiri.

$\sum_{r=1}^{j_i} count(c_j^i K^{i+1})$ untuk total jumlah kemunculan *bigram* kanan.

$\sum_{r=1}^{j_i} count(K^{i-1}c_j^i K^{i+1})$ untuk total jumlah kemunculan *trigram*.

b. Menghitung Probabilitas *N-Gram*

Terdapat berbagai cara menghitung probabilitas *n-gram*, salah satunya adalah dengan menggunakan aturan *Markov chain.* Dalam model ini, probabilitas kalimat tidak dihitung, namun mengambil asumsi lemah bahwa kemunculan kata bergantung dengan kata sebelum dan setelah dari kata-kata lain dalam kalimat. Dalam tahap ini, menggunakan perhitungan *Maximum Likehood Estimation* (MLE), dengan menghitung jumlah kemunculan n-gram dari korpus kemudian dinormalisasi dengan membaginya dengan nilai total agar nilai berada di antara 0 dan 1. Berikut rumus-rumus yang dihasilkan dengan mengggunakan MLE.

$$P_1\left(c_j^i \middle| K^{i-1}\right) = \frac{count(K^{i-1}c_j^i)}{\sum_{r=1}^{j_i} count(K^{i-1}c_j^i)} \quad \text{…… (7)}$$

$$P_2\left(c_j^i \middle| K^{i+1}\right) = \frac{count(c_j^i K^{i+1})}{\sum_{r=1}^{j_i} count(c_j^i K^{i+1})} \quad \text{…….. (8)}$$

$$P_3\left(c_j^i \middle| K^{i-1}, K^{i+1}\right) = \frac{count(K^{i-1}c_j^i K^{i+1})}{\sum_{r=1}^{j_i} count(K^{i-1}c_j^i K^{i+1})} \quad \text{... (9)}$$

dimana P_1 adalah probabilitas bigram kiri, P_2 probabilitas *bigram* kanan, dan P_3 merupakan probabilitas *trigram*.

Rumus-rumus di atas akan mendapatkan probabilitas yang bernilai 0, maka dapat diatasi menggunakan *additive smoothing* dengan menambahkan suatu kontanta δ pada *count*. Konstanta δ yang digunakan adalah 0.01, dengan pertimbangan agar nilai yang dihasilkan tidak terlalu mempengaruhi perhitungan skor pada tahap selanjutnya. Berikut rumus-rumus probabilitas *n-gram*.

$$P_1\left(c_j^i \middle| K^{i-1}\right) = \frac{count\left(K^{i-1}c_j^i\right)+\delta}{\sum_{r=1}^{j_i} count(K^{i-1}c_j^i)+\delta+V_j} = \frac{count\left(K^{i-1}c_j^i\right)+0.01}{\sum_{r=1}^{j_i} count(K^{i-1}c_j^i)+0.01+V_j} \quad \text{…. (10)}$$

$$P_2(c_j^i|K^{i+1}) = \frac{count(c_j^i K^{i+1})+\delta}{\sum_{r=1}^{j_i} count(c_j^i K^{i+1})+\delta+V_j} = \frac{count(c_j^i K^{i+1})+0.01}{\sum_{r=1}^{j_i} count(c_j^i K^{i+1})+0.01+V_j} \quad \ldots. (11)$$

$$P_3(c_j^i|K^{i-1}, K^{i+1}) = \frac{count(K^{i-1} c_j^i K^{i+1})+\delta}{\sum_{r=1}^{j_i} count(K^{i-1} c_j^i K^{i+1})+\delta+V_j} = \frac{count(K^{i-1} c_j^i K^{i+1})+0.01}{\sum_{r=1}^{j_i} count(K^{i-1} c_j^i K^{i+1})+0.01+V_j} \quad \ldots.. (12)$$

Melakukan perhitungan untuk semua anggota kandidat kata salah untuk setiap kata.

c. Perhitungan Skor

Langkah selanjutnya adalah menghitung skor masing-masing anggota dengan menjumlahkan skor bigram dan trigram dengan menggunakan *weighted combination score*. Berikut rumus menghitung skor.

$$Score(c_j^i) = \lambda_1 P_1(c_j^i|K^{i-1}) + \lambda_2 P_2(c_j^i|K^{i+1}) + \lambda_3 P_3(c_j^i|K^{i-1}, K^{i+1}) \quad \ldots\ldots\ldots (13)$$

λ_1 merupakan bobot untuk *bigram* kiri, λ_2 merupakan bobot untuk *bigram* kanan dan λ_3 merupakan bobot untuk *trigram*. Nilai bobot terbaik yang digunakan untuk menghitung skor adalah $\lambda_1 = 0.25$, $\lambda_2 = 0.25$, dan $\lambda_3 = 0.5$. Maka rumus perhitungan skor seperti berkut ini.

$$Score(c_j^i) = 0.25\, P_1(c_j^i|K^{i-1}) + 0.25\, P_2(c_j^i|K^{i+1}) + 0.5 P_3(c_j^i|K^{i-1}, K^{i+1}) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (14)$$

F. Rencana Pengujian

Kinerja dari metode *Jaro-Winkler Distance* dan *N-Gram* untuk prediksi kesalahan penulisan kata ini diukur dengan melakukan pengukuran akurasi prediksi. Pengukuran akurasi prediksi perbaikan dilakukan dengan menggunakan persamaan sebagai berikut.

$$akurasi\ prediksi(koreksi) = \frac{kb}{kb+ks} \times 100\% \quad (15)$$

dimana,

kb : jumlah prediksi/koreksi benar yang dihasilkan oleh sistem

ks : jumlah prediksi/koreksi salah yang dihasilkan oleh sistem.

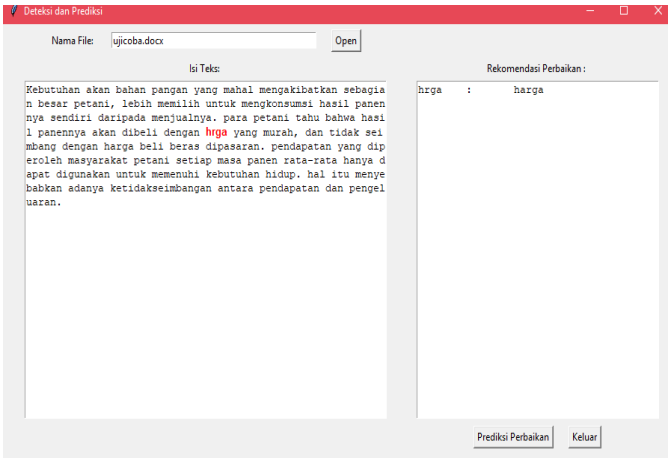
G. Teks Korpus

Teks korpus yang digunakan untuk membuat korpus *n-gram* dan deteksi kesalahan penulisan kata. Teks korpus *n-gram* dambil dari kumpulan kalimat di wikipeda dalam bahasa Indoneisa. Dokumen kospus *n-gram* tersebut berformat .docx dan berisi 42.469 kata. Teks korpus untuk deteksi kesalahan penulisan kata diambil dari Kamus Besar Bahasa Indonesia (KBBI) dan kata *unique* dari kamus n-gram yang berisi 105.234 kata dalam bentuk format .txt.

III. HASIL DAN PEMBAHASAN

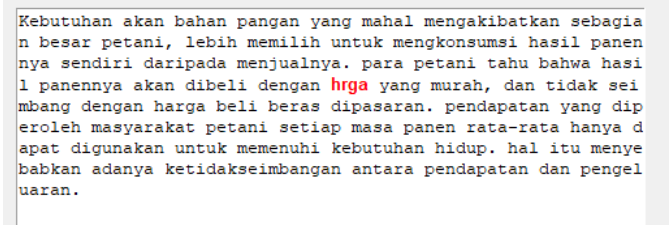
A. Implementasi Sistem

Sistem deteksi dan prediksi perbaikan kesalahan penulisan kata ini dapat digunakan untuk membantu proses memperbaiki kesalahan penulisan. Pada Gbr. 3 merupakan tampilan halaman sistem. Halaman ini terdapat kolom nama file, button open file, kolom isi teks, kolom rekomendasi perbaikan, button prediksi perbaikan dan button keluar sistem.



Gbr. 3 Tampilan Halaman Sistem

Berikut adalah contoh uji coba dari sistem deteksi dan prediksi perbaikan kesalahan penulisan kata.



Gbr. 4 Contoh Teks Pengujian

Pada Gbr. 4 merupakan contoh teks pengujian dari sistem deteksi dan prediksi perbaikan kesalahan penulisan kata. Pada contoh tersebut, terdapat kata yang teridentifikasi salah yaitu kata "hrga". Dari hasil algoritma *Jaro-Winkler Distance* mendapatkan saran rekomendasi kata yaitu kata "harga", "hara" dan "homorgan". Nilai *Jaro-Winkler Distance* antara kata

identifikasi salah "hrga" dengan daftar rekomendasi adalah sebagai berikut.

1. Nilai *Jaro-Winkler Distance* antara "hrga" dengan "harga" adalah 0.939.
2. Nilai *Jaro-Winkler Distance* antara "hrga" dengan "hara" adalah 0.775.
3. Nilai *Jaro-Winkler Distance* antara "hrga" dengan "homorgan" adalah 0.849.

Setelah dilakukannya proses pencarian daftar kandidat kata adalah penentuan rekomendasi perbaikan kesalahan penulisan kata. Proses ini menggunakan model *n-gram* yaitu sebagai berikut.

1. Membuat model *n-gram*

Pembuatan model *n-gram* untuk setiap daftar kandidat kata yaitu dengan membangkitkan *bigram* kiri, *bigram* kanan dan *trigram*. Pada tabel I merupakan hasil *bigram* kiri, *bigram* kanan, dan *trigram* dari daftar kandidat perbaikan kata dengan kalimat yang diujikan.

TABEL I
HASIL BIGRAM, DAN TRIGRAM DARI DAFTAR KANDIDAT PERBAIKAN KATA

Daftar Kandidat Kata	Bigram Kiri	Bigram Kanan	Trigram
harga	imbang harga	harga beli	imbang harga beli
hara	imbang hara	hara beli	imbang hara beli
homorgan	imbang homorgan	homorgan beli	imbang homorgan beli

Setelah membuat model *n-gram* adaah menghitung total jumlah kemunculan *bigram* kiri, *bigram* kanan dan *trigram*. Berikut pada Tabel II adalah hasil perhitungan total jumlah kemunculan *bigram* dan *trigram* data uji dalam kamus *n-gram*.

TABEL II
HASIL PERHITUNGAN TOTAL JUMLAH KEMUNCULAN BIGRAM DAN TRIGRAM DATA UJI

Daftar Kandidat Kata	Bigram Kiri	Bigram Kanan	Trigram
harga	1	1	1
hara	0	0	0
homorgan	0	0	0

2. Perhitungan Probabilitas *N-Gram*

Setelah melakukan perhitungan total jumlah kemunculan *bigram* dan *trigram* untuk data uji adalah melakukan perhitungan probabilitas *bigram* dan *trigram* untuk setiap daftar kandidat perbaikan kata. Berikut pada Tabel III adalah hasil perhitungan probablitas *bigram* kiri, *bigram* kanan dan *trigram*.

TABEL III
HASIL PERHITUNGAN PROBABILITAS BIGRAM DAN TRIGRAM

Daftar Kandidat Kata	Probabilitas *Bigram* Kiri	Probabilitas *Bigram* Kanan	Probabilitas *Trigram*
harga	0.9805	0.9805	0.9805
hara	0.0097	0.0097	0.0097
homorgan	0.0097	0.0097	0.0097

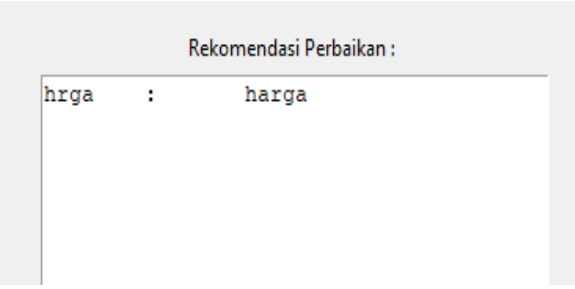
3. Perhitungan Skor

Selanjutnya adalah menghitung skor masing-masing anggota dengan menjumlahkan skor bigram dan trigram untuk masing-masing anggota dengan menggunakan *weighted combination score.* Pada tabel IV merupakan hasil skor model *n-gram* dari daftar kandidat perbaikan penulisan kata salah.

TABEL IV
SKOR N-GRAM DAFTAR KANDIAT PERBAIKAN KATA

Daftar Kandidat Kata	*Score*
harga	0.9805
hara	0.0097
homorgan	0.0097

Dari hasil perhitungan skor di atas, sistem memberikan rekomendasi terbaik dari kata salah "hrga" yaitu kata "harga" dengan skor 0.9805. Gbr. 4 menunjukkan tampilan hasil rekomendasi perbaikan kesalahan penulisan kata oleh sistem. Prediksi terbaik dari kata "hrga" yang diberikan oleh sistem adalah "harga".



Gbr. 5 Tampilan Rekomendasi Perbaikan

B. Pengujian Prediksi Perbaikan Kesalahan Perbaikan Penulisan Kata

Pengujian prediksi perbaikan kesalahan penulisan kata ini dilakukan dengan 14 data dengan

jumlah kata yang berbeda. Pengujian ini merupakan hasil prediksi perbaikan kesalahan dengan algoritma *Jaro-Winkler Distance* dan *N-Gram*. Tabel VI menampilkan 14 data persentase perbandingan jumlah prediksi perbaikan yang sesuai atau benar yang dihasilkan oleh sistem dengan jumlah prediksi perbaikan yang salah atau tidak sesuai yang dihasilkan oleh sistem. Langkah pertama melakukan penghitungan jumlah prediksi perbaikan yang benar oleh sistem, kemudian melakukan penghitungan jumlah prediksi perbaikan yang salah atau kurang tepat oleh sistem. Langkah selanjutnya adalah dilakukan perhitungan prosentase untuk mencari prosentase nilai akurasi prediksi perbaikan kesalahan penulisan kata oleh sistem.

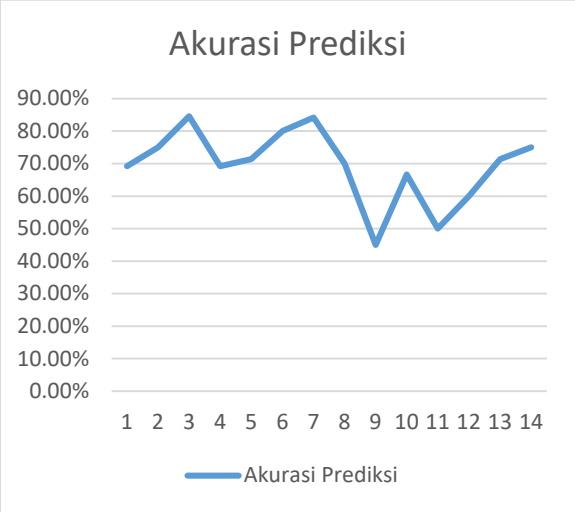
TABEL V
DATA AKURASI PREDIKSI PERBAIKAN KESALAHAN PENULISAN KATA OLEH SISTEM

Data Uji	Akurasi Prediksi
Data 1	69.2%
Data 2	75%
Data 3	84.6%
Data 4	69.2%
Data 5	71.4%
Data 6	80%
Data 7	84.2%
Data 8	70%
Data 9	45%
Data 10	66.7%
Data 11	80%
Data 12	83.3%
Data 13	71.4%
Data 14	85.7%

Tabel V telah menunjukkan hasil persentase sistem prediksi perbaikan, data 1 hingga data 10 menunjukkan hasil yang berbeda-beda. Hasil persentase tertinggi yaitu sebesar 85.7% pada data 14 dan hasil terendah sebesar 45% terdapat pada data 9. Data 11, data 12, data 13, dan data 14 merupakan uji coba dengan kata-kata yang di dalamnya terdapat salah penulisan kata dan keseluruhan kata terdapat dalam korpus. Pada data 11 terdapat 5 kesalahan penulisan kata, sistem dapat memberikan 4 kata saran perbaikan yang sesuai dengan kalimat dan 1 kata saran perbaikan yang tidak sesuai dengan kalimat. Maka hasil akurasi prediksi perbaikan data 11 tersebut adalah 80%.

Data 12 terdapat 6 kesalahan penulisan kata, sistem dapat memberikan 5 saran perbaikan yang sesuai dan 1 saran perbaikan perbaikan yang tidak sesuai dengan kalimat. Maka, hasil akurasi prediksi perbaikan kata pada data 12 adalah sebesar 83.3%. Data 13 terdapat 7 kesalahan penulisan kata, sistem dapat memberikan 5 saran perbaikan kata yang sesuai dengan kalimat dan 2 saran perbaikan yang tidak sesuai dengan kalimat. Hasil akurasi prediksi perbaikan kata pada dara 13 adalah sebesar 71.4%. Data 13 terdapat 7 kesalahan penulisan kata, sistem dapat memberikan 6 saran perbaikan kata yang sesuai dan 1 saran perbaikan kata yang tidak sesuai dengan kalimat. Data 1 hingga data 10 terdapat berbagai macam kesalahan penulisan yang terdeteksi, yaitu kata yang tidak ada dalam korpus dan kata yang benar-benar salah penulisan yang tidak sesuai dengan KBBI.

Gbr. 6 menunjukkan grafik dari hasil data uji akurasi prediksi perbaikan kesalahan penulisan kata menggunakan sistem. Dari grafik tersebut dapat dilihat bahwa nilai tertinggi akurasi prediksi yang dihasilkan oleh sistem adalah 85.7% dan nilai terendah akurasi prediksi yang dihasilkan oleh sistem adalah 45%.



Gbr. 6 Grafik Nilai Akurasi Prediksi Perbaikan Kesalahan Penulisan Kata

Dilihat dari hasil persentase 14 data uji tersebut, dapat disimpulkan bahwa sistem deteksi dan prediksi perbaikan kesalahan penulisan kata bahasa Indonesia dapat berjalan baik. Namun, dapat dilihat bahwa sistem tidak dapat menghasilkan prediksi perbaikan kata yang relevan di beberapa kata. Algoritma *Jaro-Winkler Distance* bisa mengasilkan daftar kandidat prediksi perbaikan kata yang bernilai mirip seperti kata-kata yang teridentifikasi salah. Metode *n-gram* mengambil nilai *bigram* dan *trigram* berdasarkan seluruh daftar kandidat kata yang dihasilkan oleh algoritma *Jaro-Winkler Distance*. Skor model *n-gram* yang paling tinggi akan menjadi final kandidat prediksi perbaikan kesalahan kata. Namun, jika skor *n-gram* tidak ada yang lebih dari 0,

maka *n-gram* tidak memberikan hasil final kandidat prediksi perbaikan kata dan tidak menampilkan kandidat prediksi perbaikan yang diberikan oleh algoritma *Jaro-Winkler Distance*. Kuantitas korpus *n-gram* yang digunakan sangat mempengaruhi hasil kandidat karena nilai probabilitas n-gram dari kandidat kata yang sesuai lebih rendah dari daftar kandidat kata lainnya. Karena hal tersebut, metode n-gram tidak memberikan hasil kata yang sesuai.

III. KESIMPULAN

Berdasarkan hasil perhitungan akurasi yang telah dilakukan sebelumnya, dapat diambil kesimpulan bahwa algoritma *Jaro-Winkler Distance* dan *N-Gram* untuk prediksi perbaikan kata bahasa Indonesia pada sistem ini menghasilkan nilai akurasi tertinggi sebesar 85.7% dan nilai akurasi terendah sebesar 45%. Sistem dapat mendeteksi kesalahan penulisan kata, namun sistem juga mendeteksi adanya kesalahan penulisan kata yang seharusnya tidak teridentifikasi salah. Hal ini terjadi karena kata-kata yang teridentifikasi salah tidak ada dalam korpus. Oleh karena itu, kualitas korpus untuk deteksi kata salah sangat mempengaruhi hasil deteksi kesalahan penulisan kata. Kombinasi algoritma *Jaro-Winkler Distance* dan *N-Gram* dapat digunakan untuk prediksi perbaikan kesalahan penulisan kata. Kedua metode tersebut dapat berjalan dengan baik dan bergantung pada kuantitas korpus yang digunakan. Hal tersebut dikarenakan kuantitas korpus/kamus *n-gram* yang digunakan sangat mempengaruhi hasil prediksi perbaikan. Semakin tinggi kuantitas korpus/kamus *n-gram* yang digunakan, maka semakin baik pula sistem dapat menemukan prediksi kata yang sesuai dengan kata salah pada kalimat.

UCAPAN TERIMA KASIH

Puji syukur yang senantiasa saya ucapkan kepada Allah SWT yang telah memberi kelancaran dalam mengerjakan dan menyelesaikan penelitian ini. Serta saya ucapkan terima kasih kepada semua pihak terkait yang telah memberikan bantuan dan dukungan sehingga jurnal ini dapat diselesaikan dengan baik.

REFERENSI

[1] Yeni Rochmawati, dan Retno Kusumaningrum, "Studi Perbandingan Algoritma Pencarian *String* dalam Metode *Approximate String Matching* untuk Identifikasi Kesalahan Pengetikan Teks," Jurnal Buana Informatika, hal. 125-134, 2016.

[2] Agung Prasetyo, W. M. Baihaqi, dan I. S. Had, "Algoritma *Jaro-Winkler* Distance: Fitur *Autocorrect* dan *Spelling Suggestion* pada Penulisan Naskah Bahasa Indonesia di BMS TV," Jurnal Teknologi Informasi dan Ilmu Komputer (JTIK), hal. 435-444, 2018.

[3] A. I. Fahma, I. Cholissodin, dan R. S. Perdana, "Identifikasi Kesalahan Penulisan Kata (*Typographical Error*) pada Dokumen Berbahasa Indonesia Menggunakan Metode N-*gram* dan *Levenshtein Distance*," Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, hal. 53-62, 2018.

[4] S. Supeni, dan Yusuf, "Penulisan Karya Ilmiah Sebagai Implementasi Pengembangan Kompotensi Profesi Guru Pada Guru SMP Widyawacana I," Adiwidya, hal 146-152, 2018.

[5] P. Sharma, A. L. Alai, dan A. Garg, "*Challenges and Techniques in Preprocessing for Twitter Data*," *International Journal of Engineering Science and Computing*, hal. 6611-6613, 2017.

[6] F. Okta'mal, R. Saotono, dan M. E. Sulistyo, "*Jaro-Winkler Distance* dan *Stemming* untuk Deteksi Dini Hama dan Penyakit Padi," Solo, Seminar Nasional Sistem Informasi Indonesia, hal 305-312, 2015.

[7] Wenyang Zhang, "*Comparing the Effect of Smoothing and n-gram order: Finding the best way to combine the smoothing and order of n-gram*," A Thesis Submitted to the College of Engineering at Florida Institute of Technology, 2015.

[8] V. C. Mawardi, N. Susanto dan D.S. Naga, "*Spelling Correction for Text Document in Bahasa Indonesia Using Finite State Automata and Leveinshtein Distance Method*," *MATEC Web of Conferences*, hal. 1-16, 2018.

[9] Ana Rosmiati, "Dasar-dasar Penulisan Karya Ilmiah," Surakarta, ISI Press, hal. 84, 2017.

[10] S. Fernando, D. E. Kania, "Perbandingan Metode Smoothing untuk Deteksi dan Koreksi Kesalahan Kata Dalam Teks Berbahasa Indonesia,", Universitas Komputer Indonesia, 2019.